

Inhaltsverzeichnis

Klausurthemen

Introduction	1.1
Vorbereitung	1.2
Co-Economy: Wertschöpfung im digitalen Zeitalter	1.2.1
1 Connectedness	1.2.1.1
2 Collaboration	1.2.1.2
3 Cases	1.2.1.3
4 Change	1.2.1.4
5 Conclusion	1.2.1.5
Die Digitalisierung der Welt	1.2.2
1 Industrie 4.0 oder das Industrial Internet of Things	1.2.2.1
2 Nur Science Fiction?	1.2.2.2
3 Denken Sie in Produkten - erst danach in (IT-)Prozessen	1.2.2.3
4 Aufrüsten für die digitale Zukunft	1.2.2.4
5 Bringen Sie Ihre Technik auf Vordermann	1.2.2.5
Erfolgsfaktoren für eine digitale Zukunft	1.2.3
1 Einleitung	1.2.3.1
2 Was verstehen wir unter Informationstechnologie?	1.2.3.2
3 Welchen Nutzen hat Informationstechnologie?	1.2.3.3
4 Einsatz im schlanken Unternehmen	1.2.3.4
5 Was benötigt eine erfolgreiche IT-Organisation?	1.2.3.5
6 Wie lässt sich eine IT-Organisation am besten optimieren?	1.2.3.6
7 CIO	1.2.3.7
8 Schlusswort	1.2.3.8
Soft Skills	1.2.4
1 Career Teil 1	1.2.4.1
1 Career Teil 2	1.2.4.2
2 Marketing yourself	1.2.4.3
3 Learning	1.2.4.4

4 Productivity	1.2.4.5
7 Spirit	1.2.4.6
Trusted Web 4.0 - Konzepte einer digitalen Gesellschaft	1.2.5
2 Rechtliche und organisatorische Grundlagen	1.2.5.1
3 Konzepte der Zukunft	1.2.5.2
Gemeinwohlökonomie	1.2.6
1 Kurzanalyse	1.2.6.1
2 Die Gemeinwohl-Ökonomie - der Kern	1.2.6.2
3 Die Demokratische Bank	1.2.6.3
4 Eigentum	1.2.6.4
5 Motivation und Sinn	1.2.6.5
6 Weiterentwicklung der Demokratie	1.2.6.6
7 Beispiele, Verwandte und Vorbilder	1.2.6.7
8 Umsetzungsstrategie	1.2.6.8
Zusammenfassung	1.3
1.0.0 Gesellschaft und Veraenderung	1.3.1
1.1.0 Kurzanalyse	1.3.2
1.1.1 Die Gemeinwohl-Ökonomie - der Kern	1.3.3
1.1.2 Die Demokratische Bank	1.3.4
1.1.3 Eigentum	1.3.5
1.1.4 Motivation und Sinn	1.3.6
1.1.5 Weiterentwicklung der Demokratie	1.3.7
1.1.6 Beispiel Verwandte Vorbilder	1.3.8
1.1.7 Umsetzungsstrategie	1.3.9
1.2.0 Sharing Economy	1.3.10
1.2.1 Crowdsourcing als neue Organisationsform	1.3.11
1.2.2 Beispiel: Vernetzte Arbeits- und Lebensräume	1.3.12
1.2.3 Vernetztes Wissen	1.3.13
1.2.4 Beispiel: Vernetzte Dienstleistungen	1.3.14
2.0.0 Softskills	1.3.15
2.1.0 Career	1.3.16
2.1.1 Beginnen mit einem Bang: Mache nicht, was alle anderen machen	1.3.17
2.1.2 Denk über die Zukunft nach	1.3.18
2.1.3 People Skills	1.3.19

2.1.4 Vorstellungsgespräche	1.3.20
2.1.5 Welche Beschäftigungsmöglichkeiten gibt es?	1.3.21
2.1.6 Welche Art von Softwareentwickler bist du?	1.3.22
2.1.7 Nicht alle Firmen sind gleich	1.3.23
2.1.8 In der Firma aufsteigen	1.3.24
2.1.9 Professionell sein	1.3.25
2.1.10 Freiheit: Den Job kündigen	1.3.26
2.1.11 Freelancing	1.3.27
2.1.12 Das erste Produkt	1.3.28
2.1.13 Möchtest Du ein Startup gründen?	1.3.29
2.2.0 Marketing yourself	1.3.30
2.2.1 Erschaffe eine Marke, die auf dich aufmerksam macht	1.3.31
2.2.2 Einen sehr erfolgreichen Blog kreieren	1.3.32
2.2.3 Ich kann dir keinen Erfolg garantieren	1.3.33
2.2.4 #UsingSocialNetworks	1.3.34
2.2.5 Vorträge, Präsentationen und Schulungen: Sprachkünstler	1.3.35
2.2.6 Schreibe Bücher und Artikel die eine Anhängerschaft generieren	1.3.36
2.2.7 Habe keine Angst, dumm dazustehen	1.3.37
2.3.0 Lernen	1.3.38
2.3.1 Die 10 Schritte	1.3.39
2.3.2 Mentor	1.3.40
2.3.3 Lehren	1.3.41
2.3.4 Wissenslücken	1.3.42
2.4.0 Produktivität (Productivity)	1.3.43
2.4.1 Fokus	1.3.44
2.4.2 Produktivitätsplanung	1.3.45
2.4.3 Promodoro Technik	1.3.46
2.4.4 Weitere Hinweise	1.3.47
2.7.0 Spirit	1.3.48
2.7.1 How the mind influences the body	1.3.49
2.7.2 Having the right mental attitude: Rebooting	1.3.50
2.7.3 Building a positive self-image: Programming your brain	1.3.51
2.7.4 Love and relationships: Computers can't hold your hand	1.3.52

2.7.5 Facing failure head-on	1.3.53
3.0.0 Erfolgsfaktoren für Unternehmen im Hinblick auf die Digitalisierung	1.3.54
3.1.0 Informationstechnologie	1.3.55
3.2.0 Connectedness	1.3.56
3.3.0 Collaboration	1.3.57
4.0.0 Digitalisierung	1.3.58
4.1.0 Beginn und Industrie 4.0	1.3.59
4.2.0 Transformation	1.3.60
4.3.0 Anforderungen	1.3.61
4.4.0 Auswirkungen	1.3.62
4.5.0 Die Cloud	1.3.63
4.6.0 Industriestand Deutschland & die Welt	1.3.64
4.7.0 Beispiele für aktuelle und zukünftige Technologien	1.3.65
4.8.0 Zusammenfassung	1.3.66
5.0.0 Einleitung	1.3.67
5.1.0 Rechtliche und organisatorische Grundlagen	1.3.68
5.2.0 Konzepte der Zukunft	1.3.69
X.X.X Quelle: Die Gemeinwohl-Ökonomie	1.3.70
X.X.X Quelle: Soft Skills	1.3.71
X.X.X Quelle: Erfolgsfaktoren für eine digitale Zukunft	1.3.72
X.X.X Quelle: Die Digitalisierung der Welt	1.3.73
X.X.X Quelle: Was treibt die Digitalisierung?	1.3.74
X.X.X Quelle: Trusted Web 4.0 - Konzepte einer digitalen Gesellschaft	1.3.75
Zusammenfassung: Cloud	1.4
Einführung	1.4.1
Cloud im Detail	1.4.2
Sicherheit	1.4.3
Cloud Dienste am Beispiel	1.4.4
MS Cloud Design Patterns	1.5
Einleitung	1.5.1
Cache-Aside Pattern	1.5.2
Circuit Breaker Pattern	1.5.3
Competing Consumers Pattern	1.5.4
Compute Resource Consolidation Pattern	1.5.5

Command and Query Responsibility Segregation (CQRS) Pattern	1.5.6
Event Sourcing Pattern	1.5.7
External Configuration Store Pattern yannick kloss	1.5.8
Federated Identity Pattern	1.5.9
Gatekeeper Pattern	1.5.10
Health Endpoint Monitoring Pattern	1.5.11
Index Table Pattern	1.5.12
Leader Election Pattern	1.5.13
Materialized View Pattern	1.5.14
Pipes and Filters Pattern	1.5.15
Priority Queue Pattern	1.5.16
Queue-Based Load Leveling Pattern	1.5.17
Retry Pattern	1.5.18
Runtime Reconfiguration Pattern	1.5.19
Schedular Agent Supervisor Pattern christian holzberger	1.5.20
Sharding Pattern	1.5.21
Static Content Hosting Pattern	1.5.22
Throttling Pattern	1.5.23
Valet Key Pattern	1.5.24
Asynchronous Messaging Primer	1.5.25
Autoscaling Guidance	1.5.26
Clean Code	1.6
2 Aussagekräftige Namen	1.6.1
3 Funktionen	1.6.2
4 Kommentare	1.6.3
5 Formatierung	1.6.4
6 Objekte und Datenstrukturen	1.6.5
7 Fehler-Handling	1.6.6
8 Grenzen	1.6.7
9 Unit-Tests	1.6.8
10 Klassen	1.6.9
11 Systeme	1.6.10
12 Emergenz	1.6.11

Soft Skills für Softwareentwickler	1.7
1 Projektarchitektur und Kommunikationsschnittstellen	1.7.1
2 Mit Fragetechniken zu besseren Informationen	1.7.2
4 IT-Kommunikationstypen	1.7.3
5 Konfliktmanagement	1.7.4
Soft Skills für IT-Berater	1.8
1 Beratung in der IT	1.8.1
2 Kommunizieren und verstehen	1.8.2
3 Workshops gezielt einsetzen	1.8.3
4 Workshops leiten Teil 1	1.8.4
4 Workshops leiten Teil 2	1.8.5
5 Grundtechniken für Workshops	1.8.6
6 Methodische Beratung	1.8.7
7 Methodische Beratung als Prozess	1.8.8
8 Der Sinn in unserer Arbeit	1.8.9
9 Unternehmenskultur greifbar machen	1.8.10
10 Veränderungsmanagement im Überblick	1.8.11
11 Veränderungsmanagement konkret	1.8.12
12 Werkzeuge des Veränderungsmanagers	1.8.13
13 Veränderungen und das Troja-Prinzip	1.8.14
14 Fallbeispiel	1.8.15
Soft Skills für IT-Führungskräfte und Projektleiter	1.9
2 Kommunikation	1.9.1
3 Komplexe Systeme	1.9.2
4 Selbstorganisation und Troja-Prinzip	1.9.3
5 Ziele und Prioritäten	1.9.4
6 Erfolgreiche Besprechungen	1.9.5
7 Zeitmanagement	1.9.6
8 Wie funktioniert Führung?	1.9.7
9 Kontakt und Motivation	1.9.8
10 - 19	1.9.9
SW Architecture for Developers Vol. 1	1.10
1 What is architecture?	1.10.1
2 Types of architecture	1.10.2

3 What is software architecture?	1.10.3
4 Architecture vs design	1.10.4
5 Is software architecture important?	1.10.5
7 The software architecture role	1.10.6
8 Should software architects code?	1.10.7
9 Software architects should be master builders	1.10.8
10 From developer to architect	1.10.9
11 Broadening the T	1.10.10
12 Soft skills	1.10.11
13 Software development is not a relay sport	1.10.12
14 Software architecture introduces control?	1.10.13
15 Mind the gap	1.10.14
16 Where are the software architects of tomorrow?	1.10.15
17 Everybody is an architect, except when they're not	1.10.16
18 Software architecture as a consultant	1.10.17
20 Architectural drivers	1.10.18
21 Quality Attributes (non-functional requirements)	1.10.19
22 Working with non-functional requirements	1.10.20
23 Constraints	1.10.21
24 Principles	1.10.22
25 Technology is not an implementation detail	1.10.23
26 More layers = more complexity	1.10.24
27 Collaborative design can help and hinder	1.10.25
28 Software architecture is a platform for conversation	1.10.26
30 The conflict between agile and architecture - myth or reality?	1.10.27
31 Quantifying risk	1.10.28
32 Risk-storming	1.10.29
33 Just enough up front design	1.10.30
34 Agility	1.10.31
35 Introducing software architecture	1.10.32
Enterprise Architecture at Work	1.11
Introduction to Enterprise Architecture	1.11.1
State of the Art	1.11.2

Foundations	1.11.3
Communication of Enterprise Architectures	1.11.4
A Language for Enterprise Modelling	1.11.5
Combining ArchiMate with Other Standards and Approaches	1.11.6
Guidelines for Modelling	1.11.7
Viewpoints and Visualisation	1.11.8
Architecture Analysis	1.11.9
Architecture Alignment	1.11.10
Tool Support	1.11.11
Case Studies	1.11.12
Beyond Enterprise Architecture	1.11.13

Projekte

Projekte	2.1
VR	2.2
Einführung	2.2.1
Geschichte der VR	2.2.2
Stand der Technik	2.2.3
Einsatzgebiete	2.2.4
Motion Sickness	2.2.5
AR	2.3
Einführung	2.3.1
Grundlegende Technologien	2.3.2
Konzept	2.3.3
Zusammenfassung	2.3.4
Modell getriebene Systementwicklung	2.4
Einführung	2.4.1
Systems Engineering	2.4.2
Model Based Systems Engineering	2.4.3
Systems Modeling Language	2.4.4
Metamodeling in SysML / UML	2.4.5
Product Line Engineering	2.4.6
Model Transformation	2.4.7

Quellen	2.4.8
Anhang	2.4.9
Continous Software Engineering	2.5
Agile Vorgehensmodelle	2.5.1
Unterstützende Prozesse / Workflows	2.5.2
Unterstützende Tools	2.5.3
Quellen	2.5.4
AI	2.6
Übersicht Data Mining Algorithmen	2.6.1
Knowledge Discovery in Databases	2.6.2
Datenvorverarbeitung	2.6.3
Decision Tree	2.6.4
Grundlagen	2.6.4.1
Implementierung	2.6.4.2
Bayes	2.6.5
Einleitung	2.6.5.1
Mathematische Grundlagen	2.6.5.2
Das Theorem	2.6.5.3
Rechenbeispiel	2.6.5.4
Anwendungsgebiete	2.6.5.5
Referenzen	2.6.5.6
Support Vector Machine	2.6.6
Algorithmus im Detail	2.6.6.1
Implementierungen	2.6.6.2
Embedded Computing	2.7
IOT-Protokolle	2.7.1
Einführung	2.7.1.1
MQTT	2.7.1.2
Rest	2.7.1.3
CoAP	2.7.1.4
AMQP	2.7.1.5
XMPP	2.7.1.6
Vergleich und Fazit	2.7.1.7
Quellen	2.7.1.8

IoT-Sicherheit	2.7.2
Einleitung	2.7.2.1
IoT-Sicherheitsarchitektur	2.7.2.2
Vorfälle	2.7.2.3
Distributed Denial of Service (DDoS)	2.7.2.4
Probleme	2.7.2.5
Mögliche Lösungsansätze	2.7.2.6
SmartHome	2.7.3
Einführung	2.7.3.1
SmartHome: Definitionen	2.7.3.2
Verschiedene Aspekte des Smart Home	2.7.3.3
Technische Umsetzung	2.7.3.4
Fazit	2.7.3.5
Bibliografie	2.7.3.6
Gesichtserkennung mit OpenCV	2.7.4
Einführung	2.7.4.1
Was ist OpenCV?	2.7.4.2
Haar Cascades	2.7.4.3
Eigenfaces	2.7.4.4
Fisherfaces	2.7.4.5
LBPH	2.7.4.6
Fazit	2.7.4.7
Quellen	2.7.4.8
IOT-Effizienz und IOT-Hub-Cloud (Azure)	2.7.5
Einführung	2.7.5.1
Problemstellung	2.7.5.2
About Azure	2.7.5.3
IOT-Hub virtual Machines	2.7.5.4
Storage	2.7.5.5
Business cases	2.7.5.6
Conclusions	2.7.5.7
Fullstack Development	2.8
NoSQL	2.8.1

Einführung	2.8.1.1
Spaltenorientierte DB	2.8.1.2
Dokumentenorientierte DB	2.8.1.3
Key-Value DB	2.8.1.4
Graphen DB	2.8.1.5
Objektorientierte DB	2.8.1.6
Objektrelationale DB	2.8.1.7
Fazit	2.8.1.8
Quellen	2.8.1.9
Single Page Application	2.8.2
Einführung	2.8.2.1
Entwurfsmuster	2.8.2.2
Angular	2.8.2.3
Quellen	2.8.2.4
Hybride App-Entwicklung	2.8.3
Einführung	2.8.3.1
Unterschiede zwischen Web-Apps, Native Apps und Hybride Apps	2.8.3.2
PhoneGap/Cordova	2.8.3.3
Ionic	2.8.3.4
Fazit	2.8.3.5
Quellen	2.8.3.6
Realtime Datenbanken	2.8.4
Einleitung	2.8.4.1
Technische Aspekte	2.8.4.2
Vor- und Nachteile	2.8.4.3
Einsatzmöglichkeiten	2.8.4.4
Quellen	2.8.4.5
Kommunikation in verteilten Systemen	2.8.5
Einführung	2.8.5.1
RPC	2.8.5.2
WebSocket	2.8.5.3
REST	2.8.5.4
GraphQL	2.8.5.5
Fazit	2.8.5.6

Quellen	2.8.5.7
Backend Entwicklung	2.8.6
Node.js	2.8.6.1
Grundlagen	2.8.6.1.1
Node.js Event-Loop	2.8.6.1.2
Patterns	2.8.6.1.3
Express.js	2.8.6.2
Was ist Express.js?	2.8.6.2.1
Was ist eine Middleware?	2.8.6.2.2
Beispiel-Applikation	2.8.6.2.3
Authentication	2.8.6.3
JWT vs. Session Cookie	2.8.6.3.1
External Identity Providers	2.8.6.3.2
Quellen	2.8.6.4
JavaScript Testing	2.8.7
Einführung	2.8.7.1
Testvarianten	2.8.7.2
Aufbau einer Testumgebung	2.8.7.3
Mocking Frameworks	2.8.7.3.1
E2E-testing Libraries	2.8.7.3.2
RESTful API Test	2.8.7.3.3
Quellen	2.8.7.4
Design	2.8.8
Einführung	2.8.8.1
Grundlagen	2.8.8.2
Konzepte und Designansätze	2.8.8.3
CSS-Technologien und Erweiterungen	2.8.8.4
Quellen	2.8.8.5
Cloud-Security	2.9
Einführung	2.9.1
Grundlagen	2.9.2
Einsatzgebiete	2.9.3
Cloud Management Platforms	2.9.4

Sicherheit	2.9.5
Ausblick	2.9.6
Quellen	2.9.7
Software-Architektur	2.10

Spezielle Gebiete zum Software Engineering Script

Dies ist das gemeinsam erstellte Modul Script zu den Themen, die im Sommersemester 2017 an der FH-Bielefeld im Modul behandelt wurden.

Anleitung:

- die Texte werden mit Mardown erstellt
- pro Kapitel wird eine Datei erstellt und nicht eine Datei pro Person
- pro Buch existiert eine Datei, in der die wichtigsten Eckdaten stehen und evtl. eine kleine Einleitung oder Zusammenfassung
- daraus ergibt sich folgende Struktur pro Buch:
 - buchname.md
 - buchname (Ordner)
 - buchname/kapitelnummer_kapitelname.vorname_nachname
 - Beispiel:
 - co-economy_wertschoepfung_im_digitalen_zeitalter.md
 - co-economy_wertschoepfung_im_digitalen_zeitalter (Ordner)
 - co-
 - economy_wertschoepfung_im_digitalen_zeitalter/1_connectedness.fabian_lorenz.md
 - co-
 - economy_wertschoepfung_im_digitalen_zeitalter/2_colaboration.fabian_lorenz.md
 - co-
 - economy_wertschoepfung_im_digitalen_zeitalter/3_cases.lutz_winkelmann.md
 - ...
 - assets/bildname.png (o. *.jpg ...) (Bildordner)
- um das ganze einheitlich zu gestalten folgende Formatierung innerhalb der Kapitel:
 - Kapitelüberschrift in H1 (z. B. # 3 Cases)
 - Unterkapitel in H2 (z. B. ## 3.1 Die neue Sharing Economy)
 - Abschnittsüberschriften in H3 (z. B. ### Technologien und Innovationen)
 - Bilder einfügen (z. B. ![Alternativer Bildtext] (/assets/bildname.png) (Abbildung 1: Bildbeschriftung))
- es wird nur noch mit Github gearbeitet und nicht mehr auf dem GitBook direkt
- jeder arbeitet auf einem eigenen Branch (z. B. "florenz" und nicht master branch)

- dieser wird per pull request an github geschickt und vom Admin gemerget bzw. mit Kommentar abgelehnt, falls etwas nicht passt
- auf gitbook kann man sich die aktuelle Version ansehen (sobald der merge durchgeführt wurde)
<https://www.gitbook.com/book/fh-bielefeld-mif-sw-engineerin/script/details>
- um die Seitenzahl herauszufinden, falls jemand einen Markdown Editor verwendet, der das Dokument nicht als PDF anzeigen:
 - im Chrome-Browser rechte Maustaste
 - Drucken
 - nach PDF exportieren
- **Termin: Monday 12 Jun 2017 Inhaltliche Struktur des eigenen Themas im Gitbook (10:30 Uhr)**
 - aktuelle Version vom Masterbranchen holen
 - in die Datei summary.md im unteren Bereich "Projekte" die Struktur des Projektes nach folgendem Schema kopieren und Pull Request senden:
 - Projektname (z. B. [Fullstack Development](#))
 - individuelles Thema Student1 (z. B. [NoSQL](#))
 - Unterkapitel1 (z. B. [Einführung](#))
 - Unterkapitel2 (z. B. [ObjektbasierteDB](#))
 - Unterkapitel3 (z. B. [DokumentbasierteDB](#))
 - ...
 - individuelles Thema Student2
 - ...
 - am besten macht dies einer pro Gruppe, da alle in die selbe datei schreiben müssen, um Konflikte zu vermeiden

Vorbereitungskapitel, welches die originalen Ausarbeitungen enthält.

Co-Economy: Wertschöpfung im digitalen Zeitalter

Zusammenfassung des Buches:

Titel: Co-Economy: Wertschöpfung im digitalen Zeitalter - Netzwerke und agile Organisationsstrukturen erfolgreich nutzen

Verfasser: Claudia Pelzer, Nora Burgard

Verlag: Springer Gabler

Jahr: 2014

ISBN: 978-3-658-00954-0, 978-3-658-00955-7 (eBook)

Zusammenfassung von: Fabian Lorenz, Lutz Winkelmann

Einleitung

In der heutigen Zeit ist es dank der Digitalisierung für viele kleine Unternehmen einfacher, sich auf dem Markt gegen große Wettbewerber durchzusetzen.

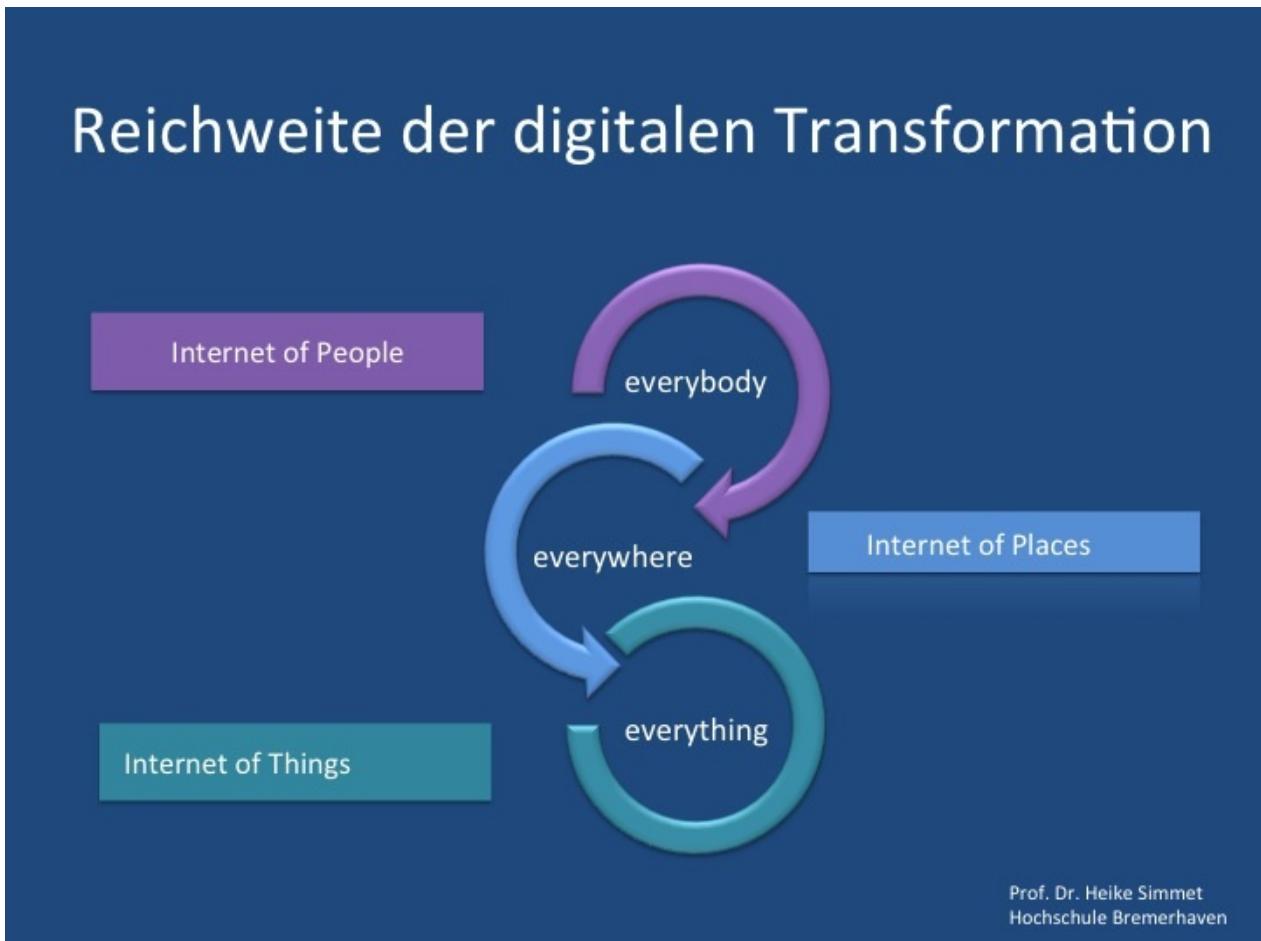
Hauptgrund dafür sind Agilität, Flexibilität und Kreativität, die von großen Unternehmen, hervorgerufen durch ihre starren Strukturen, häufig nicht so gelebt werden können, wie es bei kleinen Firmen der Fall ist.

Um den Anschluss nicht zu verlieren, ist es also notwendig, diesen Strukturellen Wandel durchzuführen und die Vorteile dieser neuen "netzbasierten Wertschöpfung" zu nutzen.

1 Connectedness

Die wichtigste Ursache für diesen Wandel ist dabei das Internet.

Es durchdringt unser gesamtes Leben und vernetzt Personen, Dinge und Orte. Ein Leben ohne Internet und diese Vernetzung ist heutzutage kaum mehr vorstellbar.



"Reichweite der digitalen Transformation." (Quelle:<http://hsimmet.com/2013/12/15/wearable-devices-neue-gamechanger-in-der-digitalen-tansformation/>)

Gerade die Sozialen Netzwerke wie z.B. Facebook und LinkedIn erfreuen sich einer großen Beliebtheit und bieten Unternehmen viele verschiedene neue Möglichkeiten, aber auch Gefahren.

Mithilfe von LinkedIn können beispielsweise neue Mitarbeiter fürs Unternehmen auf völlig neue Art gewonnen werden. Statt den ehemals klassischen Weg über Stellenausschreibungen zu gehen, können passende Kandidaten aktiv gesucht werden und somit Stellen schneller und besser besetzt werden.

Durch Facebook hingegen ändert sich die Marketingstrategie vieler Firmen. Statt in teure Marketingaktionen zu investieren, gewinnt die Mund-zu-Mund-Propaganda einen neuen Stellenwert. Durch die Kommunikation in den sozialen Netzwerken kann direkt auf den

Kunden eingegangen werden, beispielsweise durch bereitstellen von Informationen, dem beantworten von Fragen oder dem eingehen auf Beschwerden. Die Kundenbindung wird dadurch gestärkt und Kunden fangen an sich mit dem Unternehmen stärker zu identifizieren, was dazu führt, dass sie selbst das Unternehmen gegenüber anderen Mitgliedern empfehlen.

Das wichtigste Ziel für ein Unternehmen im Bezug auf Soziale Netzwerke ist dabei die Reichweite zu steigern, welche durch Kennzahlen wie Klicks, Besucherzahlen oder Followern etc. angegeben ist. Um diese Reichweite zu steigern gibt es die Taktiken des "Growth Hackings", also speziellen Taktiken ("Hacks") die darauf abzielen diese Kennzahlen zu erhöhen. Mittlerweile gibt es sogar den Job des "Growth Hackers", dessen Aufgabe es ist mit kleinen Tricks die Reichweite seines Unternehmens zu steigern.

Ein weiterer Wandel existiert bei den Business-Modellen. Der Wertschöpfungsprozess von digitalen Gütern, Services und Netzwerken wird durch folgende drei Faktoren bestimmt:

1. Niedrige Grenzkosten

Virtuelle Güter sind meist ohne weitere Kosten reproduzierbar. Zwar ist die Entwicklung von beispielsweise Online-Plattformen teuer, die Kosten einen neuen Nutzer zu integrieren tendieren jedoch gegen Null. Zudem übernehmen Nutzer auch selbst Aufgaben, wie beispielsweise das Melden von illegalen Inhalten oder als Moderator. Somit fallen auch in diesen Bereichen keine Kosten an. Ein weiterer Schritt ist, dass Nutzer sogar selbst die Inhalte liefern, wie beispielsweise bei Youtube. Die Kosten für das Unternehmen belaufen sich somit nur noch auf Entwicklung und Betrieb der Plattform.

2. Netzwerkeffekte

Der Wert eines Netzwerks orientiert sich meist an der Anzahl seiner Nutzer. Je mehr Nutzer ein Netzwerk hat, desto attraktiver ist es für eventuelle neue Interessenten. Mit jedem neuen Mitglied steigt somit der Wert des Netzwerks. Neben der Anzahl ist jedoch auch die Qualität der Teilnehmer eines Netzwerks relevant. Je mehr ein spezieller Teilnehmer des Netzwerks interessant für andere Nutzer ist, desto höher steigt auch hier der Wert des Netzwerks. (Beispiel: Größere/Bekanntere Youtuber)

3. "Long-Tail"-Effekte

Als Long-Tail Effekt wird bezeichnet, dass Anbieter im Internet ihren Gewinn nicht durch ein bestimmtes Kernprodukt erwirtschaften, sondern durch eine Vielzahl von verschiedenen Nischenprodukten.

2 Collaboration

Durch Zusammenarbeit lassen sich Ziele erreichen, die von einem einzelnen nur äußerst schwer oder gar nicht erreicht werden können. Durch die Digitalisierung sind verschiedene neue Möglichkeiten der Zusammenarbeit entstanden, so kann beispielsweise gleichzeitig von mehreren weit entfernten Orten aus an einem Projekt gearbeitet werden. Es sind eine Vielzahl von Plattformen und Tools entstanden um diese Kollaboration umzusetzen oder zu nutzen, wie beispielsweise Online-Projektmanagement-Plattformen, von Nutzern gepflegte Wikis, Instant-Messaging-Systeme und Services wie Dropbox und Google-Docs. Durch diese Möglichkeiten der (Interdisziplinären-)Kollaborationen eröffnen sich neue Innovationschancen.

Damit eine soziale Zusammenarbeit im Kreativbereich auch zu einem Erfolg führt müssen folgende Aspekte enthalten sein:

1. Dialogisch

Die Fähigkeit zuzuhören, ohne dass das Gegenüber etwas sagt; zu erspüren, was die Intention des anderen ist, wenn Sprache nicht mehr effizient ist.

2. Konjunktivische Rede

Diese Form der Kommunikation lässt Raum für Mehrdeutlichkeiten und Interpretationen. Dadurch entsteht Raum für Geselligkeit und alle Meinungen/Standpunkte können miteinbezogen werden.

3. Informeller Rahmen

Improvisation statt Verfolgen eines von vornherein vorgegebenen Ziels, um den Prozess der Kreativität nicht einzuschränken.

4. Empathie statt Sympathie

Sympathie ist das Identifizieren mit uns ähnlichen Individuen, Empathie das Verstehen anderer Personen. Sympathie kann bei der Zusammenarbeit hinderlich wirken, Empathie, also der Versuch Kollegen die anders sind als man selbst, zu verstehen, ist jedoch Horizont erweiternd.

Des Weiteren existieren Regeln, die dazu dienen sollen geteilte Ressourcen zu managen:

1. Grenzen zwischen den Nutzern und Ressourcengrenzen

Es existieren klare, lokal akzeptierte Grenzen zwischen 'legitimen Nutzern' und 'Nichtnutzungsberechtigten', sowie klare Grenzen zwischen einem spezifischen Gemeinressourcensystem und einem größeren sozio-ökologischen System.

2. Übereinstimmung mit lokalen Gegebenheiten (Kohärenz)

Die Regeln für eine Aneignung und Reproduktion einer Ressource entsprechen den örtlichen Gegebenheiten, sie überfordert die Menschen nicht und sind aufeinander

abgestimmt, das heißt sie müssen aufeinander bezogen sein. Die Verteilung der Kosten erfolgt proportional zur Verteilung des Nutzens.

3. Gemeinschaftliche Entscheidungsfindung

Teilnehmer haben Mitspracherecht an den Entscheidungen zur Bestimmung und Änderung der Nutzungsregeln eines Systems.

4. Monitoring der Nutzer und Ressourcen

Die Überwachung der Ressourcen wird entweder von den Nutzern selbst übernommen oder muss mit den Nutzern geteilt werden.

5. Abgestufte Sanktionen

Die Bestrafung von Regelverletzungen beginnt auf niedrigem Niveau und verschärft sich, wenn der Nutzer die Regel mehrmals verletzt hat. Alle Sanktionen sind dabei glaubhaft und nachvollziehbar.

6. Konfliktlösungsmechanismen

Konfliktlösungsmechanismen müssen gleichzeitig schnell, direkt und günstig sein. Es gibt lokale Räume die für die Lösung von Konflikten genutzt werden und zwischen Nutzern sowie Nutzern und Institutionen.

7. Anerkennung

Es ist ein Mindestmaß an Anerkennung des Rechts der Nutzer erforderlich, ihre eigenen Regeln bestimmen zu können.

8. Eingebettete Institutionen

Wenn beispielsweise eine Gemeinressource eng mit einem Ressourcensystem verbunden ist, sind Governance-Strukturen auf mehreren Ebenen miteinander verknüpft.

Diese kreative Zusammenarbeit bietet einem Unternehmen diverse Vorteile, beispielsweise sind Produktionsprognosen, die kollaborativ im Unternehmen erstellt wurden meist genauer, als wenn sie nur von einer verantwortlichen Person erstellt wurden.

Auch eine Zusammenarbeit mit dem Kunden und seine Einbeziehung in die Unternehmensprozesse tritt immer häufiger auf. Durch die wie bereits beschriebene Möglichkeit von Kunden beispielsweise durch soziale Netzwerke viel stärker als früher einen Einfluss auf Unternehmen zu haben, ist es notwendig auf die Wünsche des Kunden einzugehen und ihn bei der Produktentwicklung zu beteiligen. Dies hat den entscheidenden Vorteil, dass das Vertrauensverhältnis zwischen Produzent und Konsument gefördert wird.

Um diese strukturelle Kollaboration zwischen Unternehmen und Kunde zu einem Erfolg zu führen, existiert ein fünfstufiges Konzept. Der Kunde wird dabei nicht nur in einen Teil des Entstehungsprozesses miteingebunden, sondern in alle Entscheidungsprozesse der Firma, vom Brainstorming über mögliche neue Produkte über Co-Creation der Werbung, bis hin zum Gestalten des Preises.

1. Die Unternehmenskultur

Es ist darauf zu achten das die geplante Kollaboration zu der bestehenden

Unternehmenskultur passt. Das Unternehmen soll also nicht Kollaborationen angepasst werden, sondern genau andersherum. Veränderungen von firmeninternen Strukturen erfolgen langsam. Wurde eine neue Idee beispielsweise bisher von einem Mitarbeiter entwickelt, sollte dieser nicht ersetzt werden, sondern die Kollaboration sollte darauf abzielen diesen Mitarbeiter zu unterstützen. Nach mehreren erfolgreichen Kollaborationen wird sich die Kultur des Unternehmens nach und nach ändern und tiefergreifende Kollaborationen mit dem Kunden können angegangen werden.

2. Die richtigen Menschen

Nicht jeder beliebige Kunde ist für einen Kollaborationsprozess geeignet. Es sollten Kunden gefunden werden, die eine hohe Eigenmotivation haben und bestenfalls ein bestimmtes Fachwissen vorweisen und Erfahrung mit den relevanten Produkten/Dienstleistungen haben, da diese letztendlich auch die Kunden sind, die diese Produkte/Dienstleistungen kaufen/nutzen.

3. Die aktive Einbeziehung der Führungsebene

Top-Manager sollten Kollaborationsprozesse nicht nur unterstützen, sondern auch aktiv miteinbezogen werden, da ansonsten die Gefahr besteht, dass Führungskräfte Entscheidungen treffen ohne auf die Ergebnisse der Kollaboration zu achten.

4. Grenze zwischen interner und externer Kommunikation

Diese Grenze gilt es zu durchbrechen. Wenn die Kollaboration nur innerhalb der Firma stattfindet, wird niemals die maximale Reichweite erzielt werden.

5. Das Ausmaß abschätzen

Es ist notwendig, das Ausmaß der erforderlichen Arbeitsaufwände im Vorfeld abzuschätzen, sowohl firmenintern, als auch extern, um das finanzielle Risiko zu verringern. Dafür ist es erforderlich die Ziele zu definieren, Beteiligten mit einzubinden und Erwartungen zu definieren. Im Zuge der Kollaboration mit der Community ist auch die Ernennung eines Community-Managers ratsam, der für die Kommunikation zwischen Unternehmen und Community zuständig ist.

3 Cases

3.1 Die neue Sharing Economy

Bei der Sharing Economy (auch Shareconomy oder Collaborative Consumption) rückt die Nutzung von Ressourcen in den Vordergrund. Um für Umweltschutz, Effizienz und Nachhaltigkeit zu sorgen wird immer öfter nicht der Besitz eines Guts sondern nur die Nutzung dieses angestrebt. Nur selten ist ein permanenter Zugriff auf eine Ressource, welche sich als Maschine, Örtlichkeit oder etwas abstraktes darstellen lässt, nötig. In diesem Fall soll durch das Teilen der Ressource für mehr Nachhaltigkeit und Kosteneffizienz gesorgt werden.

Der Gedanke der Shareconomy wurde bereits in den 70er Jahren gefordert, passte allerdings nicht zu den Gesellschaftsbedürfnissen, welche auf den Besitz von Gütern ausgelegt waren. Heutzutage hingegen wird bereits vieles in sozialen Netzwerken geteilt und somit ist der Gedanke des Teilens bereits in der Gesellschaft verankert.

In einer Statistik von 2010 aus den USA werden folgende Werte für geteilte Güter aufgezählt:

- Wohnraum (58%)
- Arbeitsraum (57%)
- Nahrungszubereitung (57%)
- Haushaltszubehör (53%)
- Kleidung (50%)

Zu den geteilten Gütern zählt auch der Kauf oder Verkauf von gebrauchten Gütern. Durch das Internet und die damit einhergehenden Möglichkeiten ist das Teilen von Ressourcen einfacher geworden. So werden öfter Immobilien, Verkehrsmittel oder selten genutzte Dinge wie Gartengeräte vermietet und geteilt.

Das momentan bekannteste Beispiel für das Verleihen von Gütern ist in der Autobranche zu finden. Vorallem in Städten wird ein Auto oftmals nichtmehr als Statussymbol, sondern als Belastung empfunden. Durch das Mieten und Teilen von Autos ist ein riesiger, neuer Markt entstanden, welcher sich Dank des Internets und mobiler Applikationen immer stärker verbreitet.

Der Grundgedanke hinter der Shareconomy ist der, ein Gut günstig zu leihen, anstatt es teuer zu kaufen. Dieser Gedanke findet sich auch immer öfter in großen Firmen und nicht nur bei Privatpersonen wieder. Um Resourcen wie zum Beispiel Speicherplatz zu sparen, werden eBooks, Musik und andere Medien digital oftmals zum Leihen angeboten. Der Konsument hat geringere Kosten für das Leihen und beansprucht nur für kurze Zeit seinen Speicherplatz.

Die Sorge, dass mit verliehenen Gütern nicht ordnungsgemäß umgegangen wird oder dass die Güter gestohlen werden können, ist momentan der stärkste Gegner der Shareconomy. Allerdings gab es ähnliche Sorgen bereits bei der Einführung des Online-Shoppings, welche sich schnell verflüchtigten.

Es gibt kritische Stimmen gegen die Shareconomy, welche behaupten, dass das Teilen ohne den eigentlichen Besitz eines Guts nicht möglich ist, wobei die Anzahl der Güter, welche sich im Umlauf befinden, allerdings verringert würde. Auch wird davon gesprochen, dass nicht die Nachhaltigkeit oder der Umweltschutz sondern lediglich der Kostenfaktor der Antrieb für die Shareconomy sei.

3.2 Crowdsourcing als neue Organisationsform

Der Begriff Crowdsourcing beschreibt den Zugriff auf ein gemeinschaftliches Wissen und personelle Ressourcen, um ein Problem zu lösen.

Unterkategorien des Crowdsourcing sind unter anderem:

- Crowdfunding (Finanzierung eines Projekts durch die Community)
- Co-Creation (Erschaffung eines Werkes durch die Community)
- Microworking (Teilaufgaben eines Projekts werden durch die Community erledigt)

Durch das Internet zählt die Crowdsourcing-Branche aktuell zu den am schnellsten wachsenden. Der Zugriff auf eine immer größer werdende Community, in welcher alle Arten von Fähigkeiten abgedeckt sind, wird durch Plattformen im Internet ermöglicht. Teams können Aufgaben zur Fertigstellung eines Projekts durch einfache Kommunikationsmöglichkeiten unabhängig von räumlichen und zeitlichen Gegebenheiten realisieren.

Die hohe Nachfrage an Crowdsourcing Dienstleistungen basiert auf folgenden sechs Grundprinzipien:

1. Die Anforderungen an technische und soziale Kompetenzen steigen.

2. Durch moderne Kommunikationstechnologien sind Personen nicht an Freizeit und Arbeitszeit gebunden.
3. Produkte werden durch Ideen, Innovationen und Informationen ersetzt.
4. Arbeitsszeiten und Löhne können flexibler an die Fähigkeiten der Personen angepasst werden.
5. Wertschöpfungsketten werden transparenter und der Konsument erlangt Eingriff in diese.
6. Durch Crowdfunding werden Produkte von Konsumenten vorfinanziert.

Im Gegensatz zu Outsourcing ist Crowdsourcing nicht an feste Arbeitszeiten und einen festen Arbeitsplatz gebunden. Außerdem sind die einzelnen Teammitglieder einfacher auszutauschen und die Bezahlung erfolgt nach dem Festpreisprinzip anstatt auf Stundenlohnbasis.

Die Motivation für Crowdsourcing erstreckt sich über viele Bereiche, wobei der materielle Ansatz ein sehr geringer ist. Ein Beispiel für Crowdsourcing ist Wikipedia, wobei Leute ihr Wissen teilen um selber von dem Wissen anderer profitieren zu können. Auch Wohltätigkeit, Spaß und Langeweile können motivieren. Geld ist nicht die beste Motivation für Crowdsourcing Projekte, da die Bezahlung oftmals nicht besonders hoch ist und die Teilnehmer dementsprechend auch nur einen Teil ihres Könnens einsetzen.

Für Unternehmen sind gibt es mehrere interessante Vorteile. Im Folgenden werden einige davon aufgelistet:

- Austausch von Wissen
- erweiterter Ideenpool
- Einbeziehung geographischer Besonderheiten
- Ressourcenersparnisse durch Spezialisierung

3.3 Beispiel: Vernetzte Arbeits- und Lebensräume

Vor allem in Städten ist der Effekt der Shareconomy deutlich zu spüren. Sei es bei dem Teilen von Büroflächen, bei Wohngemeinschaften oder bei der Autovermietung.

Der Begriff Co-Working beschreibt das Mieten eines Arbeitsplatzes mit Internetanschluss und einer Postadresse. Dieses Konzept ist vor allem bei Startups und Freelancern beliebt, da Kosten minimiert werden können, die Arbeiter nicht zu sehr abgelenkt werden wie wenn sie von zu Hause arbeiten würden und da der Austausch mit anderen Mieter möglich ist, wodurch ein professionelles Netzwerk aufgebaut werden kann.

Unter dem Begriff Co-Living wird das Teilen einer Wohnung verstanden. Vorallem für Städtetrips ist diese Hotelalternative besonders interessant, da Geld gespart werden kann und der Austausch zwischen verschiedenen Kulturen gefördert wird.

3.4 Beispiel: Vernetztes Wissen

Sogar für das Wissen an sich lassen sich Unterschiede zwischen alter und neuer Denkweise finden. Früher konnte Wissen als persönlicher Besitz angesehen werden. Oftmals blieben Arbeiter bis zur Rente nur in einer Firma, da sie sich in dieser Branche auskannten und der Umstieg in eine andere Firma eine Art neue Ausbildung mit sich ziehen würde. Heute ist dies zum Teil immernoch so allerdings wird das Wissen heutzutage vermehrt geteilt. Im Internet findet man leicht Vorlesungen von Professoren oder auch Berichte von Experten, wodurch es einfacher ist, am technologischen Wandel teilzuhaben.

Der Begriff Co-Learning stellt das Lernen geteilten Wissens in einer Gruppe unabhängig vom Ort dar. So ist es beispielsweise möglich, Nischenangebote zu erstellen, welche in einer Universität nur sehr wenige Studenten zum teilhaben bewegen können, allerdings weltweit eine große Anzahl an Interessenten finden.

Foren und Wikis dienen als Wissens- und Datensammlungen, welche auf einer technologischen Organisationskultur aufbauen, ein nötiges Maß an Vertrauen und Transparenz vermitteln und die Beitragileistenden in Kategorien einstufen.

Das Wissen, welches in diesen Sammlungen geteilt werden soll kann entweder explizit oder implizit sein.

Folgende Punkte treffen für das explizite Wissen zu:

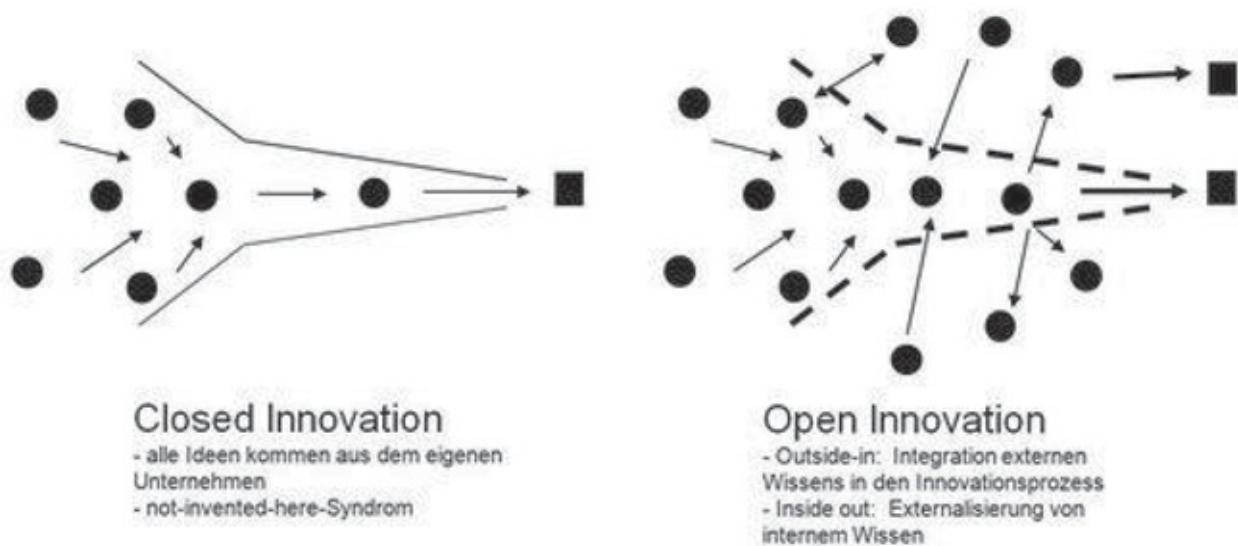
- Der Wissensinhaber kann die Information beschreiben
- Der Wissensempfänger muss Kenntnis über das Vorhandensein der Information haben

- Der Zugang zu den geteilten Informationen muss für den Wissensempfänger möglich sein
- Die Wissenssammlung muss strukturiert werden

Implizites Wissen hingegen kann vom Wissensinhaber nicht artikuliert werden. Hierunter fallen beispielsweise Fähigkeiten wie das Fahrradfahren. Der Austausch von implizitem Wissen kann somit nur in informellen Netzwerken stattfinden.

Auch innerhalb von Unternehmen ist ein Austausch von explizitem und implizitem Wissen oft nötig und kann durch ein Wiki abgewickelt werden, welches unter anderem Produktbeschreibungen, Anwenderhandbücher oder Allgemeines Markt- und Branchenwissen beinhalten kann. Hierdurch wird die Kommunikation oft vereinfacht, da der Zugang für alle betreffenden Personen möglich ist.

Durch Co-Creation und Open Innovation können auch externe Personen am Prozess der Entwicklung eines finalen Produkts in einem Unternehmen teilhaben. Impulse können hierbei von innen und von außen angestoßen und Informationen von innen nach außen weitergegeben werden. Dieser Prozess wird in der folgenden Abbildung dargestellt und wird als Crowdstorm (Brainstorming der Crowd -> Zusammensetzung von Brainstorming und Crowdsourcing) bezeichnet.



Je mehr sich die Unternehmen in diesem Prozess der Crowd öffnen, desto innovativer werden schlussendlich die Produkte, da die internen Mitarbeiter sich ansonsten in einer Art Blase befinden und sich vor externen Anreizen verschließen.

3.5 Beispiel: vernetzte Dienstleistungen

Durch die Digitalisierung konnte das Angebot und die Nachfrage von Dienstleistungen ins Internet verschoben werden und somit können Zeit- und Ortsunabhängig Spezialisten für gewisse Problemstellungen gefunden werden. Hierfür sind Online-Dienste wie Auftrags-Datenbanken oder das Micro-Payment unablässlich.

Es gibt verschiedene Tools, welche die ortsunabhängige Zusammenarbeit von Teams unterstützen wie beispielsweise Skype. Soll die Entwicklung von Projekten auch zeitunabhängig sein, so kann auf Dienste wie Google Drive zurückgegriffen werden. Diese Tools umfassen die Hauptaufgaben Online-Projektmanagement, Datentransfer und Kommunikation und werden oftmals gemeinsam eingesetzt.

Nicht nur Dienstleistungen werden vermehrt über das Internet angeboten, sondern auch Produktionsvorgänge werden immer mehr digitalisiert. Durch die Einführung des 3D-Drucks werden viele druckbare Objekte über das Internet erreichbar gemacht und können somit zu Hause erstellt werden. Natürlich ist auch das Teilen eines Objekts mit der Community möglich. Hierdurch werden Innovationen ins heimische Arbeitszimmer verlegt und somit werden Produktionsabläufe grundlegend beeinflusst.

Auch das Finanzwesen ist von der Co-Economy betroffen. Im Internet werden bereits viele Zahlungsmöglichkeiten wie Paypal angeboten. Auch Crowdfunding ist durch das Internet bekannt geworden und somit konnten bereits viele Innovationen umgesetzt werden.

Mittlerweile hat sich sogar digitales Geld in Form von Bitcoins im Finanzwesen durchgesetzt. Der Vorteil liegt hierbei darin, dass Geldbeträge banken- und zeitunabhängig den Besitzer tauschen können. Außerdem sind die Transaktionen sehr sicher und machen Rückbuchungen unmöglich.

Zur Zeit ist die Bitcoin allerdings noch sehr instabil im Kurs und der mögliche Diebstahl durch Hacker wird ebenfalls oft als ein hohes Sicherheitsrisiko angesehen.

Das Prinzip Crowdfunding ist eine Form des Crowdsourcings, in welcher Geld für Projekte gesammelt werden kann. Das Geld kommt hierbei von der Community welche automatisch einen gewissen Einfluss in die Entwicklung des Produkts bekommt.

Beim Crowdfunding kann durch die Analyse der zahlenden Community automatisch die Zielgruppe und der mögliche zu erschließende Markt erforscht werden.

Die zahlende Community kann das Geld entweder spenden, wodurch dieses nicht versteuert werden muss, oder allerdings eine materielle Gegenleistung erwarten.

Die Begriffe Crowdinvesting und Crowdloaning fallen unter den Begriff
Crowdfunding.

Das Crowdinvesting stellt hierbei die klassische Finanzierung eines Startups
dar, während beim Crowdloaning Geld an eine Privatperson in Form eines Kredits
fließt. Die Privatperson zahlt das erhaltene Geld dann verzinst an die
geldgebende Community zurück.

4 Change

Eine weitere Auswirkung des Wandels sind Änderungen beim Beschäftigungsverhältnis. Wie bereits erwähnt können Firmen dank der Digitalisierung selbständig, beispielsweise durch soziale Netzwerke, neue Arbeitskräfte finden, die in der heutigen Zeit flexibel und lernfähig sein müssen, dies aber auch vom Unternehmen fordern. Neue und flexiblere Regelungen bezüglich der Arbeitszeit sind dabei ein Aspekt und auch Lösungen wie das Home-Office Prinzip entstehen. Zudem gibt es immer mehr Personen die selbständig, beispielsweise auch als Freiberufler arbeiten und klassische Hierarchische Aufstiegschancen ablehnen. Arbeitnehmer suchen heute viel intensiver nach Arbeit die man tun möchte, Unternehmen müssen daher flexible Strukturen garantieren, sodass individuelle Potentiale voll ausgelebt werden können. Dies hat jedoch auch den positiven Effekt, dass Arbeitnehmer deutlich motivierter sind, auch ohne klassische Anreize wie gute Bezahlung.

Auch auf Business-Modelle hat die Digitalisierung Einflüsse, häufig geht es nicht mehr darum selbständig bestimmte Produkte zu vertreiben, sondern darum, Plattformen aufzubauen die von Personen genutzt werden und mit Revenue-Share Modellen Gewinn einbringen. Beispielsweise kann dies bei Verkaufsplattformen so umgesetzt werden, dass Nutzer selbständig Waren handeln können, wobei ein bestimmter Prozentsatz jeder Transaktion an den Betreiber der Plattform abgegeben wird. Hauptgründe für die Probleme klassischer Geschäftsmodelle sind dabei die freie Verfügbarkeit von Produktionsmitteln, freier Zugang zu Wissen und Dienstleistungen und der individuelle Wertewandel im Arbeits- und Sozialleben.

Entsprechende neue Business-Modelle haben daher meist ein Wertschöpfungsnetzwerk statt einer klassischen Wertschöpfungskette, sie haben offene Strukturen, sind kommunikativ und die Grenzen zwischen Produzenten, Konsumenten und Investoren verschwimmen.

Häufig spielen auch nicht mehr klassische Rohstoffe eine Rolle, sondern es geht viel häufiger um Daten, Wissen und (Vertrauens-)Beziehungen. Bezeichnet wird dies durch den Begriff Social Currency. Wie bereits erwähnt steigt beispielsweise der Wert eines sozialen Netzwerkes unter anderem durch seine Nutzer. Auch die angesprochene angestrebte Reichweite eines Unternehmens ist hierbei zu nennen. Je offener sich ein Unternehmen sich gegenüber dem Kunden verhält und je mehr mit ihm kommuniziert wird, desto besser wird seine Social Currency. Aber nicht nur für Unternehmen selbst spielt Social Currency eine Rolle, sondern beispielsweise auch für jeden Nutzer des Netzwerkes selbst. Je besser die eigene Social Currency ist, desto höher ist der Stellenwert der eigenen Person im Netzwerk. Konkret bedeutet dies zum Beispiel, dass man bei einem Verkaufsportal deutlich bessere Chancen hat Artikel zu verkaufen, wenn man bereits viele andere Verkäufe im Netzwerk

durchgeführt hat und für diese positive Rückmeldungen von anderen Nutzern bekommen hat. Hat man allerdings nur negative Rückmeldungen erhalten, oder hat man nur wenig verkauft, sinkt die Glaubwürdigkeit.

5 Conclusion

Die Veränderung durch die Digitalisierung findet bereits statt und Unternehmen müssen schnellstmöglich beginnen sich ihr anzupassen, oder sie laufen Gefahr ins Vergessen zu geraten. Geschäftsmodelle müssen angepasst, Organisationsprozesse verändert und kollaborative Strukturen eingerichtet werden. Firmen müssen flexibler, offener, agiler und vor allem Kreativer und Innovativer werden um auf schnelle Veränderungen angemessen agieren zu können. Dabei gilt: nicht der stärkste oder größte gewinnt, sondern der Anpassungsfähigste. Um diesen Wandel zu vollziehen lassen sich drei Handlungsebenen definieren:

1. Ebene 1 Nutzen von externen (Kreativ)-Ressourcen, Dienstleistungen und entsprechenden Plattformen für die unternehmenseigene Zwecke nutzen, um Innovation und Flexibilität zu erlernen und fördern.

Beispiel: Nutzen von Crowdsourcing-Kampagnen für die Ideenentwicklung, Auslagern von Tätigkeiten in die Crowd oder Aufbau von Kooperationen mit Start-Ups zwecks Wissenstransfer

2. Ebene 2

Anpassen des eigenen Geschäftsmodells an die digitale Co-Economy in Form von Pilotprojekten oder Prototypen, während das Kerngeschäft weiter läuft.

Beispiel: Ein Unternehmen transformiert sein Kerngeschäft, im Ansatz ist dies bereits bei Automobilherstellern zu beobachten, die in den Carsharing Markt eingestiegen sind.

3. Ebene 3

Ein kompletter strategischer Wandel des unternehmenseigenen Geschäftsmodells, welcher über mehrere Jahre hinweg erfolgt und einen integrierten Gesamtplan erfordert.

Wenn Unternehmen all diese notwendigen Änderungen beherzigen und dementsprechend handeln, werden sich ihnen eine Vielzahl von neuen Möglichkeiten und Chancen bieten, die ihnen in der Zukunft zu großem Erfolg verhelfen können.

Die Digitalisierung der Welt

Zusammenfassung des Buches:

Titel: Die Digitalisierung der Welt - Wie das Industrielle Internet der Dinge aus Produkten Services macht

Verfasser: Peter Samulat

Verlag: Springer Gabler

Jahr: 2017

ISBN: 978-3-658-15510-0, 978-3-658-15511-7 (eBook)

Zusammenfassung von: Nils Kohlmeier, Tolga Aydemir, Yannick Kloss

1 Industrie 4.0 oder das “Industrial Internet of Things”

Industrie 4.0

Der Begriff “Industrie 4.0” bezeichnet die vierte industrielle Revolution. Bei der ersten industriellen Revolution, die Ende des 18. Jahrhunderts begann, wurden erstmals mechanische Produktionsanlagen eingesetzt, die Wasser- und Dampfkraft verwendeten. Dieser Abschnitt wird “Industrie 1.0” genannt. Die darauffolgende industrielle Revolution (“Industrie 2.0”) begann Anfang des 20. Jahrhunderts und führte die arbeitsteilige Massenproduktion ein. Maschinen wurden hierbei erstmals mit elektrischer Energie betrieben. Nachdem in den 1970er Jahren Elektronik und IT soweit waren, um in der Industrie eingesetzt zu verwenden, konnte die Produktion von Gütern (teil-)automatisiert werden. Dies wird als “Industrie 3.0” bezeichnet.

Heute soll die Industrie 3.0 zur Industrie 4.0 umgebaut werden, um die gesamte Wertschöpfungskette zu automatisieren, damit die Effizienz der Produktionsanlagen weiter gesteigert werden kann. Weiterhin soll die Qualität der Produkte verbessert und die Wettbewerbsfähigkeit der deutschen Industrie gesteigert werden. Um eine selbstorganisierende Produktion zu etablieren, ist es notwendig, dass Produktionsprozesse untereinander kommunizieren. So ist eine Unternehmen zum Beispiel in der Lage, flexibel auf Änderungen bei der Ressourcenverfügbarkeit zu reagieren.

Durch die automatisierte Organisation der Fertigung kann gut und effizient auf Kundenwünsche reagiert werden und individuelle Produkte können kostengünstig hergestellt werden.

Durch die Anbindung des Internet of Things (IoT), welches hauptsächlich die Gesamtheit der “intelligenten” elektronischen Geräte (Wearables, Haushaltsgeräte, Autos, Unterhaltungselektronik, ...) beim Endverbraucher/Kunden beschreibt, kann auf Aktionen von Endverbrauchern reagiert werden.

Die aktuelle Situation in Sachen “Industrie 4.0” sieht in Deutschland nicht gut aus. Seit einiger Zeit gehören Industrie-4.0-Produkte zwar zum Standardsortiment der Fabrikausrüster doch die Unternehmen sind bei der Umsetzung und Implementierung von Industrie-4.0-Prozessen sehr zögerlich. Einige Unternehmen setzen schon Industrie-4.0-Produkte ein, hauptsächlich aber nur mit Fokus auf Effizienzsteigerung. Die Forschung, die Arbeitsgruppen und die verschiedenen Gremien treiben dagegen die Standardisierung von Prozessen und Schnittstellen voran.

Bei der Umstellung der Industrie auf die Industrie 4.0 werden nicht nur unternehmensinterne Prozesse umgestellt, sondern auch nach außen hin werden Informationen und Dienstleistungen verknüpft. Somit kann mit den Kunden digital interagiert werden, was genutzt werden kann, um neue Produkte und Dienstleistungen zu entwickeln. Um Kosten zu sparen, werden (Teil-)Prozesse zum Kunden ausgelagert: Sie erledigen zum Teil Arbeiten, die früher das Unternehmen gemacht hat. Einige Beispiele sind zum Beispiel Selbstbedienungskassen im Supermarkt, bei denen der Kunde seine Artikel selbst einscannt, das Buchen und Einchecken von Flügen über Online-Portale sowie das Tätigen von Überweisungen bei der Bank, welches von Kunden selbst vorgenommen werden kann. Für die Umsetzung ist es notwendig, die Prozesse soweit zu vereinfachen, dass diese von jedem durchgeführt werden können.

Treiber der digitalen Transformation sind folgende: Digitalisierung, Vernetzung, Mobilität und Analytik. In allen Bereichen des Lebens werden immer mehr Dinge digitalisiert und in Software umgesetzt. So hat man zum Beispiel heute Smartphones, die mit den richtigen Apps viele verschiedene Aufgaben lösen können, für die man früher noch eigenständige Geräte brauchte. Dadurch werden aber auch neue Möglichkeiten entdeckt, die das Leben beeinflussen. Viele Dinge werden zudem heute erst im Computer designet und anschließend gefertigt. Kunden haben hiermit die Möglichkeit, ein Produkt noch vor der Fertigung zu betrachten.

Die zunehmende Vernetzung nicht nur von Menschen sondern auch von Geräten ermöglicht die Steuerung der Produktion in Echtzeit. Ein weiterer Punkt ist die Mobilität. So sind viele Menschen heute auch unterwegs online, um auch an Informationen zu kommen und austauschen zu können, wenn sie nicht zu Hause sind. Die Industrie 4.0 soll ermöglichen, auch unterwegs auf den Status der Produktion zugreifen zu können.

Die Aufgabe der Analytik ist, die gesammelten Daten zu analysieren und entsprechende Kenntnisse daraus zu gewinnen. Die Technik hierfür steht noch ganz am Anfang. Am Ende sollen die Daten Handlungsempfehlungen liefern. Ein vielversprechender Weg ist hierbei die Künstliche Intelligenz.

Auch das Thema Sicherheit ist wegen der Digitalisierung ein wichtiger Punkt bei der Industrie 4.0. Wenn immer mehr Geräte digitalisiert und vernetzt werden, müssen diese Geräte und Netze vor Angriffen geschützt werden.

Smarte Eingabegeräte

Wearables, also kleine tragbare elektronische Geräte, sind derzeit sehr beliebt und können gut in Unternehmen eingesetzt werden. Mögliche Anwendungsszenario für Wearables in Unternehmen sind zum Beispiel der Einsatz von "smarten" Handschuhen, die Monteure bei der Montage anweisen und Datenbrillen, die dem Träger wichtige Informationen einblendet.

Heutige Wearables enthalten immer mehr Sensoren, um möglichst viele Daten erfassen zu können. Auch das direkte Integrieren von Wearables in Kleidung ist derzeit sehr beliebt. Ein sehr wichtiger Faktor bei dem Erfolg der Wearables ist die Bedienbarkeit der Geräte.

Die Forschung zum Thema "Mensch-Maschinen-Kommunikation" ist sehr aktiv und es werden immer neue Eingabe- und Steuerungsmethoden entwickelt und alte ständig verbessert. So sind heute zum Beispiel schon Gestensteuerungen in Autos und Unterhaltungselektronik zu finden. Ein bekanntes Gerät ist unter anderem die Microsoft Kinect, welche die Positionen und Bewegungen von Personen mithilfe einer 3D-Kamera erkennt.

Auch beliebt sind Sprachassistenten, die per Sprache gesteuert werden und allerhand nützliche Informationen bereitstellen sowie Smart-Home-Geräte steuern. Beispiele für Sprachassistenten sind "Alexa" von Amazon und "Google Home" von Google. "Google Home" soll, um möglichst hilfreich zu sein, mit den verschiedenen Google-Diensten wie Google Mail und Google Maps verknüpft werden. Die Steuerung der Heimautomatisierung "Google Nest" mit Google Home ist (vorerst) nur in den USA verfügbar. Die Smartphone-App "Google Assistant" verwendet zudem künstliche Intelligenz um gesprochene Wörter zu erkennen.

Neben den Sprachassistenten gibt es zudem noch die "Eye Tracker", die Augenbewegungen mithilfe von Kameras und Bildverarbeitungsalgorithmen erkennen und so eine Steuerung von Geräten über die Augen ermöglichen.

Einen anderen Steuerungsansatz untersucht das Projekt "Skin Track": Die eigene Haut soll als Touchscreen dienen. Die Technik "Thermal Touch" erkennt über Wärmebildkameras Flächen, die von einem Benutzer durch Anfassen erwärmt wurden. So können beliebige Gegenstände, die sich erwärmen lassen, zu Eingabegeräten werden.

Visualisierung

Es gibt zwei Visualisierungsarten: "Virtual Reality" (VR) und "Augmented Reality" (AR). Der Unterschied der beiden Visualisierungsarten liegt dabei an der Einbindung der Umgebung: Während bei der Augmented Reality die Umgebung mit zusätzlichen Informationen ergänzt wird bzw. reale mit virtuellen Elementen kombiniert werden, ist der Nutzer bei der Virtual Reality von der realen Umgebung komplett abgeschottet und vollständig in einer virtuellen Umgebung.

Schon Anfang der 1990er Jahre war Virtual Reality ein großes Thema. Heute helfen aber verbesserte Sensoren und eine höhere Rechenleistung dabei, Virtual Reality besser umzusetzen als vor 20 Jahren. Verschiedene empfindliche Sensoren ermöglichen es

aktuellen VR-Headsets, Kopfbewegungen genauestens zu erkennen und die Position des Benutzers zu bestimmen. Im Gegensatz zu Wearables haben VR-Applikationen andere Anforderungen an das Benutzerinterface: Die Benutzung sollte hauptsächlich mit den Händen erfolgen und so sehr intuitiv sein.

Bei der Augmented Reality wird die reale Umgebung durch virtuelle Elemente ergänzt. Die wird zum Beispiel bei den so genannten "Smart Mirrors" gemacht, die Informationen wie Nachrichten und das aktuelle Wetter direkt im Spiegel einblenden.

Neben den Smart Mirrors gibt es noch die Datenbrillen. Eine gutes Beispiel ist hierfür die "Google Glass", die in den Medien viel Aufmerksamkeit erregt hat. Die erste Version der Datenbrille wurde vom Markt genommen, aber Google arbeitet derzeit an einer "Enterprise Edition" der Brille, die zum Beispiel im Logistikbereich eingesetzt werden soll.

Die "HoloLens" von Microsoft dagegen unterstützt 3D-Projektionen, welche durch projizierte Lichtpunkte erzeugt werden.

Eine dritte Art von AR-Anwendungen sind Augmented Reality Browser, die als App auf einem Smartphone oder Tablet laufen und bestimmte Muster oder Bilder erkennen und dann entsprechende Informationen anzeigen. AR-Browser gibt es auch als "Location Based Services", welche anhand des aktuellen Standortes des Benutzers Informationen anzeigen.

Smarte Gebäude

In Smart Homes sind alle Geräte miteinander vernetzt und können automatisiert Umgebungsparameter wie etwa Temperatur und Licht steuern. So weiß das System beispielsweise, wann der Eigentümer nach Hause kommt und kann entsprechend die Temperatur in den Räumen anpassen, bevor der Eigentümer nach Hause kommt.

Neben den Smart Homes gibt es noch die sogenannten "Smart Rooms", welche von Einzelhändler eingesetzt werden, um die Bereiche im Laden zu erkennen, die von vielen Kunden aufgesucht werden. Dies wird unter anderem mithilfe von Wärmebildkameras realisiert.

Des Weiteren gibt es Bluetooth Beacons, welche sich über Bluetooth-Low-Energy (BLE) mit auf den Smartphones der Kunden installierten Laden-App verbinden und zum Beispiel Nachrichten über aktuelle Angebote senden.

Die Smart Factory ist eine Fabrik, in der alle Maschinen miteinander vernetzt sind und automatisiert alle notwendigen Arbeitsabläufe geplant und ausgeführt werden.

In intelligenten Städten ("Smart Cities") sollen Verkehr, Ver- und Entsorgung und Logistik automatisiert gesteuert werden. Ziel ist es, das Leben bequemer zu machen. Werden alle Vorgänge in einer Stadt automatisiert, gehören zum Beispiel Staus der Vergangenheit an.

2 Nur Science-Fiction?

Die Entwicklung von neuen Technologien ist rasant. Schon heute sind die Technologischen Tendenzen zu beobachten, welche sich erst in einigen Jahren etablieren. Ein gutes Beispiel ist die Entwicklung der Smartphones und was sich innerhalb von wenigen Jahren entwickelt hat. Es ist wichtig zu beobachten, in welche Richtungen sich Technologien entwickeln und was in den nächsten Jahren Marktreife erlangen könnte und welche Auswirkungen diese haben werden.

Technologien und Innovationen

Nachfolgend werden einige Beispiele von neuen und interessanten Technologien aufgelistet, welche in der Zukunft eine große Rolle spielen könnten:

- **Kristallkerne als Datenspeicher:** Mithilfe von speziellen Lasern werden Daten in Kristallkerne eingebrannt. Die kristallinen Strukturen werden dabei neu angeordnet. Die Daten werden in Form von Punkten in Schichten gespeichert. Somit können neben den drei Raumdimensionen auch die Größe und Orientierung der Punkte genutzt werden. Dies wird von Forschern der University of Southampton als 5D-Speicher bezeichnet. Dies könnte zukünftig eine neue Form der Langzeitdatenspeicherung sein..
- **Aquaman-Kristall:** Die University of Southern Denmark hat im Rahmen des sogenannten Aquaman- Kristall-Projekts ein Material entwickelt, welches Sauerstoff aus der Luft oder aus dem Wasser in hoher Konzentration speichern kann. Der Sauerstoff kann bei Bedarf auch unter Wasser wieder abgegeben werden. Der Aquaman-Kristall speichert so viel Sauerstoff, dass man über einen langen Zeitraum unter Wasser bleiben kann. Eine Teelöffel große Menge kann Sauerstoff eines mittelgroßen Raums speichern.
- **Graphen:** Das im Jahr 2004 erstmals hergestellte Material Graphen besteht aus einer einzigen Lage von Kohlenstoffatomen. Es ist sehr dünn, widerstandsfähiger als Stahl, leitfähig, biegsam und durchsichtig. Beispielhafte Anwendungen könnten sein: Faltbare berührungsempfindliche Displays, Schaltungen auf unterschiedlichen Produkten (z. B. Tasse, Milchpackung usw.) zum interagieren mit der Umwelt und intelligente Kontaktlinsen mit Zoom. Die Zahlen sprechen für sich: Im Jahr 2014 wurden über 9000 Patente mit Graphenbezug angemeldet.
- **Smart Glass:** Ein Material, welches beim Anlegen einer Steuerspannung die Transparenz ändert. Viele Anwendungsfälle sind denkbar. Die Entwicklung geht in Richtung immer größerer Flächen und geringerer Kosten. Es ist vorstellbar das in der Zukunft ganze Gebäudeflächen mit diesem intelligenten Glas versehen werden um

tagsüber Wärme zu speichern und nachts wieder abzugeben.

Ein weitere zu beobachtende Entwicklung ist die der **künstlichen Intelligenz**. Das folgende Zitat ist von Volkswagen Group CIO Martin Hofmann:

Selbstlernende Systeme, deren künstliche Intelligenz die gesammelte Leistungsfähigkeit der Gehirne aller Menschen um ein Vielfaches übersteigen wird, stellen die Welt bald auf den Kopf.

Es ist anzunehmen, dass Computer in den kommenden Jahren die Intelligenz von Menschen erreichen. Mithilfe von künstlicher Intelligenz werden Systeme den Benutzer verstehen können, was in vielen Bereichen, z. B. im Marketing für die Analyse von Kundenbedürfnissen, eine große Bedeutung finden wird.

Rasante Entwicklungen haben große Auswirkungen

Die rasanten Entwicklungen haben heute und zukünftig **große Auswirkungen auf das Rechtssystem**. Neue Technologien verarbeiten immer mehr und unterschiedliche Daten. Elektronische Geräte werden zukünftig u. a. auch menschliche Emotionen und Gedanken erkennen. Rechtlich ungebremste Entwicklungen würde zu einer hohen Transparenz auf allen Ebenen führen: Im Privaten, im Unternehmen, in Staat und Gesellschaft. Es stellen sich die Fragen, wie der Umgang mit Daten geregelt wird und wessen Eigentum die Daten sind. Laut einer Studie der Industrie- und Handelskammer München und Oberbayern geben 50% der befragten an, dass rechtliche Unsicherheit eines der größten Hindernisse der Digitalisierung ist.

Beobachtung neuer Technologien und Innovation ist wichtig

Zusammenfassend ist zu sagen, dass es von großer Bedeutung ist, den technologischen Fortschritt sowie neue Innovationen zu beobachten, da diese große Auswirkungen in vielen Bereichen haben könnten. Alte Technologien können schnell von neuen ersetzt werden, welche in kürzester Zeit Marktreife erlangen können.

3 Denken Sie in Produkten - erst danach in (IT-)Prozessen

Im Bereich der Digitalisierung stehen Führungskräfte vor neuen Herausforderungen: Es müssen neue Ausrichtungen von Unternehmensstrategien vorgenommen und neue Orientierungen vorgegeben werden. Es müssen Entscheidungen darüber getroffen werden, was zu tun ist. Dies ist ein komplexer Prozess, welcher von vielen Faktoren abhängt. Neue Geschäftsmodelle sind notwendig, um auch im internationalen Bereich marktfähig zu sein.

Zu verstehen ist, dass Technologien nicht im Vordergrund stehen: Es ist eher so, dass am Anfang Konzepte stehen, welche Dienstleistungen auf eine bessere Art und Weise erbringen. Dadurch werden Produkte und Dienstleistungen adaptiv und decken einen großen Kundennutzen ab. Es entstehen sogenannte **hybride Leistungsbündel**. Ein gutes Beispiel ist das Internet der Dinge: Produkte werden zu Services bzw. zu hybriden Leistungsbündeln.

Digitalisierung von Produkten und Dienstleistungen

Eine zentrale Fragestellung ist, wie sich traditionelle Produkte und Dienstleistungen Digitalisieren lassen. Im nachfolgenden werden zwei Muster für Geschäftsmodelle im Zeitalter der Digitalisierung vorgestellt:

- **Digitally Charged Product:** Im Internet der Dinge verbinden sich digitale Geschäftsmodellmuster mit solchen aus der nicht digitalen Welt zu einem hybriden Konstrukt. Zu den Kernbestandteilen von Digitally Charged Products gehören:
 - **Physical Freemium:** Zunächst kostenfreier digitaler Service zu einem gekauften Produkt um Interesse für kostenpflichtige Services zu wecken
 - **Digital Add-on:** Kostengünstiges physisches Gut mit zahlreichen kostenpflichtigen digitalen Services
 - **Digital Lock-in:** Nur Original-Komponenten sind kompatibel und keine Fälschungen
 - **Product as Point of Sales:** Physisches Produkt kann direkt den Verkauf eines weiteren Produkts einleiten
 - **Object Self Service:** Ausführung vom autonomen Bestellungen
 - **Remote Usage and Condition Monitoring:** Daten über Zustände und Umgebung übertragen

- **Sensor as a Service:** Sensordaten werden gesammelt, aufbereitet und zur Verfügung gestellt. Im Fokus stehen also die Daten selbst. Ein Beispiel ist die Firma Streetline: Sensoren werden auf Parkplätzen installiert um die Belegungen zu überwachen. Die ermittelten Daten werden anschließend verkauft.

Aspekte der Digitalisierung

Nachfolgend werden einige wichtige Aspekte der Digitalisierung beleuchtet, welche in der heutigen Zeit entscheidende Rollen einnehmen:

- **Digitale Verträge:** Digitale Verträge sind ein wichtiger Bestandteil der Digitalisierung. Digitale Kaufverträge führen beispielsweise selbst Aktionen aus, von der Abbuchung der Raten bis hin zur digitalen Sperrung des Produkts. Einige Vorteile von digitalen Verträgen sind unter anderem: geringere (bis gar keine) Transaktionskosten, beschleunigte Wirtschaft, Intermediäre (Banken, Börsen, Notare, ...) werden überflüssig.
- **Edison-Prinzip:** Ein bewährtes Prinzip für die Digitalisierung ist die Verwendung von vorhandenen Komponenten: Digitale Geschäftsmodelle werden durch das Zusammenfügen von bereits bekannten Technologien und Produkten entwickelt. Es sind somit nicht zwangsläufig neue Technologien o. ä. erforderlich. Oftmals ist die Verknüpfung und Vernetzung von bekannten Technologien erfolgreich.
- **Soziale Medien:** Der Einsatz von sozialen Medien wie Facebook, Instagramm, Snapchat, WhatsApp, YouTube, Twitter, Linkedin und Xing bringt Unternehmen heutzutage viele Vorteile in der externen sowie internen Nutzung: Verbesserung der Zusammenarbeit, bessere Nutzung des Mitarbeiterpotenzials, höhere Motivation und Zufriedenheit, Reduktion des Koordinierungsbedarfs, besseres Wissensmanagement, Verbesserung der Kundenkommunikation und des Unternehmensimage.
- **Customerization:** Aufgrund der hohen Transparenz durch die Digitalisierung ist der Kunde zum Mittelpunkt geworden. Die Konsequenzen für Unternehmen sind Vielfältig. Customerization hat an Bedeutung gewonnen: Ob Mitarbeiter oder Kunde, alle nehmen größeren Einfluss auf die Gestaltung von Informationssystemen, Produkten und Dienstleistungen.
- **Social Listening:** Laut einer Studie von Adobe hat sich der Beruf eines Marketingprofis stark gewandelt: Aufdringliches Marketing bzw. Push-Marketing existiert nicht mehr. Heutzutage geht es vor allem um aktives Zuhören bzw. Social Listening. Es geht darum zu wissen, worüber die Kunden reden, wo es Probleme gibt und wie das Unternehmen

diese lösen können. Unternehmen müssen folgendes tun: Großartige Kundenerlebnisse schaffen, auf viele Kanälen Kontakt zu Kunden herstellen, Kunden das Gefühl geben verstanden zu werden sowie Kundenverhalten und -bedürfnisse vorhersehen.

Digitalisierung ist die Zukunft

Die Akzeptanz digitaler Produkte ist hoch: Aus der Historie von Industrie 4.0 ist ein deutlicher Trend hin zu Produkten zu erkennen, welche auf Kundenwunsch hin gebaut wurden. Laut dem Fraunhofer IAO wollen Kunden von produzierende Firmen vermehrt Losgrößen mit einer großen Variantenvielfalt und kurzer Lieferzeit. Eine **effiziente Produktionsplanung, produktionsnahe IT- sowie geeignete ERP-Systeme, ein klares Lean Management und optimale Montageprozesse sind erforderlich**. Ein Beispiel hierfür ist das Konzept Storefactory von Adidas: Der Konzern will Produkte nach Kundenwunsch direkt in kleinen Fabriken in unmittelbarer Nähe von Geschäften fertigen.

Entsprechend werden Geschäftsprozesse auf Softwarestrukturen so angepasst, dass Services interoperabel automatisch miteinander vernetzt werden. Eine Verbindung von Verkaufsorten zu Produktionsstätten ermöglicht die Fertigung von individuellen Gütern. Unternehmenssoftware werden soweit adaptiert, dass diese in Produktionsketten integriert werden.

Unternehmen wie z. B. Uber, Spotify und car2go haben zudem neue IT-Strategien entwickelt, um gleichzeitig das Nutzungsverhalten von Kunden zu studieren damit mit diesem Wissen neue Produkte entwickelt werden können, während bestimmte Services angeboten werden.

Im Kern digitaler Geschäftsmodelle geht es um Daten, um den Erkenntnisgewinn mit dem Ziel der Transparenz über die eigenen Prozesse und Produkte, über das Nutzungs- und Kundenverhalten sowie über Kundenwünsche und -reklamationen.

Flexibilität in der technologischen Ausstattung ist entscheidend. Es ist wichtig, dass diese homogen und skalierbar sind. Die Kosten sollten im Hinblick auf den Betrieb proportional zur Last sein. Cloud-Lösungen spielen hierbei eine große Rolle. Geschwindigkeit ist entscheidend: Neue Technologien und Innovationen können den Markt schnell ändern. Profitable Geschäftsmodelle können sich innerhalb von kürzester Zeit ändern.

4 Aufrüsten für die digitale Zukunft

Die Digitalisierung der Welt, und hierbei insbesondere eines Unternehmens, sollte nicht nur gleichgesetzt werden mit der Nutzung von Technologien an offensichtlichen Stellen, wie zum Beispiel in Produktionsanlagen. Viele Möglichkeiten der Digitalisierung für Unternehmen liegen dabei auch in produktionsfernen Bereichen wie dem Vertrieb, der Preissetzung, der Planung, dem Controlling und auch dem Einkauf. Aus diesem Grund muss die Digitalisierung des Ganzen Unternehmens vorangetrieben werden und während dieses Prozesses müssen möglichst viele Angestellte und Entscheider dem Prozess gegenüber positiv gestimmt sein.

Dabei kann die Digitalisierung dieser produktionsfernen Bereiche in vier Stufen unterteilt werden:

- **Stufe A** – Information: Dazu gehören das Bereitstellen allgemeiner Unternehmensinformationen, Produkt- und Dienstleistungskataloge, Kontaktinformationen oder das Unterhalten von Stellenbörsen.
- **Stufe B** – Kommunikation: Hierzu zählen Dienste wie Suchfunktionen, Formulare, FAQ, E-Mails, Newsletter, Chats, Diskussionsforen, Corporate Blogs und soziale Netzwerke, die die Interaktivität mit den Kunden über das Web fördern.
- **Stufe C** – Transaktion: Bei dieser Stufe geht es um die elektronische Geschäftsanbahnung und -abwicklung, mit Online-Offerte-Erstellung, Bestellwesen, Bezahlung und Distribution.
- **Stufe D** – Integration: Die höchste Stufe betrifft die Integration und Kundenbindung, z. B. durch personalisierte Websites, One-to-One-Marketing, Online-Order-Tracking sowie den Einsatz digitaler Agenten für Beratung und Verkauf individueller Produkte und Dienste.

Dabei gilt, dass umso höher die Stufe (**Stufe D**), desto mehr Wert wird für das Unternehmen generiert. Der Aufwand für das Erreichen der Stufe wächst allerdings proportional mit. Dies hängt auch mit der Kultur im Unternehmen zusammen und ob Mitarbeiter diese Form der Digitalisierung annehmen und fördern anstatt zu denken, dass das die Arbeit der IT-Abteilung sei.

Außerdem kann dieser Wandel in der Unternehmenskultur auch dazu führen, dass neue Absatzmöglichkeiten entdeckt werden. Sogenannte “As-A-Service”-Geschäftsmodelle können dabei eine Rolle spielen, da hier nicht klassisch ein Produkt verkauft wird, sondern die Nutzung von Verbrauchseinheiten.

Um diesen Wandel in der Unternehmenskultur herbeizuführen, ist es notwendig einige Punkte zu beachten bzw. den Wandel aktiv zu steuern. Ein “Management of Change” ist also notwendig. Dieser Prozess muss auf der organisatorischen, persönlichen und auch auf der technologischen Ebene greifen. Dabei können agile Methoden und Arbeitsweisen motivationsfördernd wirken, da durch die vielen inkrementellen Schritte leichter ein Fortschritt zu erkennen ist.

Um diesen Wandel erfolgreich herbeiführen zu können und das Risiko des Scheiterns zu minimieren, müssen folgende Punkte erfüllt sein und bei den Mitarbeitern ankommen:

- Vision
- Fähigkeiten
- Anreize
- Ressourcen
- Aktionsplan

Eine fehlende Vision, kann zu Verwirrungen bei den Mitarbeitern führen. Wenn die für diesen Wandel notwendigen Fähigkeiten im Team fehlen, dann führt das wiederum zur Besorgnis. Ohne vernünftige Anreize für die Mitarbeiter, besteht die Gefahr, dass diese Widerstand zeigen und das Projekt nicht unterstützen. Wenn das Unternehmen nicht die notwendigen Ressourcen zur Verfügung stellen kann, steigt die Frustration bei den Mitarbeitern, da alle anderen Punkte erfüllt wären und nur die Ressourcen fehlen, um im Gegenzug einen Mehrwert für das Unternehmen zu generieren. Ein fehlender Aktionsplan wird einen falschen Start des Wandels nach sich ziehen und so die geleistete Arbeit und Motivation der Mitarbeiter gefährden.

Der digitale Arbeitsraum kann dabei helfen, dass die Mitarbeiter gemeinsam Wissen erarbeiten und miteinander teilen und ist aus diesem Grund erstrebenswert. So können Unternehmensweit alle wichtigen Erkenntnisse gesammelt und ausgewertet werden.

Der Manufacturing Service Bus (MSB) ist der digitale Arbeitsraum für die **produktionsnahen** Bereiche des Unternehmens und vernetzt in einer “smarten” Fabrik die verteilten Maschinen. Außerdem kann der MSB bestimmte Ereignisse weitermelden und ermöglicht im idealfall eine dynamische Regulierung der Produktion.

Durch die Digitalisierung nach außen kann man den Kunden eine bessere Möglichkeit zur Interaktion mit den eigenen Produkten und Dienstleistungen ermöglichen. Als Beispiel sei hier das Bankwesen genannt. Die Prozesse im Bankwesen wurden über die Jahre so vereinfacht und digitalisiert, dass viele Kunden die Geschäfte selbst erledigen.

Überweisungsträger werden von den Kunden ausgefüllt bzw. sogar nur noch abfotografiert. Kreditanträge werden vom Kunden ausgefüllt und zunächst vollautomatisch von Algorithmen überprüft, bevor ein Bankmitarbeiter den Antrag zu Gesicht bekommt.

5 Bringen Sie Ihre Technik auf Vordermann

Dieses Kapitel zeigt anhand einiger Beispiele die “Must Haves” der Industrie 4.0 und gibt damit Empfehlungen ab wie der Wandel zu einem Digitalisierten Unternehmen aussehen kann. Das Ziel ist das Zusammenwachsen von produktionsfernen und produktionsnahen Unternehmensbereichen und der Produktionstechnik durch die IT.

Zu Beginn des Kapitels liegt die Betonung auf “Safety- und Security-Infrastruktur” und, dass die Definition und Einhaltung gewisser Standards notwendig ist, um durch die Digitalisierung des Unternehmens nicht neue Angriffsvektoren zu ermöglichen. Stuxnet und Flame werden dabei als Beispiele für Angriffe gegen industrielle Systeme angeführt.

Dabei spielen die folgenden Punkte eine wichtige Rolle für die Sicherheitsstrategie im Produktionsumfeld:

- Absicherung der Kommunikation
- Absicherung der Endgeräte und der Sensoren/Aktoren selbst
- Absicherung sämtlicher Schnittstellen
- Autorisierung der Zugriffe

Aus diesen Punkten ergeben sich folgende Anforderungen für die Digitalisierung des Unternehmens:

- Flexible, elastische Plattformen orchestrieren
- Übergreifende standardisierte Kommunikation
- Hohe Sicherheitsstandards in Hinblick auf Safety und Security
- Data Analytics zur Bewertung von Prozessen und Systemen
- Integration in die globale IT-Landschaft
- Orchestrierung von Systemen statt Programmierung
- Herstellerübergreifendes Engineering
- Verwaltung des gesamten Prozess- und Produktlebenszyklus
- Zusammenarbeit über Teams, Ressorts, Standorte, Organisationen hinweg ermöglichen

Dabei muss jedoch jede Ebene abgesichert werden: Defense-in-Depth Sicherheit.

Zusätzlich zu dem Defense-in-Depth kann die Sicherheit durch einen Null-Trust-Konzept gesteigert werden. Dazu wird wiederum die Mikrosegmentierung auf der Software Ebene genutzt. Das heißt, die Prozesse können nicht alle miteinander kommunizieren, sondern nur mit den notwendigen Prozessen. Eine Punkt-zu-Punkt Kommunikation muss also ermöglicht werden. Auf dieser minimalen Kommunikationsebene greift dann das Null-Trust-Konzept und erlaubt ebenfalls nur minimalen Zugriff von außen (z.B. den Mitarbeitern), um so möglichst wenige Angriffsvektoren auf dem Ganzen System zu haben.

Das ERP-System, was in den meisten Unternehmen vorhanden ist, wird inkrementell zur Schaltzentrale des digitalisierten Unternehmens ausgebaut. Dadurch ergibt sich langfristig ein adaptives ERP-System, das durch Simulationen und Vorhersagen die Parameter des Unternehmens anpassen kann.

Auch in diesem Kapitel wird wieder auf die Mitarbeiter und deren Willen zur Digitalisierung eingegangen. Dies positive Einstellung der Mitarbeiter gegenüber der Digitalisierung des Unternehmens ist offensichtlich eine Notwendigkeit. Dabei gibt es verschiedene Reifetypen, die sowohl auf Unternehmen als Ganzes, als auch auf einzelne Mitarbeiter bezogen werden können:

- **Digitale Konservative:** zögern, sich umzustellen, und lassen somit Chancen verstreichen
- **Digitale Anfänger:** niedrige digitale Intensität, sowohl bei der Nutzung neuer Technologien als auch beim Führungsstil
- **Digitale Fans:** einige digitale Initiativen, aber keine Maximierung des Geschäftsnutzens
- **Digitale Experten:** haben eine digitale Kultur und Investitionen geschaffen und profitieren dadurch von Wettbewerbsvorteilen – die digitale Elite sozusagen

Erfolgsfaktoren für eine digitale Zukunft

Zusammenfassung des Buches:

Titel: Erfolgsfaktoren für eine digitale Zukunft - IT-Management in Zeiten der Digitalsierung und Industrie 4.0

Verfasser: Egmont Foth

Verlag: Springer Vieweg

Jahr: 2017

ISBN: 978-3-662-53176-1, 978-3-662-53177-8 (eBook)

Zusammenfassung von: Malte Berg, Jonas Wiese, Niklas Harting

1 Einleitung

Durch die Digitalisierung wird die Verarbeitung und Vernetzung von riesigen Datenmengen zu einem immer größer werdenden Markt. Aktuelle Technologien unterstützen diese Prozesse und führen Unternehmen zum Erfolg.

2 Was verstehen wir unter Informationstechnologie?

Informationstechnologische Systeme haben eine enorme gesellschaftliche und technische Veränderung ausgelöst. Diese bilden durch das Sammeln und Verarbeiten von Informationen eine Erleichterung vieler Prozesse, was es stark in der Gesellschaft und Wirtschaft expandieren ließ.

2.1 Der Ursprung

Erste Entwicklungen begannen mit dem Telefon, welches viele für unzureichend hielten und niemals erwartet hätten, dass es einmal weltweit für vernetzte Kommunikation sorgen wird. Ebenso erging es den ersten Computern. Die großen Rechnerräume, welche benötigt wurden um die ersten Computer in Betrieb zu nehmen, gaben vielen das Gefühl, dass es nur einen äußerst kleinen Markt für solche Geräte gäbe.

2.2 Die Gegenwart

In der Wirtschaft werden informationstechnologische Systeme beispielsweise zur Unterstützung von Geschäftsprozessen verwendet um eine reibungslose und optimale Verwaltung von Ressourcen über vernetzte Computer zu gewährleisten. Diese Systeme, auch Enterprise-Resource-Planning-Software genannt, geraten immer häufiger in einen Konflikt mit der stetig wachsenden Komplexität der Anforderungen. Hieraus resultieren Ausfälle der Systeme, schlechte Laufzeiten bei Suchanfragen, unbefriedigende Ergebnisse für den Kunden und erhöhte Supportanfragen von Kunden. Daher ist es erforderlich Korrekturmaßnahmen durchzuführen um die Komplexität dieser Systeme zu verringern, beispielsweise mithilfe von Standardisierungen oder ausgereiften Architekturmodellen.

2.3 Die Zukunft

In Zukunft werden diese vernetzten Systeme im Alltag viel präsenter sein. So können in einer Vielzahl an Alltagsgegenständen Sensoren für einen komfortableren Lebensstil, sowie für optimierte Geschäftsprozesse sorgen. Die Vielzahl unterschiedlicher Sensoren sorgt für riesige Mengen an Daten, was eine effizientere Verarbeitung von Daten voraussetzt um nicht an alten Daten hängen zu bleiben, wenn sich die Bedingungen bereits geändert haben

und bestimmte Reaktionen erwartet werden. Durch eine Vernetzung von Kunden und Geschäftspartnern (wie beispielsweise Lieferanten) können mithilfe der Analyse von Datenmengen sowie der Vernetzung verschiedenste Geschäftsbereiche, effizienter und flexibler auf Anforderungen bzw. Änderungen von Auftragsgebern reagiert werden. Diese Vernetzungen, Nutzungen von Services und die daher gehende Flexibilität, wird durch das Cloud Computing voran gebracht. Es vereinfacht und spart Kosten bei der Bereitstellung solcher Dienste. Dabei kann die Infrastruktur frei gewählt werden und entschieden werden, ob die Dienste in der eigenen Umgebung oder in der verteilten (privaten oder öffentlichen) Cloud angeboten werden soll.

3 Welchen Nutzen hat Informationstechnologie

Die Informationstechnologie ermöglicht einerseits die Entstehung neuer Marktsegmente, andererseits aber auch die Optimierung bestehender.

3.1 Wirtschaftlichkeit

Bei der Optimierung von Geschäftsprozessen wird erst überprüft, ob eine Optimierung überhaupt wirtschaftlich ist. Hierfür werden verschiedenste Analysen bezüglich der Kosten und des Nutzens durchgeführt. Es wird dabei zwischen zwei Fällen unterschieden, bei welchen eine Wirtschaftlichkeit vorliegt. Einerseits ein identisches Ergebnis zum vorherigen Verfahren bei verringertem Aufwand, andererseits ein verbessertes Ergebnis (beispielsweise die Qualität des Produktes) bei nahezu gleichem Aufwand. Bei der Analyse ist es noch wichtig zu betrachten, dass die zu tätige Investition in Zukunft abgedeckt wird und höhere Einnahmen erzielt werden.

3.2 Wettbewerbsfähigkeit

Ein weiterer Nutzen der Informationstechnologie ist die Wettbewerbsfähigkeit. Ein Verzicht auf informationstechnologische Systeme würde ein Unternehmen so starke Defizite in Punkto Planung, Zuverlässigkeit, Effizienz und einigen weiteren Punkten einbringen, sodass es nicht mehr konkurrenzfähig wäre. Daher werden solche Systeme unerlässlich für Unternehmen und werden immer weiter entwickelt um bessere Verfahren gegenüber der Konkurrenz zu entwickeln.

4 Einsatz von Informationstechnologie in schlanken Unternehmen

4.1 Woran können wir ein schlankes Unternehmen erkennen?

In schlanken Unternehmen wird die kontinuierliche Verbesserung von Geschäftsprozessen und Kundenzufriedenheit angestrebt um einen möglichst hohen Gewinn bei einer möglichst geringen Verschwendungen zu erzielen.

4.2 Wie setzen schlanke Unternehmen Informationstechnologie ein?

Im Fokus eines Unternehmens sollten zu allererst die Prozesse liegen. Sind die Geschäftsprozesse noch nicht ausgereift und könnten noch optimiert werden, sollte noch nicht mit der Arbeit eines Softwaresystems begonnen werden. Erst wenn ein Geschäftsprozess ausgereift und stabil im Unternehmen durchgeführt wird, ist es sinnvoll ein Softwaresystem zur Verbesserung der Durchführung zu entwickeln. Ein weiterer wichtiger Punkt für den Einsatz, ist die Flexibilität. Hierbei sollte darauf geachtet werden, dass Informationen immer zum richtigen Zeitpunkt versendet werden. Werden zu früh Informationen versandt, kann es unnötige Prozesse hervorrufen, die aufgrund von Änderungen nicht nötig waren.

4.3 Wie arbeiten schlanke IT-Organisationen?

Schlanke Organisationen verwenden Echtzeit-Prinzipien für ihre Services. Dabei wird schnellstmöglich auf Fehler reagiert, unter ständiger Verbesserung Services von hoher Qualität angeboten und die Antwort- und Verarbeitungszeiten möglichst gering gehalten. Ebenfalls wird darauf geachtet Verschwendungen zu finden wie beispielsweise nicht benötigte Geschäftsprozesse und diese zu eliminieren.

Fließ-Prinzip (One piece flow)	Synchronisations-Prinzip (Customer's cycle)	Nachfrage-Prinzip (Pull principle)	Null-Fehler-Prinzip (Zero faults principle)
Ein direkter Material- und Informationsfluss ermöglicht kurze und stabile Durchlaufzeiten.	Am Kundentakt ausgerichtete und aufeinander abgestimmte Prozesse erzielen Effizienz.	Überproduktion lässt sich vermeiden, wenn nur das produziert wird, was tatsächlich gebraucht wird.	Stabile und fehlerfreie Prozesse erhöhen die Planbarkeit und reduzieren die Verschwendungen.

Anhand der Abbildung können Verfahren zu kontinuierlichen Entwicklung, besseren Wertschöpfung und erhöhter Kundenzufriedenheit eingesehen werden.

4.4 Wie erfolgen Problemlösungen in schlanken Organisationen?

Zum problemlösen wurde ein Problemlösungsprozess entwickelt, welche bei dessen Durchführung helfen soll, Probleme zu diagnostizieren und zu umgehen. Dazu wird erst geplant, was überhaupt das Problem ist und in wie fern es der optimalen Lösung abweicht. Ebenfalls wird überprüft, was eine machbare und geeignete Gegenmaßnahme sein könnte. Im nächsten Schritt wird diese Gegenmaßnahme ausgeführt und die darauf folgenden Ergebnisse überprüft. War die Gegenmaßnahme erfolgreich wird die Korrektur standardisiert. War es nicht erfolgreich muss der komplette Prozess neu durchlaufen werden.

4.5 Wie lassen sich schlanke Managementmethoden in Entwicklungsprojekten nutzen?

Es gibt verschiedene Managementmethoden, welche unterschiedliche Ziele verfolgen. Ist es vorgesehen, dass Fehler schnell erkannt werden sollen und eine hohe Präsenz des Kunden in den Planungsschritten vorhanden sein soll, bietet sich Scrum an. Hierbei werden Anforderungen ausgearbeitet, welche in Sprints (2-4 Wochen) ausgearbeitet werden. Die entstandenen Prototypen werden mit dem Kunden analysiert, welcher Verbesserungsvorschläge und Anmerkungen äußern kann. Hierdurch werden außerdem frühzeitig Fehler erkannt. Als Alternative gibt es die klassische Projektmanagementmethode, welche eine bessere Abhilfe in Punkt Zeit- und Budgetmanagement bietet, jedoch eine geringere Kundennähe und spätere Fehlererkennung beinhaltet. Eine weitere Möglichkeit ist es die Komplexität in einfache kleine Arbeitspakete einzuteilen und regelmäßig Meetings mit

dem Kunden zu vereinbaren um die Zufriedenheit mit dem Produkt zu gewährleisten. Die Arbeitspakete sollten hierbei eine Dauer von einem Tag haben und werden in täglichen Meetings besprochen, analysiert und Probleme behoben.

4.6 Welche Erfolgsfaktoren gibt es für schlanke IT-Organisationen?

Um Unternehmen langfristig ihren Erfolg zu gewährleisten, sollten diese nicht auf bestehenden Verfahren hängen bleiben, sondern eine stetige Weiterentwicklung und Standardisierung dieser Fortschritte durchführen.

4.7 Wie lässt sich der Reifegrad schlanker Organisationen ermitteln?

Der Reifegrad misst sich an verschiedensten Eigenschaften des Unternehmens. Die wichtigsten sind die fortlaufenden Entwicklungen. Das bedeutet, dass Unternehmen immer wieder Problemanalysen durchführen, Prozesse dabei optimieren und überflüssige Prozesse minimieren bzw. eliminieren sollten. Ebenfalls spielt hierbei die Kundennähe stark mit ein und die Präsenz der Führungskräfte.

5 Was benötigt eine erfolgreiche IT-Organisation?

5.1 Organisationsstruktur

Eine IT-Organisation besteht aus dem Governance-Bereich, in dem Verwaltungs- und verschiedenste Managementtätigkeiten (z.B. Standardisierungs- und Qualitätsmanagement) ausgeführt werden, der Anwendungsentwicklung und –betreuung, welche sich sowohl mit der Entwicklung als auch mit dem Support und Projektmanagement befasst, sowie dem IT-Betrieb, welcher verschiedene administrative Dienste leistet. Ein weiterer entscheidender Punkt in der Organisationsstruktur ist die Entscheidungsfreiheit und Verteilung von Verantwortlichkeiten. Werden den Arbeitern die Aufgaben vorgesetzt ohne dass diese Verbesserungen oder ähnliches äußern können, führt dies zu einer Demotivation. Die Kreativität wird somit unterdrückt und der Arbeiter sieht sich selbst nicht als Verantwortlichen für die Aufgabe. Das löst ebenfalls eine stärkere Fehleranfälligkeit aus, da der Arbeiter nicht mehr an Verbesserungsmöglichkeiten denkt. Ein freies Arbeitsumfeld für den Arbeiter weckt die Begierde Erfolg zu haben und verbessert die Leistungen. Jedoch sollten auch nicht zu früh zu viele Freiheiten gegeben werden. Gerade in dem anfänglichen Reifungsprozess sollte eine ausreichende Kontrolle vorhanden sein. Diese sollte sich jedoch mit der Zeit immer mehr verringern und den Teams die Entscheidungsfreiheit lassen.

5.2 IT-Mitarbeiter

Die Funktion eines IT-Mitarbeiters wird über die Stellenbeschreibung festgelegt. Hierbei wird nicht nur Wert auf Fachwissen gelegt. Viel entscheidender sind Soft Skills, wie der Umgang im Team und mit Kunden, sowie das Herstellen von fachübergreifenden Verknüpfungen. Ebenfalls werden IT-Mitarbeiter häufig in den Kontakt mit neuen Technologien kommen. Daher benötigen diese fundamentale Kenntnisse von Technologien und die Fähigkeit sich in neue Sachverhalte einzuarbeiten. Da Technologien stetig weiterentwickelt werden, ist es ebenfalls erforderlich, dass IT-Mitarbeiter permanent lernen und sich weiterentwickeln. Um diese Fähigkeiten in einem IT-Mitarbeiter zu erkennen gibt es Assessment Center. Hierbei werden beispielsweise Stresssituationen getestet. Assessments werden nicht nur bei der Bewerbung genutzt, um die Qualifikation von Bewerbern zu testen. Es kann auch Verwendet werden um Mitarbeitern intern die richtige Rolle zuzuordnen. Um eine individuelle Bewertung aufzustellen, wird der Mitarbeiter nach bestimmten Kriterien analysiert. Angefangen wird mit einem Anforderungskatalog, wo niedergeschrieben wird, welche

Kompetenzen und Anforderungen überhaupt gesucht werden. Hierbei können beispielsweise bestimmte Führungsqualifikationen niedergeschrieben werden. Nach dem Assessment werden die Kompetenzen im Kompetenzmodell individuell bewertet. Danach kann das Ergebnis in einem Potential- und Leistungsdiagramm eingestuft und verglichen werden. Hieraus können nun neue Entschlüsse gezogen werden, wie beispielsweise Fortbildungsmaßnahmen.

5.3 IT-Buisness-Alignment

Die korporative Zusammenarbeit zwischen dem Geschäftsbereich und der IT-Organisation, welche sich gegenseitig gleichberechtigt behandeln prägt den Begriff IT-Buisness-Alignment. Arbeiten beide Parteien zusammen, wird die Kreativität beider Seiten genutzt, eine höhere Effizienz erzielt, Fehler vermieden und eine gesunde Arbeitsatmosphäre geschaffen.

5.4 IT-Strategie

Eine IT-Strategie ist ein mittel- bis langfristiger Plan für die Entwicklung der IT in einem Unternehmen. Die IT-Strategie sollte mindestens einen Zeitraum von 2 Jahren umfassen. Für die Entwicklung einer IT-Strategie müssen die Bereiche der geschäftlichen Anforderungen an die IT, sowie der aktuelle Zustand der Unternehmens IT analysiert werden.

Eine fertige IT-Strategie sollte folgende Punkte enthalten:

- Grundsätze
- IT-Architektur mit IT-Applikationen, IT-Infrastruktur, Cloud-Computing und Mobilität
- IT-Services
- IT-Sourcing
- IT-Organisation
- IT-Projekt-Portfolio

5.5 IT-Governance

Die IT-Governance umfasst die Grundsätze, Verantwortlichkeiten und Prozesse zur Steuerung des Umgangs mit der IT im Unternehmen, inkl. des dazugehörigen Risiko- und Compliance-Managements.

Die Aufgaben des IT-Governance werden vom CIO, seinem IT-Führungsteam, den IT-Boards und dem CIO-Office mit Partnermanager, Sicherheitsmanager, IT-Controller etc. wahrgenommen. Zu den Aufgaben gehören u.a. die Entwicklung der IT-Strategie sowie die Steuerung und Überwachung ihrer Umsetzung, aber auch das IT-Risiko-, IT-Compliance- und IT-Sicherheitsmanagement.

5.6 IT-Architektur

Die IT-Architektur bildet eine Grundstruktur für die IT-Infrastruktur. Sie ist auf "das Design, die Auswahl, die Entwicklung, die Implementierung, die Wartung und das Management der IT-Infrastruktur" anzuwenden. Mit ihr erfolgt eine "einheitliche strategische Ausrichtung" der IT. Die gewählte Architektur muss in der Lage sein, zukünftige Anforderungen zu erfüllen. Währenddessen muss aber eine hohe Verfügbarkeit und Leistungsfähigkeit der Systeme gewährleistet werden. Ähnlich wie bei der Planung einer Stadt gibt es bei der Entwicklung einer IT-Architektur einen "Bebauungsplan". Dieser dokumentiert den Ist-Zustand, zeigt aber auch die notwendigen Schritte hin zur neuen Soll-Architektur auf. Bei der Entwicklung der Soll-Architektur ist es wichtig, dass sie zum allgemeinen Geschäftsprozess des Unternehmens passt. Auch müssen die Anforderungen an die neue Architektur definiert werden. Zudem dürfen aktuelle Trends nicht ignoriert werden. Die zukünftige IT Landschaft soll: zentral, einfach, integriert und flexible sein, ohne unnötige Risiken zu beinhalten. So sollte innerhalb einer Unternehmensgruppe eine einheitliche Software-Plattform eingesetzt werden. Aber auch die genutzte Hardware sollte einer Standardkonfiguration entsprechen. Einige Aktuelle Trends sind: Hybrid Infrastruktur: Unternehmenskritische Dienste lokal hosten, andere Dienste auslagern SaaS: Software kann kostengünstig aus der Cloud bezogen werden. Industrie 4.0: Hoch vernetzte Fertigungsprozesse Big Data: Einsatz neuer Analyse Methoden für große Datenmengen Arbeitsplatz der Zukunft: Hohe Vernetzung ermöglicht trotz großer Mobilität einfache Kommunikation Design Thinking: Multidisziplinär arbeitende Teams entwickeln Produkte die sich nah am Kunden befinden

5.7 IT Standards:

Informationstechnologien sind Produktionsmittel und somit ist ihre Qualität und Verfügbarkeit für ein Unternehmen wichtig. Hierzu müssen Standards definiert und eingehalten werden. Sowohl das BSI als auch die ITIL definieren für ihren Bereich „best practises“.

Verschiedene Standards:

Corporate Security Policy: Jeder Mitarbeiter muss alle Regeln verantwortungsvoll einhalten (z.B. Passwörter und Virenschutz) IT-Architektur: Standardisiert die IT Architektur und führt damit auch zu einer strategischen Ausrichtung der IT IT-Prozesse: Definiert für ein Unternehmen Soll Abläufe IT-Sicherheit: Risikoanalyse mit CIA Aspekte. Um den Ablauf zu vereinfachen wurde ein IT-Sicherheits-Basis-Check erstellt. Dieser gliedert sich in verschiedene Kategorien und je einer gestellten Frage. Ausfalltoleranz: Wie lange kann das Unternehmen ohne IT auskommen? Sicherheitsrisiken: Finanzielles Risiko bei IT Ausfall bekannt? Notfallablauf: Wer wird informiert? Vorkehrungen: Virenschutz, Firewall, Sicherheits-Patches aktuell?

5.8 IT Budget

Das IT Budget bildet eine Entscheidungsgrundlage für Unternehmen. Die Angaben des „IT Demand Managers“ fließen in die IT Budgetplanung ein. Sie wird vom IT Controller in Zusammenarbeit mit den IT Führungskräften erstellt. Auftretende Änderungen sind aufgrund des Vorjahresberichts gut festzustellen. Das Budget ermöglicht zudem Aussagen zur Realisierbarkeit von IT Projekten zu treffen. So darf der geschätzte Projekt Aufwand nicht die verfügbaren Mittel überschreiten.

Um einen transparenten Soll-Ist-Vergleich des Budgets zu schaffen wird vom Controlling ein monatliches „Kosten reporting“ erstellt. Neben dieser Aufgabe hat das Controlling noch weitere Beratende und Unterstützende Aufgaben. Auch muss es verschiedene Be- und Verrechnungen erstellen.

5.9 IT-Partner-Management

Das IT-Partner-Management überwacht, steuert und optimiert Ausgelagerte (outgesourcete) Dienstleistungen. Um zu erkennen, ob eine Partnerschaft nicht mehr profitable, ist ein regelmäßiger Informationsaustausch zwischen den Partnern nötig.

Das Outsourcing ermöglicht es dem Unternehmen sich auf die Kernaufgaben zu konzentrieren. Gründe für das Outsourcing können kosten Reduktion, Verteilung von Kompetenzen oder die fehlende technische Innovation des eignen Unternehmens sein. Um einen Outsourcing Dienstleister zu wählen sind verschiedene Kriterien anzuwenden. Sie gliedern sich von Preis über Referenzen hin zur finanziellen Stabilität des Dienstleisters.

5.10 Business-Process-Management

„Business-Process-Management (BPM) verbessert die Unternehmensleistung durch ein konsequentes Geschäftsprozessmanagement und die fortlaufende Optimierung der Geschäftsprozesse. Dies umfasst die Analyse, das Design, die Modellierung, die Implementierung, die Überwachung und die Optimierung von Geschäftsprozessen. Optimale Prozesse orientieren sich am Kundenbedarf und haben einen hohen Wertschöpfungsanteil. Sie sind effektiv und effizient.“ Da alle Aktivitäten optimal aufeinander abgestimmt sein müssen, wächst die Bedeutung von BPM weiter. Die definierten Prozesse können aber nicht dauerhaft perfekt sein, deshalb müssen sie ständig angepasst werden. Dieses resultiert aus dem sich ständig wandelnden des Marktes, der Technologien und des Wettbewerbs. Die Aufgaben eines Prozessberaters im BPM Team umfassen: Die Analyse und den Entwurf von Dokumenten, die Erfassung der Anforderungen an die Prozesse sowie Beratung und Moderation von Prozess Design Workshops.

5.11 Service-Level-Management

Das Service-Level-Management stimmt mit den internen Kunden ab, welche Dienstleistungen in welcher Qualität (Service-Level-Agreement) zu erbringen sind. Die Service-Level-Agreements definieren die Qualität und Verfügbarkeit der von der IT-Organisation angebotenen IT-Dienstleistungen. Außerdem überwacht es die Einhaltung der Vereinbarungen und erstellt Reports. Zu den häufig benötigten Services gehören IT-Service-Desk, Telekommunikation und Computerarbeitsplatz (File und Print Service, Mail, LAN).

5.12 IT-Demand-Management

„Das IT-Demand-Management ist die zentrale Stelle für die strukturierte Aufnahme, Bündelung, Bewertung und Umsetzungskoordinierung von IT-Anforderungen“. Es dient dazu den "IT-Unterstützungsbedarf der internen Kunden zu identifizieren". Zudem sorgt es für den zielgerichteten Einsatz von IT Ressourcen. Hierbei werden wichtige Aufgaben höher Priorisiert als andere. Insgesamt ist das Demand Management eine elementare Komponente erfolgreicher IT Organisationen.

5.13 IT-Leistungsverrechnung

IT Organisationen erbringen die vereinbarten Leistungen als Shared-Service-Center. Alle möglichen Leistungen werden in einem IT Leistungskatalog zusammengefasst. Er wird veröffentlicht. In diesem Katalog wird auf die Leistungsart, die SLA Zuordnung, die Verrechnungseinheit und die Preise eingegangen. Mit Hilfe dieses Katalogs erkennen die Kunden der Wert der erhaltenen Leistung einfacher und die Kostentransparenz steigt. Für

die Anbieter hat diese Methode den Vorteil, dass die Kunden nicht mehr benötigte Leistungen früher melden, da sie direkt Kosten einsparen können. Der Anbieter kann so knappe Ressourcen besser verwalten. Die zugrundeliegenden Preise werden jährlich festgelegt und bleiben über das Jahr konstant.

5.14 IT-Risikomanagement

Das IT-Risikomanagement dient zur Sicherstellung der Kontinuität des Geschäftsbetriebs. Es beinhaltet die „Erfassung, Bewertung, Behandlung und Überwachung von Risiken“. Das BSI unterscheidet zwischen 6 Kategorien („Elementare Gefährdung, höhere Gewalt, organisatorische Mängel, menschliche Fehlhandlung, technisches Versagen, vorsätzliche Handlung“). Mögliche resultierende Risiken sind der Verlust von Daten, der Ausfall von Systemen oder Verstöße gegen rechtliche Vorschriften. Um die Risiken und dessen Folgen einschätzen zu können wird eine Risikoanalyse durchgeführt. Hier werden Risiken mit ihren erwarteten Eintrittswahrscheinlichkeiten und dessen finanziellen Folgen aufgeführt. Außerdem wird für jedes unternehmenskritische System eine Gefährdungsanalyse durchgeführt. Bei dieser wird erfasst, welche Vorkehrungen bereits getroffen wurden und welche noch getroffen werden müssen (Redundanz, Wartungsvertrag). Hieraus ergibt sich ein Realisierungsplan. In diesem sind Verantwortliche, sowie Kosten und Termine aufgeführt.

5.15 IT-Asset- und Lizenzmanagement

Um nicht gegen die vertraglich festgelegten Nutzungsbedingungen zu verstößen, werden immer aktuelle Informationen über die eingesetzten Produkte benötigt. Andernfalls drohen Strafen. Ein IT-Asset ist jedes IT-System und Software-Lizenz die dem Unternehmen gehört. Die Informationen über diese kommen aus verschiedenen Quellen (Buchhaltung System bis Active Directory) und können in einem IT-Asset-Managementsystem verwaltet werden. Der Einsatz eines solchen bringt verschiedene Vorteile mit sich: Lizenzverträge können optimiert werden, Lizenzrechtsverletzungen sind vermeidbar, Vermeidung von nicht genutzten IT Assets.

5.16 BI-Management (Business Intelligence)

BI-Management beschreibt die systematische Erfassung, Auswertung und Darstellung von geschäftlichen Daten. Diese Daten dienen den Führungskräften bessere Entscheidungen zu treffen. Um ein solches zentrales Berichtssystem aufzubauen wird ein Business-Intelligence-

Manager benötigt. Er arbeitet mit dem Controlling und der IT Organisation zusammen und Standardisiert unter anderem Kennzahlen und Berichte.

5.17 Master Data Management

Um die Stammdaten eines Unternehmens (Produkte, Lieferanten, Kunden, Mitarbeiter) zu verwalten arbeitet der Stammdatenmanager übergreifend über eine Unternehmensgruppe.

5.18 KPIs

Kennzahlen oder auch „Key Performance Indicators“ (KPIs) machen es möglich komplexe Sachverhalte besser zu verfolgen zu können. Außerdem dienen sie dazu, den Erfüllungsgrad wichtiger Ziele zu messen. Daher sind sie unverzichtbar um wichtige Entscheidungen zu treffen. Mit Hilfe dieser Zahlen ist es einfacher Erfolg oder Misserfolg einer Entscheidung festzustellen. So können die gemessenen Zahlen vor der Entscheidung mit denen, die nach der Entscheidung erstellt worden verglichen werden.

6 Wie lässt sich eine IT-Organisation am besten optimieren?

6.1 Due Diligence

Als Due Diligence wird die Prüfung einer Organisation nach den Kategorien Stärken, Schwächen, Chancen und Risiken bezeichnet. Gründe für eine solche Prüfung könnten u.a. die Entwicklung einer IT-Strategie seien oder die Übernahme einer neuen Organisation.

6.2 Business Plan

Der Business Plan beschreibt den Ablauf, die Steuerung, die Vermarktung und die Finanzierung des Geschäfts.

Bestandteile eines Business Plans sind:

- Management Summary
- Unternehmen
- Portfolio
- Markt und Wettbewerb
- Marketing und Vertrieb
- Management und Organisation
- Drei-Jahres-Planung
- Chancen und Risiken Finanzbedarf
- Anlagen

6.3 Business Case

Ein Business Case dient dazu, die Wirtschaftlichkeit zu ermitteln und beinhaltet alle einmaligen sowie laufenden Kosten, den finanziellen Nutzen, sowie en daraus resultierenden Nettonutzen. Außerdem den ermittelten Return on Investment (ROI) und den Amortisationszeitraum.

6.4 Programm- und Projektmanagement

Zum **Programmmanagement** gehört die übergreifende Leitung und Steuerung inhaltlich zusammengehöriger Projekte, welche ein gemeinsames Ziel haben und dessen Termine und Inhalte meist voneinander abhängen.

Das **Projektmanagement** umfasst das Initiiieren, Planen, Steuern, Kontrollieren und Abschließen von Projekten, mit dem Ziel eines effizienten Ressourceneinsatzes.

6.5 Change Management

Änderungen an unternehmenskritischen Komponenten der IT-Landschaft, welche den bestehenden Betrieb von Business-Services gefährden können, werden mit Hilfe eines Change Managements durchgeführt. In einem Change Management werden Änderungsanforderungen (Change Requests) nach ihren Risiken und ihrer Notwendigkeit analysiert. Dies wird gemacht, um vorbeugende Maßnahmen für die Risiken sowie Fall-Back-Maßnahmen festzulegen und außerdem kann an dieser analysiere über die Umsetzung entschieden werden.

7 CIO

7.1 Wie wird man CIO?

Um CIO zu werden hat man zwei Möglichkeiten, zum einen kann man diesen Posten über die klassische Laufbahn in einer IT-Organisation erreichen oder als Quereinsteiger mit einer General-Management-Laufbahn. Doch unabhängig welchen Weg man wählt, es gelten dieselben Anforderungen. Diese sind Engagement, ein hohes Maß an Eigeninitiative, überdurchschnittliche Leistungsbereitschaft, Lernbereitschaft und Offenheit zeigen.

7.2 Aufgaben

Die Aufgaben eines CIOs sind vielfältig, zu seinen Hauptaufgaben gehören, einen stabilen Betrieb der IT-Landschaft zu gewährleisten und Services sowie Innovationen in der vereinbarten Zeit, zum vereinbarten Preis und in der vereinbarten Qualität zu liefern.

7.3 Führungsinstrumente



7.4 Erfolgsfaktoren

Um als CIO Erfolg zu haben sind einige Faktoren sehr wichtig. Zum einen sollte der CIO eine sehr gute Kommunikationsfähigkeit haben, komplexe Sachverhalte verständlich erläutern, erfolgreich verhandeln und gut präsentieren können.

Doch der CIO alleine garantiert keinen Erfolg für die IT-Organisation, denn dazu gehört ebenso ein leistungsstarkes Team, welches selbstständig und eigenverantwortlich Arbeit und den wichtigen Situation vom CIO unterstützt und gefördert wird.

7.5 Networking mit anderen CIOs

Das Netzwerken mit anderen Geschäftspartner, Kollegen oder Bekannten ist wichtig, denn durch diese Beziehungen ist es möglich, auf das Wissen und die Erfahrungen von anderen zurückzugreifen. Bekannte Networking Plattformen sind z.B. LinkedIn.com oder Xing.de welches im deutschsprachigen Raum das bekannteste ist.

7.6 Work-Life-Balance

In Zeiten des mobilen Arbeitens und der dauerhaften Erreichbarkeit, ist die Balance zwischen Berufs- und Privatleben besonders wichtig um sowohl im Job sowie auch Privat Erfolg zu haben.

Um eine gute Balance zu erreichen sollte man in seiner Freizeit mal das Diensthandy ausschalten und sich die Zeit mit Sport vertreiben oder zusammen mit Freunden oder der Familie verbringen.

8 Schlusswort

Durch die immer weiterwachsende Bedeutung der IT in Unternehmen, werden die Unternehmen Marktführer, welche die neuen Technologien richtig nutzen. Denn z.B. ohne die Vernetzung verschiedenster Geräte wäre die Fertigung bzw. der Betrieb von Produkten nicht mehr möglich.

Ebenso ist es durch die Vernetzung von Computersystem und dem Internet der Dinge vereinfacht worden die Kundenbedürfnisse optimal zu erfüllen. Dieser Gesamte Prozess wird auch als Digitale Transformation bezeichnet.

Der CIO eines Unternehmens ist in dem Prozess der Digitalen Transformation besonders gefordert, durch sein technisches sowie geschäftliches Wissen.

Soft Skills

Zusammenfassung des Buches:

Titel: Soft Skills - The software developer's life manual

Verfasser: John Z. Sonmez

Verlag: Manning

Jahr: 2014

ISBN: 978-1617292392

Zusammenfassung von: Gamze Soylev Oektem, Justin Jagieniak, Marvin Schirrmacher,
Daniel Beneker, Tim Jastrzembski

1 Career Teil 1

Einleitung

Nur wenige Softwareentwickler gestalten aktiv ihre Karriere. Aber Erfolg ist nie ein Zufall. Man muss jeden Schritt langfristig planen.

Beginnen mit einem Bang: Mache nicht, was alle anderen machen

Man sollte seine Karriere als ein Geschäft und die Firma, bei der man arbeitet, als den Kunden für sein Geschäft verstehen. Damit fühlt man sich nicht von einer bestimmten Firma abhängig, sondern ist unabhängig und autonom.

Jedes Geschäft muss ein Produkt oder eine Dienstleistung anbieten. Meistens bieten Softwareentwickler Softwareentwicklung als Dienstleistung an. Geschäfte müssen ständig ihre Produkte verbessern. Als Softwareentwickler muss man es genauso machen. Das Produkt ist allein nicht genug, die potentiellen Kunden müssen von dem Produkt Bescheid wissen, um es kaufen zu können. Deswegen ist das Marketing genauso wichtig wie das Produkt. Besseres Marketing bedeutet höhere Preise.

Denk über die Zukunft nach

Als Unternehmen muss man Ziele festlegen. Das ist einfacher gesagt als getan. Man muss sich zuerst ein großes Ziel setzen. Dieses Ziel muss nicht konkret sein, es reicht z.B., wenn man festlegt, wo man sich in 5 oder 10 Jahren sieht. Danach setzt man sich kleinere Ziele (monatlich, wöchentlich, täglich), die dabei helfen, das große Ziel zu erreichen. Man muss die Ziele regelmäßig verfolgen und auf den neuesten Stand bringen, wenn es nötig ist.

People Skills

Das Code-Schreiben ist nur ein kleiner Teil der Verantwortungen eines Softwareentwicklers. Softwareentwickler beschäftigen sich mit anderen Menschen, z.B. schreiben wir unseren Code in High-Level-Programming-Languages, weil wir möchten, dass andere Personen

unseren Code verstehen können. Deswegen muss ein guter Softwareentwickler einen Weg finden, um mit Menschen effizient zu kommunizieren.

Jeder Mensch möchte sich wichtig fühlen. Wenn wir möchten, dass die Kollegin unsere Ideen hören, müssen wir zuerst ihre Ideen hören.

Die Belohnung eines positiven Verhaltens ist immer besser als die Bestrafung eines negativen Verhaltens. Wenn man motivieren möchte, muss man nicht die Kritik benutzen, sondern das Lob.

Wenn wir mit Menschen erfolgreicher handeln möchten, müssen wir daran denken, was diese Person will und was für diese Person wichtig ist.

Soweit wie möglich sollte man Streit vermeiden, weil es nicht immer möglich ist eine Person mit konkreten Gründen zu überzeugen. Bei einer Diskussion, muss man überlegen, ob man diese bestimmte Schlacht wirklich gewinnen muss. Wenn das Thema nicht so wichtig ist, kann man akzeptieren, dass man nicht Recht hat. Dann ist man nächstes Mal stärker.

Vorstellungsgespräche

Vorstellungsgespräche sind für alle schwer, weil man nie wissen kann, was man gefragt werden wird. Allerdings gibt es eine Möglichkeit, um im Vorhinein in eine gute Ausgangsposition zu kommen.

Das Wichtigste ist, dass die Person, die das Vorstellungsgespräch führt, den Bewerber mag. Dies kann man erreichen, bevor das Vorstellungsgespräch anfängt.

Man sollte persönlichen Kontakt mit Mitarbeitern der Firma, bei der man sich bewirbt, aufnehmen und sich am besten mit den Mitarbeitern der Personalabteilung anfreunden, bevor man sich bewirbt. Wenn dies nicht möglich ist, sollte man vor dem Vorstellungsgespräch um ein „Vor-Vorstellungsgespräch“ bitten und Interesse an der Firma zeigen.

Während des Vorstellungsgespräches muss man zeigen, dass man Selbstvertrauen hat und weiß, wie man Dinge erledigt.

Selbst wenn man derzeit nicht nach einem Job sucht, ist es wichtig, dass man seine technischen Fähigkeiten auf dem Laufenden hält und sich auf potentielle zukünftige Vorstellungsgespräche vorbereitet.

Welche Beschäftigungsmöglichkeiten gibt es?

Als Softwareentwickler hat man verschiedene Berufsmöglichkeiten:

- Arbeitnehmer:
 - Vorteile: Stabilität; einfach; bezahlter Urlaub; wenig Verantwortung
 - Nachteile: Mangel an Freiheit; begrenzte Verdienstmöglichkeiten
- Freelancer:
 - Vorteile: Mehr Freiheit; abwechslungsreiche Arbeit; potentiell: höheres Einkommen
 - Nachteile: Es ist schwer, Arbeit zu finden; hohe Ausgaben; mehrere Chefs, anstelle von einem Chef
- Unternehmer:
 - Vorteile: Freiheit; extrem hohes Einkommenspotential; kein Chef
 - Nachteile: Riskant; man ist auf sich allein gestellt; man braucht mehr als nur technische Fähigkeiten; lange Arbeitszeiten Am Anfang ist es meist einfacher als Arbeitnehmer zu arbeiten.

Welche Art von Softwareentwickler bist du?

Es ist wichtig, dass man sich als Softwareentwickler in einem Bereich spezialisiert. Zu sagen, „ich bin ein C#-Entwickler“ ist nicht genug. Die Regel der Spezialisierung lautet: Je mehr man sich spezialisiert, desto weniger potentielle Arbeitsplätze hat man, aber desto wahrscheinlicher ist es, dass man in einem dieser potentiellen Arbeitsplätze eingestellt wird.

Als Softwareentwickler kann man sich in vielen verschiedenen Bereichen spezialisieren, z.B. Programmiersprachen, Plattformen, Methodologien, oder spezifische Technologien. Bei so vielen Möglichkeiten ist es für manche Softwareentwickler schwer, eine Spezialisierung zu wählen. Man sollte beachten, einen Bereich zu wählen, den nur wenige andere Entwickler wählen, damit man bessere Berufschancen hat. Das wichtigste ist immer, eine Spezialisierung zu wählen. Man kann den Bereich nachher immer noch wechseln. Spezialisierung bedeutet außerdem nicht, dass man nur von einem Bereich Ahnung hat - es ist wichtig, von verschiedenen Dingen etwas zu verstehen.

Nicht alle Firmen sind gleich

Als Softwareentwickler kann man bei verschiedenen Firmen arbeiten: Kleine Firmen oder Startups; Mittlere Firmen; Große Firmen.

Meist sind kleine Firmen Startups. Als Softwareentwickler in einem Startup ist man nicht nur Code-Schreiber. Man muss flexibler sein, weil es in diesen Firmen weniger Mitarbeiter gibt. Der Vorteil am Arbeiten in kleinen Firmen ist, dass man häufig aufregende Arbeit hat, bei der man die Ergebnisse direkt sieht. Der Nachteil sind lange Arbeitszeiten, wenig Stabilität und weniger Lohn.

Die meisten Firmen sind Firmen von mittlerer Größe. Diese Firmen bieten mehr Stabilität als kleine und als große Firmen.

Große Firmen sind meist sehr institutionalisiert, es gibt bestimmte Prozeduren für jede Art von Arbeit. Sie bieten viele Möglichkeiten zur Weiterbildung und man hat die Möglichkeit an großen, bedeutenden Projekten zu arbeiten. Ein Nachteil an großen Firmen ist, dass man meist nicht direkt die Ergebnisse seiner Arbeit sieht, weil man nur in einem kleinen Bereich eines umfassenden Projektes arbeitet. Deswegen wird man in großen Firmen leicht übersehen.

Für Softwareentwickler ist es außerdem wichtig, ob man einer Firma arbeitet, die hauptsächlich Software entwickelt, oder in einer Firma arbeitet, deren Hauptaufgabe nicht Softwareentwicklung ist. Deswegen ist es wichtig, dass man sorgfältig darüber nachdenkt, in welcher Art von Firma man arbeiten möchte.

In der Firma aufsteigen

Es ist nicht immer einfach in seinem Job aufzusteigen. Die beste Art im Job aufzusteigen ist Verantwortung zu übernehmen. Wenn einem angeboten wird, Verantwortung zu übernehmen, sollte man dies also unbedingt annehmen. Sollte man keine Verantwortung angeboten bekommen, kann man dies selber erzwingen, in dem man sich z.B. auf in der Firma vernachlässigte Bereiche konzentriert.

Außerdem ist es sehr wichtig, dass man für seine Vorgesetzten sichtbar ist. Dies kann man z.B. dadurch erreichen, dass man seinen Vorgesetzten wöchentliche Arbeitsberichte schreibt.

Ein weiterer Schritt, um in der Firma aufzusteigen besteht darin sich permanent weiterzubilden. Dabei sollte man möglichst versuchen seinem Umfeld zu zeigen, was man lernt.

Des Weiteren sollte man versuchen, wie jemand zu wirken, der Lösungen für alle Art von Problemen findet. Problemlöser sind immer beliebt.

Professionell sein

Es ist sehr wichtig ein professioneller Arbeitnehmer zu sein – und kein Amateur. Ein professioneller Arbeitnehmer nimmt seine Karriere und seine Verantwortungen ernst. Professionelle Menschen wissen nicht alles und geben dies auch gerne zu. Aber sie sind stabil und zuverlässig.

Um ein professioneller Arbeitnehmer zu werden muss man sich gewisse Angewohnheiten, wie z.B. effektives Zeitmanagement, aneignen.

Außerdem muss man als Softwareentwickler technisch und ethisch richtige Entscheidungen treffen – selbst wenn dies kurzfristige Nachteile bedeuten. Man muss zu seinen Prinzipien stehen.

Professionell bedeutet außerdem permanent daran zu arbeiten, sich selbst zu verbessern und hohe Qualität zu liefern.

Freiheit: Den Job kündigen

Wenn man seine eigene Firma möchte ist es sehr riskant seinen Job bei einer anderen Firma einfach so zu kündigen. Bevor man seinen Job kündigt sollte man anfangen die Firma, die man gründen möchte, aufzubauen und quasi einen Nebenverdienst zu haben.

Die meisten Menschen unterschätzen, wie schwer es ist, sich selbstständig zu machen. Die Arbeitszeiten sind z.B. sehr lang. Wenn man seine eigene Firma als Nebenverdienst aufbaut, bekommt man ein Gefühl dafür, wie schwer es ist, seine eigene Firma aufzubauen. Man sollte außerdem wissen, dass die meisten neugegründeten Firmen pleitegehen.

Wenn man selbstständig ist, muss man viel härter arbeiten. Als normaler Mitarbeiter in einer Firma verschwendet man die Hälfte der Arbeitszeit mit anderen Dingen und arbeitet eigentlich nur vier Stunden am Tag. Als Selbstständiger ist das nicht möglich.

Freelancing

Um als Freelancer zu arbeiten, sollte man freelancing als Nebentätigkeit anfangen und mit der Zeit baut man einen Kundenstamm auf.

Wie bekommt man seinen ersten Kunden? Höchstwahrscheinlich wird der erste Kunde jemand aus dem Bekanntenkreis sein, da Bekannte eher bereit sind einem zu vertrauen. Sollte man keine Bekannten haben, die potentielle Kunden sein könnten, sollte man Inbound Marketing anwenden. Inbound Marketing ist, dass man nicht Kunden sucht, sondern das Kunden selber zu einem kommen. Es gibt verschiedene Wege um das zu schaffen. Z.B. kann man etwas kostenfrei anbieten (bspw. in einem Blog) oder Email-Marketing machen.

Freelancing ist riskanter als Arbeitnehmer zu sein. Es gibt mehr Ausgaben (z.B. Steuern, Sozialabgaben, Nebenkosten etc.). Daher sollte man als Faustregel doppelt so viel Lohn verlangen, wie man als Arbeitnehmer bekommt. Aber man kann den Lohn nur so hoch ansetzen, wie der Markt bereit ist zu zahlen. Deswegen braucht man einen hohen Bekanntheitsgrad.

Das erste Produkt

Jedes Produkt löst ein Problem. Wenn das nicht so ist, hat das Produkt keinen Zweck. Deswegen muss man erst überlegen wer das Produkt kaufen soll, d.h. wer die Zielgruppe sein soll. Manchmal ist es besser, zuerst einige Leute zu finden, die bestimmte Probleme haben und eine Lösung für diese bestimmten Probleme zu finden.

Bevor man sein Produkt entwickelt, sollte man rausfinden, ob die Zielgruppe wirklich für das Produkt Geld ausgeben würden. Dies kann man schaffen indem man z.B. potentiellen Interessenten vorab die Möglichkeit gibt, Geld zu investieren.

Für das erste Produkt ist es außerdem ratsam mit einem kleinem Produkt anzufangen.

Möchtest Du ein Startup gründen?

Es gibt zwei Arten von Startups. Die erste Art von Startups versuchen Geld von Investoren zu bekommen. Diese Startups haben die Absicht zu großen Firmen zu werden, können aber leicht scheitern. Die zweite Art von Startups, Bootstrapped Startups werden von ihren Gründern finanziert. Diese Startups bleiben meist kleinere Unternehmen, aber scheitern weniger häufig.

Die meisten Startups haben das Ziel, irgendwann einen großen Profit zu erwirtschaften. Investoren haben fast immer eine Exit-Strategie. Die kann entweder darin bestehen das Startup an eine große Firma zu verkaufen oder das Unternehmen an die Börse zu bringen.

Ein Startup basiert meist auf einer Idee. Ein guter Startup hat a) eine einzigartige Idee oder Erfindung, die sich schwer kopieren lässt und b) das Potential stark zu wachsen.

Eine wichtige Hilfe, um mit einem Startup erfolgreich zu sein, kann man mit einem Startup-Accelerator-Programm bekommen. Accelerator-Programme bieten Hilfe und auch ein bisschen Geld für Startup-Firmen und bekommen im Gegenzug Anteile vom Startups. Viele große Technologieunternehmen waren zu Beginn in einem Accelerator-Programm (z.B. Dropbox).

Nach dem Accelerator-Programm muss das Startup seinen ersten Investor (seed money) finden. Mit diesem Geld baut man sein Business-Modell auf. Wenn man dieses Geld verbraucht hat, muss man neue und große Investoren finden (venture capitalists). Diese Finanzierung wird Series A genannt. Wenn dieses Geld nicht ausreicht um die Firma groß genug zu machen und sie zu verkaufen, muss man erneut große Investoren finden. Dieser Prozess wiederholt sich so lange, bis man scheitert oder die Firma verkauft.

1 Career Teil 2

Working remotely survival strategies

In diesem Unterkapitel diskutiert der Autor über Strategien für den Fall, dass man von zuhause aus arbeitet. Er spricht davon, dass heutzutage eine Menge unabhängiger Software-Entwickler per Remote-Desktop oder über ein virtuelles Büro von zuhause aus arbeiten. Er geht dabei auf die Vor- und Nachteile der Heimarbeit ein. Als Nachteil sieht er zum Beispiel die Isolation, Einsamkeit und mangelnde Selbstmotivation. Er erwähnt, dass er anfangs die Vorteile dabei sah, aus seinem Bett morgens direkt an den Arbeitsplatz zu kommen. Sagt aber auch, dass er die entstandenen Herausforderungen unterschätzt hat. Die erste Herausforderung für ihn ist das Zeitmanagement. Er behauptet, dass es bei der Arbeit zuhause eine Menge Arten von Ablenkung gibt, die einen daran hindern die Arbeit zu erledigen. Auch neigt man angeblich dazu, seine Arbeit später am Abend zu erledigen, wenn man sich die Zeit abseits vom Alltagsstress nehmen kann. Seiner Meinung nach endet das aber in einem Desaster. Er behauptet, wenn man von zuhause aus arbeiten will, muss man ein vernünftiges Konzept für das Zeitmanagement erstellen.

Eine weitere Herausforderung ist die Selbstmotivation. Er empfiehlt Leuten die Probleme mit Disziplin und Selbstkontrolle zu haben, nicht von zuhause aus zu arbeiten. Er meint, dass wenn man im Büro arbeitet, der Chef beobachten kann, wie man seine Arbeit erledigt. Wenn man allerdings von zuhause aus arbeitet, kann dies der Chef nicht. Das heißt, dass man für seine Disziplin und Motivation selbst verantwortlich ist. Er hält eine Planung und Routine der Zeiten bei der Heimarbeit für sehr wichtig, vor allem für die Phasen bei denen man nicht motiviert ist. Auch sollte man Versuchungen und Ablenkungsmöglichkeiten von seinem Arbeitsplatz entfernen. Wenn man sich absolut unmotiviert fühlt, schlägt er vor eine Zeituhr auf 15 Minuten zu stellen und sich in diesen 15 Minuten zu seiner Arbeit zwingt. Meist ist es der Fall, dass man nach jenen 15 Minuten mehr motiviert ist seine Arbeit fortzusetzen. Ein weiteres Hilfsmittel ist das Nutzen der Kommunikation mit Mitarbeitern per Google Hangouts oder Skype, um nicht sozial von der Arbeit abhängig zu werden.

Fake it till you make it

Hier spricht der Autor davon, dass man immer wieder auf Herausforderungen und Grenzen stößt, auf die man nicht vorbereitet ist. Er sagt, dass es einige Leute gibt, die der Herausforderung lieber ausweichen und andere die sie annehmen und kämpfen. Er behauptet, dass es nicht an ihrer Selbstsicherheit und Fähigkeit liegt, Erfolg zu haben, sondern dass sie alle in der Lage sind vorzutäuschen, dass sie es schaffen. Er meint, dass

das Vortäuschen etwas zu Können bis man es schafft, etwas mit Selbstsicherheit zu tun hat. Man handelt nach einem großen Glauben an sich selbst alle Hürden überwinden zu können. Man muss dazu bereit sein ins große Unbekannte springen zu wollen, auch wenn man dann nicht weiß was man tun muss und Angst dabei empfindet. Er hält es nämlich für unmöglich als Softwareentwickler ein Experte für alles zu sein. So würden auch die meisten Jobinterviews Fähigkeiten erfordern, die man noch nicht besitzt. Er meint, dass das Schlüsselwort hier das "noch nicht" ist, dass es wichtig ist, sein Auge auf die Zukunft zu richten statt auf das, was man noch nicht kann. Daher ist es wichtig eine Aura von Selbstsicherheit auszustrahlen mit dem Wissen, dass man Herausforderungen in der Vergangenheit gemeistert hat und es keinen Grund gibt, dass man sie nicht auch in der Zukunft meistern kann. Weiterhin soll man nach Meinung des Autors kein Lügner sein, sondern ehrlich mit den eigenen Fähigkeiten sein und zeigen, dass man in der Lage ist mit Hindernissen umzugehen.

Resumes are boring - Let's fix that

Im nächsten Unterkapitel behauptet der Autor, dass der durchschnittliche Lebenslauf eines Softwareentwicklers ein fünfseitiges monströses Dokument ist, welches eine Schriftart und zwei Spalten hat, und mit grammatischen Fehlern, Tippfehlern und erbärmlich strukturierten Sätzen voll mit Phrasen, wie "anführend" und "auf Ergebnisse fokussiert" gefüllt ist. Er vergleicht dies mit einem professionellen Lebenslauf und kommt zum Ergebnis, dass man lieber einen professionellen Lebenslauf-Schreiber anheuern sollte. Dies begründet er damit, dass das Schreibenlernen eines professionellen Lebenslauf eine Verschwendug von Zeit und Talent ist und dies besser in professionelle Hände gehört. Bei der Beauftragung eines professionellen Schreibers ist seiner Meinung nach darauf zu achten, dass er technisches Know-how besitzen sollte und auch Beispiele aus dem technischen Bereich zeigen kann. Bei der Beauftragung selbst sollte man beachten, dass man so viele Informationen wie möglich von sich selbst gibt und versucht sich selbst in einen positiven Licht mit den Informationen darstellen zu lassen. Auch ist es wichtig einen Format zu wählen, das für den Leser einfach verständlich ist.

Weiterhin regt der Autor an, dass man seinen Lebenslauf, wenn es fertiggestellt ist, auch online auf einer Homepage und Portalen wie LinkedIn zur Verfügung stellen sollte. Auch dies würde häufig von einer professionellen Fachkraft übernommen.

Wenn man trotzdem seinen Lebenslauf lieber selbst schreiben will, rät der Autor folgendes zu beachten: Man soll seinen Lebenslauf online stellen, ihn in einzigartiger Weise präsentieren, vorherige Projekte im Lebenslauf präsentieren und er sollte frei von Tipp- und Sprachfehlern sein.

Don't get religious about technology

Im letzten Unterkapitel warnt der Autor davor bei Technologien religiös zu werden. Damit meint er, dass Leute, die eine Technologie, Programmiersprache oder Software kennen, gerne an dieser festhalten ohne offen für etwas anderes zu sein. Er hält dieses Verhalten für destruktiv und limitierend. Wir würden einen Punkt erreichen, an dem wir aufhören zu "wachsen" und meinen auf alles eine Antwort gefunden zu haben. Er sagt von sich, dass er aufgehört hat religiös zu sein und in einem Punkt seiner Karriere verschiedene Betriebssysteme, Programmiersprachen und Texteditoren ausprobiert und gelernt hat, bevor er sich für seine beste Technologie entschieden hat.

Der Autor meint aus seiner Perspektive betrachtet, dass nicht alle Technologien großartig sind, sondern manche nur gut sind. Von Zeit zu Zeit kann sich dies auch ändern. Er sagt, dass es nicht nur die eine gute oder gar die beste Lösung für ein Problem gibt. Man entscheidet danach, was man am besten findet, aber das heißt nicht notwendigerweise, dass es das Beste ist. Der Autor erzählt, dass er in seiner früheren Zeit darüber debattierte, ob Microsoft oder Mac besser ist, C# besser ist als Java oder statische Programmiersprachen besser sind als dynamische. Er sagt, dass seine eigene Ansicht falsch war. Er hat in den letzten Jahren an einem Javaprojekt gearbeitet und gelernt für alles offen zu sein. Er versucht nun die Dinge auszuprobieren, bevor er eine Entscheidung trifft. Sein Punkt dabei ist, dass man in seinen Optionen keine Limits setzen sollte. Man sollte nicht seine Wahl der Technologie für die beste erklären und alles andere ignorieren. Es wird seiner Meinung nach am Ende einem nur weh tun. Andererseits sagt er, dass nur dann wenn man gewillt ist offen zu sein und man etwas für das Beste erklärt, andere einen folgen werden.

2 Marketing yourself

Marketing-Grundlagen für Programmier-Spezialisten

Der Unterschied zwischen einem erfolgreichen Musiker und einem Superstar ist ihre Vermarktung, denn Marketing ist ein Talent-Multiplikator.

Was ist Selbstmarketing?

Selbstmarketing bedeutet, sich mit Menschen zu verbinden, die etwas haben wollen, was du bei dir hast oder in dir trägst. Es bedeutet, für andere einen (Mehr-)Wert zu generieren. Dabei kontrolliert die Person, die sich selbst vermarktet, was sie anderen vermittelt, welche Nachricht sie sendet und welches Bild sie vermitteln möchte.

Warum Selbstmarketing wichtig ist

Selbstmarketing garantiert keinen Erfolg, aber es ist ein sehr wichtiges Element, welches du kontrollieren kannst. Viele Software-Entwickler können ein sehr hohes Kompetenz-Level innerhalb von zehn Jahren in ihrer Karriere erreichen. Danach kann es sehr schwierig sein, diese Kompetenzen weiter auszubauen, weil der intellektuelle Input von anderen Menschen fehlt. Das individuelle Talent wird deutlich an Bedeutung verlieren, weil jeder Software-Entwickler mit allen anderen Software-Entwicklern konkurriert, die ähnliche Kompetenzen besitzen wie das einzelne Talent. Durch Selbstmarketing kannst du herausfinden, was dich auszeichnet und welche deine Kompetenzen sind.

Wie du dich selbst vermarkten kannst

- Alles beginnt damit, eine Markenpersönlichkeit von dir zu kreieren; etwas, was dich widerspiegelt. Treffe bewusste Entscheidungen darüber, wer du sein möchtest und wie du dieses Bild in der Welt darstellen möchtest. Außerdem solltest du eine familiäre Atmosphäre erzeugen, wenn dir jemand entgegentritt oder wenn es sich um ein Produkt handelt, welches du schon viele Male kreiert hast. Benutze viele Kanäle für das Selbstmarketing. Ein Blog kann die Basis für deinen Internet-Auftritt bilden, weil du dort den Informationsfluss vollständig steuern kannst und weil du so nicht von anderen Menschen und ihren Plattformen und Regeln abhängig bist. Dazu gibt es eine Strategie von dem Unternehmer Pat Flynn, die "Be everywhere" genannt wird. Sie besagt, dass

du überall dort sein musst, wo du dich selber vermarkten möchtest; dies hat das Ziel, dass du überall dort auch eine gute Chance hast, von deiner gewünschten Zuhörerschaft gehört zu werden. Mögliche Kanäle sind Blog-Posts, Podcasts, Videos, Artikel in Fachzeitschriften, Bücher, Code-Camps und Konferenzen. Dabei solltest du nicht vergessen, dass all dies von deiner Fähigkeit abhängt, (Mehr-)Werte für andere zu generieren.

Erschaffe eine Marke, die auf dich aufmerksam macht

Was ist eine Marke

Das Logo eines Unternehmens ist eine visuelle Erinnerung an seine Marke, aber es ist nicht die Marke selbst. Wenn du dieses Logo siehst, hast du Erwartungen an die Dienstleistung oder das Produkt. Eine Marke ist ein Versprechen, einen bestimmten (Mehr-)Wert zu erzeugen, in der Art, in der du es selber erwartest.

Was eine Marke ausmacht

Du brauchst vier Dinge für eine erfolgreiche Marke: eine Nachricht, Visuelles, Konsistenz und eine wiederholte Marken-Darstellung: Eine Nachricht ist das, was du versuchst zu vermitteln und Gefühle, die du mit deiner Marke versuchst, hervorzurufen. Visuelles repräsentiert deine Marke: Ein Logo ist eine einfache Repräsentation, ein Set an Farben und ein Style, der die Marke repräsentiert, machen auf sie aufmerksam. Sei konsistent, sodass ein Kunde keine wechselnden Erfahrungen mit deiner Marke hat. Eine wiederholte Marken-Darstellung ist essentiell, weil ein Kunde sich dadurch an dich und deine Marke erinnert, wenn er das Visuelle deiner Marke einige Male gesehen hat. So kann er diese Erfahrungen mit deiner Dienstleistung oder mit deinem Produkt verknüpfen.

Kreiere deine eigene Marke

Entscheide zunächst, was du repräsentieren möchtest und definiere deine Nachricht dafür. Verkleinere deine Zuhörerschaft und wähle eine Nische aus (Spezialisierung). Der beste Weg für die Entscheidung, welche Nische du wählen sollst, erfolgt von einer rein strategischen Perspektive: Welchen Vorteil hast du von einer bestimmten, spezifischen Nische, die du nutzen kannst? Zögere nicht, deine Nische später zu verändern, wenn dies nötig ist.

Schritte, eine Marke zu kreieren, sind: Definiere deine Nachricht, wähle deine Nische, erschaffe dein Motto, kreiere einen Elevator-Pitch und designe die visuellen Komponenten

deiner Marke. Starte mit deinem Motto (Slogan), welches deine Marke in einem einzigen oder in zwei Sätzen darstellt. Ein Elevator-Pitch ist eine kurze Beschreibung dessen, was du machst und des einzigartigen Wertes, welchen du erschaffst (dies kann in der Zeit erklärt werden, die man benötigt, um mit einem Fahrtstuhl zu fahren). Dieser Elevator-Pitch stellt sicher, dass du und deine Marke konsistent seid und dass ihr immer die gleiche Nachricht vermittelt. Visuelle Komponenten sollten dabei helfen, diese Nachricht zu übermitteln und als visuelle Erinnerung dessen dienen, was deine Marke darstellen soll.

Einen sehr erfolgreichen Blog kreieren

Warum Blogs so wichtig sind

Ein Blog kann eine Menge an Informationen über einen Entwickler enthalten; z. B. Programmier-Beispiele und -sammlungen, sowie detaillierte technische Analysen verschiedenster Aspekte der Software-Entwicklung. Ein Blog kann mehr Informationen übermitteln, als jede andere Art der Kommunikation wie Interviews oder Zusammenfassungen. Durch einen Blog kannst du einen besseren Job erlangen, du kannst aber auch ein besserer Software-Entwickler und Kommunikator werden. Ein Blog kann dir Möglichkeiten verschaffen, die du dir vielleicht nie zuvor vorgestellt hast. Wenn du ein Freelancer bist, kannst du herausfinden, dass ein erfolgreicher Blog viele Kunden zu dir führen kann ohne dass du nach ihnen suchen musst. Ein Kunde, der zu dir kommt ist bereit viel mehr Geld zu zahlen. Es wird viel einfacher sein, ihn von deinem Talent und deinen Kompetenzen zu überzeugen, sodass du z. B. einen Job von ihm bekommst. Bei genügend Blog-Traffic kannst du den Blog als Plattform für die Vermarktung deiner Produkte nutzen oder ein Produkt erzeugen, welches sich die Blog-User wünschen.

Kreiere einen Blog

Anfangs kannst du eine kostenlose Software für deine Blog nutzen (z. B. WordPress). Später, oder wenn du einen Shop oder Ähnliches benötigst, ist es ratsam einen kostenpflichtigen Service zu nutzen, da so Vieles einfacher ist. Registriere deine eigene Domain, um SEO-Kriterien zu erfüllen.

Schlüssel zum Erfolg

Der wichtigste Aspekt auf dem Weg zum Erfolg als Blogger ist Konsistenz. Dies ist sehr wichtig, da dich so viele Menschen über Suchmaschinen finden können. Je höhere Qualität dein Inhalt hat, desto wahrscheinlicher ist es, dass Menschen zu deinem Blog zurückkehren

oder dort etwas schreiben oder ihn in den sozialen Medien teilen, sodass dein Blog mit anderen Websites verlinkt wird (wichtiger Aspekt um in Suchmaschinen gefunden zu werden).

Erhalte mehr Traffic

Starte damit, die Blogs anderer Leute zu kommentieren, die ähnliche Themen behandeln. So kommst du auch mit anderen Bloggern in Kontakt. Teile deine Blog-Posts in den sozialen Medien und verweise in deiner E-Mail-Signatur auf deinen Blog. Vereinfache es, deine Inhalte zu teilen, indem du Teil-Buttons verwendest. Schließlich, wenn deine Inhalte gut und kontrovers genug sind, kannst du deine Posts an Seiten wie Reddit oder Hacker News schicken.

Ich kann dir keinen Erfolg garantieren

Dein oberstes Ziel: Generiere einen (Mehr-)Wert für andere

- "If you help enough people get what they want, you will get what you want" - Zig Ziglar

Gib den Menschen, was sie möchten

Herauszufinden, was Menschen wirklich möchten, ist nicht einfach - vor allem, wenn sie dies selber nur recht vage wissen. Versuche, die Zeichen zu lesen. Gehe raus und versuche herauszufinden, für was die Menschen sich interessieren. Über welche Themen wird in Internet-Foren gesprochen, die sich auf deine Nische beziehen und die relevant für dich sind? Welche Trends siehst du allgemein in der Industrie? Welche Ängste haben die Menschen und wie kannst du diese Ängste adressieren?

Stelle 90% deiner Leistung kostenlos zur Verfügung

Kostenlose Inhalte werden häufiger geteilt als solche, für die man bezahlen muss. Kostenloses zur Verfügung Stellen ermöglicht es Kunden, den Wert deiner Inhalte zu erfahren, ohne dass sie dafür vorher Geld investieren müssen. Dadurch kannst du die Menschen viel einfacher davon überzeugen, für dein Produkt zu zahlen, das sie sich über die hohe Qualität deines Produktes schon vorher im Klaren.

Der schnelle Weg zum Erfolg

Wie kann dein Inhalt einen (Mehr-)Wert für andere generieren? Dein größter Erfolg ist es, wenn du die Probleme anderer Menschen lösen und ihnen auf diese Weise helfen kannst.

Biete mehr von dir an

Die produktivsten Menschen sind die hilfreichsten.

#UsingSocialNetworks

Vergrößere dein Netzwerk

Folge anderen Leuten oder frage ander in deinem Netzwerk beizutreten. Viele Entwickler warten geradezu auf Leute, die ihnen folgen oder mit ihnen interagieren. Platziere deine Links zu deinen Social Media-Profilen in deine Online-Biographien, ans Ende deiner Blog-Beiträge oder auch in deine Email-Signaturen.

Nutze soziale Medien effektiv

Bewege Leute dazu von einfachen Followern zu deinen Fans zu werden, sodass sie sich mehr mit deinen Inhalten befassen und deine Reputation in der Branche aufbauen. Poste alles, was du nützlich oder interessant findest, denn wenn du es wertvoll findest, finden es andere es wahrscheinlich auch. Mögliche Inhalte, die du teilen und veröffentlichen kannst wären:

- finde beliebte Blog-Einträge oder teile deine eigenen
- teile interessante Artikel, die wenn möglich inhaltlich zu deiner Nische passen oder allgemein zum Thema Software-Entwicklung gehören
- bekannte Zitate, besonders die inspirierenden sind besonders beliebt
- spezielles Wissen, dass du hast und jemand anderes wertschätzt
- ein bisschen Humor ist in Ordnung, aber sei dir sicher, dass es nicht zu provokativ, sondern einfach lustig ist
- mache ein bisschen von allem

Bleibe stetig aktiv

Du solltest ein oder zwei Plattformen wählen, auf denen du besonders engagiert bist, da du nicht alle Plattformen bedienen kannst, ohne sehr viel deiner Zeit damit zu verbringen. Ein Tipp ist es deine Social Media-Beiträge zu terminieren, sodass sie zu verschiedenen Zeiten innerhalb einer Woche veröffentlicht werden. Am besten parallel auf allen deinen Plattformen. Ein Werkzeug dafür wäre z.B. Buffer.

Netzwerke und Profile

Du solltest eine Präsenz besonders in den Technologie-relevanten und Karriere-fokussierten sozialen Netzwerken haben. Es ist empfehlenswert ein Twitte-Konto zu haben, da viele Entwickler Twitter nutzen und es leichte Mechanismen hat, um auf Tweets von anderen zu antworten und ebensolche zu referenzieren. Zudem solltest du ein LinkedIn-Konto haben, weil es ein soziales Netzwerk speziell für Fachleute ist und dadurch besonders gut für's Netzwerken ist. Eine sehr empfehlenswerte Funktion von LinkedIn ist dabei die Möglichkeit seine eigenen Kunden nach sog. "Endorsements" zu fragen, was es möglich macht Feedback für deine Aufträge zu erhalten, die du in deinem Profil aufführst.

Vorträge, Präsentationen und Schulungen: Sprachkünstler

Vorträge halten oder Schulungen geben sind die effektivsten Wege mit anderen Leuten in Kontakt zu treten, auch wenn der Einflussradius dabei nicht sehr groß ist im Vergleich zu anderen Methoden. Aber vor einem Publikum zu stehen und direkt zu ihm zu sprechen ist eines der wirkungsvollsten Dinge, die du tun kannst.

Warum live vortragen so wirkungsvoll ist

Es entsteht eine persönliche Verbindung, wenn du ein Live Event besuchst, die du nicht bekommst, wenn du einfach eine Aufzeichnung hörst oder schaust. Auf diese Art ist wahrscheinlicher, dass Leute sich an dich erinnern und eine persönliche Verbindung zu dir aufzubauen. Zudem kann Vortragen ein interaktives Medium sein, bei dem du direkt auf Fragen des Publikums eingehen kannst und dieses sich an deiner Präsentation beteiligt.

Wie man mit dem Sprechen beginnen kann

Beginne damit Präsentation an deinem eigenen Arbeitsplatz zu halten über dort relevante Themen. Die meisten Unternehmen sind glücklich darüber, wenn ihre eigenen Mitarbeiter Themen intern vorstellen. Besuche sog. User Groups bei denen du nach einer Weile den Organisator fragen kannst, ob du über ein bestimmtes Thema vortragen kannst. Eine User Group ist dabei eine kleinere Gruppe und ein Publikum, das vergibt. Zusätzlich gibt es jährlich Code Camps überall auf der Welt, bei denen jeder mit beliebigen Erfahrungs niveau vortragen darf. Diese Veranstaltungen sind Situation mit wenig Druck, da niemand für die Vorträge zahlt. Sobald du einige Erfahrung gesammelt hast, kannst du an Entwicklerkonferenzen teilnehmen.

Was ist mit Schulungen?

Online Video-Schulungen sind deutlich leichter und skalierbarere Lösungen für Entwickler, um sich Reputation aufzubauen. Man kann mit einem einfachen Screencast anfangen und diesen auf einer freien Video-Seite teilen. Danach kannst du damit anfangen dich für deine Inhalte, die du produzierst, bezahlen zu lassen. Der einfachste Weg sind Portale die dich für deine Inhalte bezahlen und dir einen Teil vom darauf generierten Profit geben in Form von Autorenhonoraren. So hast du dich nicht über Vermarktung und Vertrieb zu kümmern.

Schreibe Bücher und Artikel die eine Anhängerschaft generieren

Warum Bücher und Artikel wichtig sind

Wenn du als jemand glaubwürdiges in deiner Branche angesehen werden willst, solltest du ein Buch schreiben oder Artikel in Software-Entwicklungs-Zeitschriften veröffentlichen. Ein Buch ist eine Möglichkeit deine Nachricht sehr gezielt und fokussiert zu übermitteln. Denn wenn sich jemand hinsetzt, um ein Buch zu lesen, bekommst du als Autor die fokussierte Aufmerksamkeit des Lesers für eine lange Zeitspanne.

Bücher und Zeitschriften bezahlen nicht

Du schreibst ein Buch nicht, um Geld zu machen, sondern du schreibst ein Buch, um deine Reputation zu erhöhen. Die meisten Zeitschriften bezahlen nur sehr wenig für einen Artikel, während das Schreiben und Bearbeiten sehr lange dauern kann. Jedoch agiert die Verlagswirtschaft als eine Art Gate-Keeper für Qualität. Zudem wirst du erkennen, dass es viele andere lukrative Möglichkeiten, die sich indirekt selber präsentieren und publiziert werden können. Autoren, die bereits publiziert haben, werden es einfacher haben zu Konferenzen eingeladen zu werden und können sich selber als Autorität in einem bestimmten Themenbereich etablieren. Da führt wiederum zu mehr Kunden und besseren Jobangeboten.

Publiziert werden

Publiziert werden ist nicht einfach, besonders. Getting published isn't easy - especially for your first book as not too many publishers want to take a risk on a completely unknown author. Best way to give yourself an opportunity to get published is to have a clearly defined topic that you know there's a market for and you can demonstrate your knowledge as an

expert in that area. Start with blog posts, magazine articles and so get bigger and bigger. Publishers like to publish authors who already have a fairly large audience. You should have a solid proposal or magazine abstract

Sich selber publizieren

More authors are finding success by self-publishing - especially if they have an existing audience. It is easy to do and a good training before entering into a contract with a publisher that will have deadlines that you'll be required to meet. There are many services you can use to help you self-publish your book like Leanpub, Amazon Kindle Direct Publishing, Smashwords or BookBaby

Habe keine Angst, dumm dazustehen

Alles ist anfangs unangenehm

When you first do something that makes you feel uncomfortable, you can't imagine how you could ever feel comfortable doing that thing. But you just have to learn to overcome this kind of thinking and realize that almost everyone goes through the same kind of uncomfortable feelings when they first do anything challenging - especially in front of a group of people

Es ist in Ordnung, wie ein Idiot zu erscheinen.

Things will get easier over time. And when things go wrong while presenting you should just don't care. And after it's over, chances are no one will even remember it. If you want to succeed, you have to learn how to swallow your pride and get out there and not be afraid to make a fool of yourself

Mache kleine Schritte (oder springe ins kalte Wasser).

Just diving in is the most effective way. But if you're nervous about doing speaking, writing, or something else, try to think of the smallest thing you can do that doesn't make you quite as nervous. Start with commenting and contribute to conversations but be prepared for criticism. Once you feel a bit braver, write your own blog posts like "how-tos". From there expand further by writing a guest post for someone else's blog or you can be interviewed on a podcast. And you might even join a club like Toastmasters to help you get used to speaking in public.

3 Lernen

Als Softwareentwickler arbeitet man in der Regel mit einer großen Anzahl an Technologien zusammen. Technologien, die heute aktuell sind, sind es morgen eventuell nicht mehr. Um auf dem neusten Stand zu bleiben, muss man sich daher ständig in neue Technologien einlesen.

Einer der wichtigsten Soft Skills ist daher, sich selbst etwas beizubringen. Doch wie lernen wir etwas am effektivsten? Es ist ein Mythos, dass wir alle auf verschiedenen Arten lernen. Wir lernen alle am besten, indem wir etwas ausprobieren oder es jemandem erklären und wir tendieren dazu einfacher zu lernen, wenn es uns interessiert. Man kann noch so viele Bücher übers Fahrradfahren lesen und noch so viele Videos schauen, wenn man es das erste Mal ausprobiert, wird es nicht sofort funktionieren. Trotzdem greifen viele Softwareentwickler zu einem Buch, um sich in eine neue Programmiersprache einzulesen. Anstatt dessen sollte man jedoch möglichst früh versuchen, etwas praktisch zu tun. Wir sind von Natur aus kreativ und neugierig. Nutzen wir diese Aspekte aus, können wir sowohl unsere Motivation, als auch unsere Lerngeschwindigkeit erhöhen.

Um eine neue Technologie zu erlernen, sind im Kern drei Punkte wichtig:

- Was benötigt man um anzufangen?
- Was kann ich grob damit tun?
- Was sind die Grundlagen? Dass heißt welche 20% muss ich lernen, um 80% meiner täglichen Aufgaben zu erledigen?

Die 10 Schritte

Dieses Kapitel soll anhand von zehn Schritten zeigen, wie wir diese Fragen beantworten können und es schaffen, möglichst schnell und effektiv etwas neues zu erlernen.

Schritt 1: Sich einen Überblick verschaffen.

Im ersten Schritt geht es darum, zu verstehen worum es grob geht. Dafür reicht in der Regel eine einfache Internetrecherche aus. Auch das Lesen von Einleitungen entsprechender Bücher kann hilfreich sein. Ziel ist es, die Größe des Themas zu bestimmen. Welche Unterthemen gehören beispielsweise zu dem Thema? In diesen Schritt sollte nicht zu viel Zeit investiert werden.

Schritt 2: Festlegen, was ich lernen will.

Man kann nicht alles lernen, daher sollte man den Fokus auf ein bestimmtes Thema richten. Möchte man beispielsweise etwas über digitale Fotografie lernen, so wäre ein Fokus alles über das Schießen von Porträt-Fotos zu lernen. Wichtig ist, dass man den Fokus auf ein einziges Thema richtet, denn wir können nicht mehrere Sachen auf einmal lernen.

Schritt 3: Das Lernziel festlegen.

Bezogen auf das Beispiel mit der digitalen Fotografie kann das bedeuten, alle Funktionen der eigenen Kamera beschrieben und nutzen zu können und erklären wann und warum man welche Funktion benutzt. Diese Lernziele sollten möglichst eindeutig sein. Denn so können wir später feststellen, ob wir dem Ziel näher kommen. Ein weiterer Vorteil ist, dass wir ein konkretes Ziel vor Augen haben, welches wir erreichen wollen.

Schritt 4: Quellen suchen.

Es ist wichtig, nicht nur mit einer Quelle zu lernen. Es gibt viele verschiedene Arten von Quellen wie Bücher, Videos, Blogs, andere Experten, Programmcode, Beispielprojekte oder Dokumentationen. In diesem Schritt sollten erstmal, ähnlich wie bei einem Brainstorming, alle Quellen, die in Frage kommen, zusammengetragen werden. Die Qualität der Quelle ist an dieser Stelle weniger relevant.

Schritt 5: Einen Lernplan erstellen.

Bei den meisten Themen bietet es sich an, bestimmte Unterthemen in einer bestimmten Reihenfolge zu lernen. Zum Erstellen dieses Planes kann man sich oft an den Inhaltsverzeichnissen der Bücher (aus Schritt 4) orientieren. Oder man schaut mit welcher Struktur andere das Thema erklären.

Schritt 6: Die Quellen filtern.

Viele Quellen werden sich thematisch überschneiden und meistens reicht auch die Zeit nicht aus, alle Quellen durchzuarbeiten. Deshalb ist es wichtig, die Quellen entsprechend zu filtern. Die ausgewählten Quellen sollten natürlich die im Lernplan ausgewählten Bereiche abdecken. In diesem Schritt sollte auch auf die Qualität der Quellen geachtet werden. Bei der Auswahl von Büchern kann es z.B. hilfreich sein, Amazon Bewertungen durchzulesen.

Die folgenden Schritte 7-10 sollten für jedes Modul aus dem Lernplan durchlaufen werden. Die Schritte folgen dem Prinzip „LDLT: learn, do, learn, teach“.

Schritt 7: Genug lernen, um anzufangen.

Wichtig ist, möglichst früh praktische Erfahrungen zu sammeln, denn wir lernen am besten, indem wir etwas tun. In diesem Schritt geht es darum, nur die grundlegenden Sachen zu lernen. Das kann beispielsweise das Durchlaufen eines „Hello-World-Beispiels“ sein oder das Einrichten der Entwicklungsumgebung. Eventuell reicht es auch schon, eine Kapitelzusammenfassung eines Buches zu lesen.

Schritt 8: Freies experimentieren.

Nutze deine Neugier und Kreativität, probiere etwas aus, bis du an einen Punkt kommst, an dem du nicht mehr weiterkommst. In diesem Schritt werden sich viele Fragen ergeben. Es kann hilfreich sein, diese aufzuschreiben.

Schritt 9: Genug lernen, um etwas sinnvolles zu tun.

Die Fragen, die sich im achten Schritt ergeben haben, sollen in diesem Schritt beantwortet werden. An dieser Stelle sollen die Quellen aus Schritt 4 intensiv genutzt werden. Der Focus sollte aber immer darauf liegen, die Fragen zu beantworten. Dass bedeutet eventuell nur einzelne Kapitel eines Buches anlesen, in denen man die Antwort auf eine Frage vermutet. Wichtig ist auch, zu prüfen ob man dem, in Schritt 3 definierten Ziel, näher kommt.

Schritt 10: Selbst erklären.

"Tell me and I forget. Teach me and I remember. Involve me and I learn." - Benjamin Franklin

Sobald wir versuchen das Gelernte jemand anderem zu erklären, werden wir merken, welche Themen von denen wir dachten, wir hätten sie verstanden, wir doch noch nicht verstanden haben. Es ist die beste Möglichkeit, das Gelernte zu überprüfen und Lücken zu füllen. Wichtig ist, das Wissen in eigene Worte zu fassen und das Ganze selbst zu Strukturieren. Möglich ist das beispielsweise, indem man ein YouTube Video erstellt, sich mit einem Freund oder Mitarbeiter unterhält, eine Präsentation erstellt oder Fragen in einem Forum beantwortet. Sobald wir selbst versuchen, etwas mit unseren eigenen Worten zu erklären, ordnen wir die unterschiedlichen Informationen in unserem Gehirn, so dass sie für uns Sinn ergeben. Erst dann können wir effektiv auf das Gelernte zurückgreifen.

Diese Schritte stellen sicherlich keine „magische Formel“ dar. Wenn man merkt, dass es so formal nicht funktioniert, sollte man die Schritte anpassen oder weglassen. Die Schritte an sich sind auch nicht wichtig, wichtig ist es, das Konzept dahinter zu verstehen. Nur dann kann man ein eigenes System entwickeln, um sich selbst effizient etwas beizubringen.

Mentor

Ein Mentor oder auch Trainer kann hilfreich sein, um neue Themen zu erlernen. Doch wie erkennt man einen geeigneten Mentor? Gute Mentoren sind meistens diejenigen, die die meisten Fehler durchlaufen haben. Allerdings muss der Mentor selbst das Thema nicht unbedingt beherrschen. Tiger Woods wird von jemandem trainiert, der selbst nicht so gut spielt wie er, aber ihm fallen Aspekte auf, die Tiger Woods nicht auffallen. Man sollte sich jemanden suchen, der bereits anderen geholfen hat, das zu erreichen, was man auch selbst erreichen möchte. Einen guten Mentor erkennt man auch oft daran, wie viele Personen er beeinflusst. Letztendlich muss man natürlich auch persönlich mit der Person zureckkommen. Doch wo findet man so eine Person? Es gibt Portale, dort kann man für verschiedene Themen Mentoren bzw. Trainer mieten, doch man sollte sich eher im eigenen Umfeld umschauen. Eventuell kann ein Freund, ein Familienmitglied, ein Freund eines Freundes, ein Arbeitskollege oder eventuell auf der eigene Chef als Mentor fungieren. Doch selbst wenn man einen Mentor findet, heißt das noch lange nicht, dass er einem auch hilft. Erfolgreiche Personen sind oft beschäftigt und haben daher wenig Zeit. Eine Möglichkeit ist daher, immer etwas im Austausch anzubieten. Das kann beispielsweise schon ein Mittagessen sein, welches man für den Arbeitskollegen übernimmt. Außerdem ist es wichtig, nicht beim ersten „Nein“ aufzugeben. Man darf an dieser Stelle nicht zu nett sein und sollte wiederholt nachhaken.

Andersrum betrachtet kann und sollte man auch selbst die Rolle eines Mentors einnehmen. Im Prinzip kann das jeder tun. Jeder weiß bereits etwas, was andere versuchen zu lernen. Als Mentor muss man, wie bereits erwähnt, nicht perfekt sein. Oft hilft es dem Lernenden schon, einen anderen Blickwinkel einzunehmen oder eine zweite Meinung zu geben. Vorteil der Mentor-Rolle ist, dass man dabei i.d.R. am meisten lernt. Dazu kommt, dass die Leute, denen man hilft sich oft an einen erinnern und einem später dafür an anderer Stelle helfen. Ein großes Problem ist jedoch, dass man irgendwann nicht mehr allen helfen kann, da auch Zeit für die eigenen Aufgaben bleiben muss. Dann ist es wichtig denen zu Helfen, die wirklich Lust haben etwas zu lernen und die entsprechende Motivation mitbringen.

Lehren

Lehren ist der beste Weg, etwas zu lernen und wahrscheinlich der einzige Weg, etwas im Detail zu verstehen. Doch wir fühlen uns meist sehr unwohl, wenn wir daran denken zu lehren. Oft liegt das nicht daran, dass wir nicht lehren bzw. erklären können, sondern daran, dass wir nicht selbstbewusst genug sind. Wir möchten i.d.R. nur die Themen lehren, in denen wir selbst Experten sind. Doch um ein Experte in einem Thema zu werden, müssen wir zuerst lehren - ein Teufelskreis. Der Trick ist, viele von uns lehren, ohne es selbst zu bemerken. Lehren bedeutet nicht nur vor Gruppe von Leuten zu stehen und Themen zu erklären. Es geht vor allem darum, das Wissen zu Teilen. Wir haben alle schon einem Arbeitskollegen oder einem Kommilitonen etwas erklärt. Auch das ist Lehren. Weiter kann

es hilfreich sein, einen eigenen Blog zu starten, Präsentationen im Unternehmen durchzuführen oder Videos bzw. Screencasts zu erstellen. Um Lehrer zu sein, braucht man keine Zertifikate und keinen Abschluss und man muss auch kein Experte sein. Wir alle sind bereits Lehrer.

Wissenslücken

Wir alle haben Wissenslücken und Schwächen, doch meistens fallen uns diese Lücken garnicht auf. Eine Möglichkeit Wissenslücken zu identifizieren, ist zu überlegen, wo man am meisten Zeit investiert. Meistens gibt es tägliche, wiederkehrende Aufgaben, die durch Wissenslücken verlangsamt werden. Ein Beispiel sind die Shortcuts der IDE. Wenn diese Shortcuts, die häufig benötigt werden, nicht bekannt sind, benötigt man für einfache Aufgaben wesentlich mehr Zeit. Eine weitere Möglichkeit um Wissenslücken aufzudecken, besteht darin, aktiv auf Verständnisprobleme zu achten und diese aufzulisten. Zusätzlich sollte man notieren, wie oft welches Verständnisproblem auftritt. Nicht jede Wissenslücke, die auftritt, muss unbedingt geschlossen werden. Aber anhand der Liste lassen sich Lücken finden, die besonders oft auftreten.

In einem zweiten Schritt geht es darum, die Wissenslücken zu schließen. Der schwerste Teil, nämlich das Identifizieren der Wissenslücken, ist bereits getan. Wichtig ist herauszufinden, was man konkret lernen muss. Es ist wenig hilfreich zu wissen, dass man beispielsweise schlecht in Physik ist. Doch wenn man weiß, dass man nich versteht, wie z.B. Federn funktionieren, lässt sich diese Lücke einfach schließen. Außerdem sollte man während eines Gespräches zeitnah nachfragen, wenn man etwas nicht versteht.

4 Produktivität (Productivity)

Wenn man Produktivität definieren will, kann man es mit einfachen Worten verdeutlichen: "Mach' deine Arbeit." bzw. "Tu' es."

Trotz der Trivialität dieses Imperativsatzes fällt es vielen Menschen dennoch schwer dies auch auszuführen. Dies ist zumeist Folge von Ablenkung wie z.B. Social Media Sites oder E-Mails und Mangel an Eigendisziplin.

Zudem ist produktives Arbeiten nicht unbedingt gleich effektives Arbeiten. Man kann sehr viel produzieren, was zu von einer Produktivität zeugt, und trotzdem nicht effizient sein, denn dafür muss man das richtige machen.

In den nachfolgenden Themen werden einige Faktoren und Techniken aufgezeigt, wie man die Produktivität nachhaltig steigern kann.

Fokus

Sich auf etwas fokussieren bedeutet, dass man sich einzig und allein auf eine einzige Sache konzentriert. Entsprechend bedeutet fokussiertes Arbeiten nichts anderes als dass man eine einzige Sache mit höchster Aufmerksamkeit bearbeitet. Demnach ist Fokus sozusagen das Gegenteil von Ablenkung, da diese uns Menschen die Aufmerksamkeit nimmt. Heutzutage ist die Welt voll mit Ablenkungen wie Social Media, Online Games oder TV-Sendungen, welche das fokussierte Arbeiten immens erschweren. Trotz dieser verlockenden Ablenkungen ist Fokus ein absolutes Muss für Produktivität, da dieser die Produktivität immens steigert.

Den Zustand von Fokus kann man jedoch nicht sofort erreichen. Es erfordert etwas Überwindungskraft.

Überwindung des "Initialschmerzes"

Der Initialschmerz ist eine entscheidende Phase, welche relativ unangenehm für jede Person ist, aber die ausgehalten werden muss, damit ein Fortarbeiten wahrscheinlicher ist. In dieser Phase ist man sehr anfällig gegen externe Ablenkungen. Daher ist es ratsam, Vorkehrungen zu treffen, damit solche Ablenkung erst gar nicht zustande kommt.

Beispielsweise kann man vor Beginn seine Mails abchecken, ablenkende Internetseiten schließen, Zimmer abschließen (Sofern man ein eigenes Zimmer hat) oder Mitarbeitern signalisieren, dass man die nächste Zeit nicht gestört werden will).

Im Regelfall dauert diese Phase 5-10 Minuten an. Danach ist man auch gegen Ablenkungen resistenter.

Aufrechterhaltung des Fokus & Handhabung mit Unterbrechungen

Es kann vorkommen, dass man plötzlich eine Nachricht per E-Mail oder Chatclient bekommt oder von jemand angerufen oder angesprochen wird (z.B. Kollegen oder Familienangehörigen), und dementsprechend seinen Fokus auf die eigentliche Aufgabe verlieren kann. Danach muss man sich erneut zu seinem Fokus auf die Arbeit ringen, was Zeit und dementsprechend Produktivität kostet. Deshalb sollte man jegliche Form von Unterbrechungen vermeiden. Beispielsweise kann man seinem Umfeld vermitteln nicht gestört werden zu wollen und jegliche Form der Kommunikation zu ignorieren.

Produktivitätsplanung

Es macht Sinn sein Leben systematisch zu planen - sowohl in Quartalen als auch in Monaten, Wochen und Tagen. Tools und Techniken wie Kanban (kanbanflow) oder Trello können dabei sehr unterstützend wirken.

Quartalsplanung

Hierunter fallen Aufgaben mit langer Zeitspanne oder Großprojekte. Dabei werden kleinere Aufgaben notiert, welche wöchentlich bzw. täglich im Rahmen dieses Projekts gemacht werden müssen. Es macht daher Sinn, an dieser Stelle der Planung eine grobe Granulierung der Aufgabe/des Projekts anzusetzen, sofern nicht erfolgt.

Monatsplanung

Hier wird der grobe Arbeitsaufwand der abgeschätzt. Aufgaben, welche im Zusammenhang mit den anstehenden großen Aufgaben/Projekten aus der Quartalsplanung stehen, werden dabei bei der Planung berücksichtigt. Zudem werden Dinge geplant, welche monatlich erledigt werden müssen (z.B. Deadlines von Projekten).

Wochenplanung

Planungen von Dingen die wöchentlich erledigt werden müssen (Projektmappe, Haushalt, Garten, Meetings).

Tagesplanung

Dinge, die persönlich als wichtig erachtet werden, machen (z.B. Trainingseinheiten), um Ablenkungen zu vermeiden und fokussiertes Arbeiten zu ermöglichen (s.o. "Fokus").

Danach werden die Aufgaben, welche für den Tag anstehen evaluiert und entsprechend ihrer Wichtigkeit zeitlich angeordnet, damit die wichtigsten Aufgaben garantiert erledigt werden.

Urlaub

Es ist sinnvoll, von Zeit zu Zeit Urlaub von der Produktivitätsplanung und jeweils angewandten Produktivitätssystem zu nehmen, um dessen Wichtigkeit bzgl. der Produktivität zu realisieren. Zudem ist es gesundheitlich nicht ratsam durchgehend wie eine Maschine zu arbeiten (siehe Burnout).

Promodoro Technik

Die Promodoro-Technik wurde in dem späten 80'ern von Francesco Ciritto entwickelt und soll System in die Produktivität bringen.

Prinzipiell ist die Technik relativ einfach:

1. Zuerst erstellt man einen Arbeitsplan für den ganzen Tag.
2. Danach widmet man sich der zuerst geplanten Aufgabe. Dabei arbeitet man 25 Minuten durch und fokussiert sich nur auf die eine Aufgabe ohne sich unterbrechen zu lassen.
Dies ist ein sogenanntes Promodoro.
3. Sofern man früher die Aufgabe beenden kann, wird ein sogenanntes "Overlearning" angewandt. Das kann beispielsweise zusätzliche Optimierungen bei einem Programm oder ein nochmaliges Überfliegen der schon gelesenen Materialien beim Lernen sein.
 - i. Nach dem Promodoro nimmt man sich 5 Minuten Zeit um Pause zu machen.
Danach widmet man sich der Aufgabe wieder, sofern sie noch nicht erledigt ist, oder man beginnt mit der nächstwichtigen Aufgabe.
 - ii. Alle 4 Promodori macht man eine längere Pause (ca. 15 Minuten).

Effektives Anwenden

Viele Leute versuchen diese Technik anzuwenden, ohne ihren richtigen Potential zu erkennen, weswegen sie meistens diese nach kurzer Zeit verwerfen. Eines der Potentiale liegt darin, dass man mit dieser Technik die zu machende Arbeit messen und abschätzen kann. Das Messen der getane Arbeit in Promodori ermöglicht einen besseren Überblick, wieviel man am Tag gearbeitet und erreicht hat. Zudem wird auch das persönliche Maximum an Arbeit besser ersichtlich.

Viele Leute, die diese Technik in dieser Art angewendet haben, waren danach viel motivierter. John Z. Sonmez, der Verfasser des zusammengefassten Buches, schafft zwischen 50-55 Promodori pro Woche. Diese Zahl ist dennoch kein Richtwert und kann von Person zu Person variieren. Zudem ist er der Meinung, dass eine 40-Stundenwoche nicht automatisch 80 Promodori bedeuten muss.

Quotensystem

John Z. Sonmez benutzt zusätzlich ein Quotensystem zu der Promodoro-Technik, um seinen Produktivitätsfortschritt messbar zu gestalten. Er verspricht sich davon viel konsistenter Fortschrittsergebnisse. Prinzipiell besteht das Quotensystem aus folgenden Schritten:

1. Such dir eine Aufgabe aus.
2. Bestimme den Zeitraum, in welcher diese Aufgabe erledigt sein und wiederholt werden muss.
3. Bestimme, wie oft diese Aufgabe während dieses Zeitraums gemacht werden soll (Quote).
4. Umsetzen.
5. Quote anpassen, sofern die Quote zu tief bzw. hoch angelegt wurde.

John z. Sonmez produziert nun durch Anwendung dieses Quotensystems wesentlich mehr als zuvor und berichtet positiv von der Anwendung des Quotensystems.

Eigenverantwortung

Die Entwicklung von Eigenverantwortung ist wichtig für die Produktivität. Dabei ist Selbstbeherrschung und Selbstdisziplin der Schlüssel zur Selbstmotivation. So kann man damit anfangen, sein eigenes Leben Struktur zu verleihen, indem man sein Leben inkl. der dazugehörigen Aktivitäten plant. Dabei ist es nicht verwerflich andere Personen um Hilfe zu bitten (Ernährungs- und Sportplan von Ernährungsberatern und Fitnesstrainern als Beispiel). Auch Aktionen in der Öffentlichkeit können der Eigenverantwortung Nachdruck verleihen. Beispielsweise fallen dort fehlende Aktionen schneller auf, was zu einem Pflichtgefühl gegenüber der Öffentlichkeit mündet und somit einen Motivationsdruck zufolge haben kann.

Multitasking

Es gibt Aufgaben, die man miteinander kombinieren kann. In den meisten Fällen ist es jedoch kein wirkliches Multitasking, sondern Task-Switching, was prinzipiell das hin- und herwechseln zwischen den Aufgaben bedeutet.

Task-Switching wiederum ist für die Produktivität von Nachteil, da man immer wieder den Fokus wechseln muss (siehe Fokus). Richtiges Multitasking ist, wenn man zwei oder mehr Sachen zur gleichen Zeit macht. Ein Mann der joggt während er Musik hört, betreibt beispielsweise wirkliches Multitasking.

Dabei profitiert er durch die Musik, indem er zum Takt der Musik läuft und somit länger laufen kann. Ähnlich verhält es sich bei anderen Multitaskings auch: Zwei oder mehr Sachen gleichzeitig machen kann unter anderem die Produktivität steigern. Jedoch kann man nicht beliebige Aufgaben zum Multitasking verknüpfen. Bewährt hat sich das Kombinieren von Aufgaben, die wenig Verstand benötigen, mit Aufgaben, die mentalen Fokus benötigen.

Stapeln von Aufgaben

Aufgaben, die zwar nicht gleichzeitig erledigt werden können, können jedoch zusammengefasst und stapelweise abgearbeitet werden. Das verhindert, dass man zwischendurch den Fokus verliert, und steigert entsprechend die Produktivität.

Beispielsweise kann man einmal am Tag alle Emails abchecken und antworten, anstatt dass man simultan bei jeder eintreffenden Mail seine eigentliche Aufgabe vernachlässigt.

Burnout und dessen Lösung

Burnout ist eines der größten Probleme bzgl. Produktivität. Mit der Zeit Gewöhnt man sich an seiner Arbeit, sodass es nichts Besonderes mehr ist oder gar verabscheut wird.

Entsprechend schwindet die Motivation und Interesse. Dann kommt irgendwann der Punkt, an dem man sowohl physisch als auch psychisch erschöpft ist. Man stößt umgangssprachlich an die Mauer. Das ist meist der Punkt an dem die Produktivität gering bis gar nicht mehr auftritt und die meisten aufgeben.

Die Lösung zu diesem Problem ist durchhalten, bis man die Mauer überwunden hat. Denn nachdem man diese überwunden hat, steigt sowohl die Motivation als auch das Interesse wieder. Die Lösung liegt quasi darin, das Burnout zu ignorieren. Unter anderem muss man auch realisieren, was auf der anderen Seite der Wand wartet.

Sehr hilfreich kann auch eine Struktur im Leben sein. Das Selbstaufstellen von Regeln kann für den nötigen Antrieb im Leben sorgen.

Zeitverschwender

Zeitverschwendungen ist ein Teil des Menschen. Man kann diese nicht komplett unterdrücken. Man kann aber einige der größten Zeitverschwender unterbinden. Dabei kann das Verfolgen jeglicher Zeitnutzung in Form von Protokoll hilfreich zur Identifikation dieser sein. Tools wie „RescueTime“ wirken dabei unterstützend.

Beispielsweise ist eines der größten Zeitverschwender TV schauen. Im Durchschnitt verbringt der Mensch ca. 40 Stunden mit Fernsehen. Wenn man bedenkt, dass man 8 Stunden zum Schlafen, 8 Stunden zum Arbeiten und 2 Stunden zum Essen braucht und die restliche Zeit für die ganze Woche betrachtet, bleiben dem Menschen nur 2 freie Stunden für eine ganze Woche nach Abzug dieser Zeit. Man sollte demnach in Erwägung ziehen, das Fernsehen aufzugeben oder auf ein Minimum zu reduzieren, damit mehr Zeit für sinnvollere Tätigkeiten bleibt (z.B. Trainieren).

Aber auch Zeitverschwender wie Social Media, Nachrichten, Kaffeepausen, Videospiele oder auch unnötige Meetings kann man optimieren, indem man diese reduziert oder zu einem Zeitpunkt zusammenfasst.

Wichtigkeit von Routinen

Personen, die vielleicht nur eine Aufgabe machen, weil es sein muss, werden meist nicht so schnell diese erledigen als Personen, die es sich zur Routine gesetzt haben, diese zu erledigen. Routinen können dementsprechend die Produktivität immens erhöhen.

Daher ist es sinnvoll, Routinen zu entwickeln, die bei der Aufgabenbewältigung unterstützend wirken. Um den Überblick nicht zu verlieren, ist es auch von Vorteil, diese in Tagesform zu planen.

Gewohnheiten

Genau wie Routinen können Gewohnheiten fördernd für das Ziel sein. Jedoch gibt es gute und schlechte Gewohnheiten. Entsprechend können schlechte Gewohnheiten hinderlich für das Ziel sein (permanenter E-Mailcheck).

Gewohnheiten werden durch 3 Dinge definiert:

1. Auslöser
2. Routine
3. Belohnung

Zum Beispiel kann man den E-Mailcheck nehmen: Der Auslöser wäre die E-Mailbenachrichtigung in Form eines Pop-Ups, die Routine wäre das Abchecken der E-Mail und die Belohnung wäre die Gewissheit über den Inhalt dieser E-Mail.

Abgewöhnen von schlechten Gewohnheiten

Man kann entweder versuchen, die schlechten Gewohnheiten zu vermeiden, oder diese sogar durch andere zu ersetzen. Bei letzteren muss man sich im Klaren sein, was diese Gewohnheit auslöst, und wie man diesen Auslöser für eine andere Gewohnheit verwenden kann.

Entwicklung von neuen Gewohnheiten

Die Entwicklung von neuen Gewohnheiten kann die Produktivität immens fördern, da diese eine unterstützende Rolle für Schlüsselroutinen für die zu bearbeitende Aufgabe haben kann. Denn das Planen von Routinen bedeutet nicht unbedingt, dass diese erfolgreich umgesetzt werden.

Aufgaben granulieren

Große Projekte können sehr einschüchternd wirken, da viele Leute nicht abschätzen können, wie groß der Arbeitsaufwand und wie viel Zeit das Projekt in Anspruch nehmen kann.

Das Herunterbrechen in mehrere kleine Aufgaben kann dabei helfen, Zeit- und Arbeitsaufwand besser zu berechnen.

Wert harter Arbeit: Wieso vermeiden wir sie?

Der Mensch hat die Tendenz, harte Arbeit, die ihm wichtig erscheint, zu vermeiden oder bei dessen Umsetzung zu zögern. Selbst bei der Umsetzung fühlt er sich meist nicht wohl dabei oder die Arbeit ist ihm zumeist zu langweilig.

Dennoch muss man lernen, sich hinzusetzen und die Arbeit zu machen, die man nicht gern macht. Die Entscheidung liegt dabei nur bei ihm, ob etwas gemacht wird oder nicht. Es hängt dementsprechend von der eigenen Willenskraft ab.

Jedes Tun ist besser als nix zu tun

Nichtstun ist der schlimmste Produktivitätskiller. Die meisten Gelegenheiten und Möglichkeiten werden dadurch verschwendet, indem man nichts tut. Angst vor einer falschen Entscheidung spielt dabei eine sehr große Rolle.

7 Spirit

Im Kapitel Spirit geht es um den Geist des Menschen und wie er mit seinem Körper verbunden ist. Er teilt das Kapitel in verschiedene Unterkapitel ein. Die Unterkapitel befassen sich mit Fähigkeiten, die man im Leben braucht. So beschäftigt er sich mit dem Verstand, der mentalen Einstellung, positiven Denken, Liebe und Ausdauer.

How the mind influences the body

Der Autor erklärt, dass wir nicht einfache Maschinen sind, sondern Menschen. Der Geist gibt uns die Möglichkeit Erfolg zu haben oder uns scheitern zu lassen. (S. 396)

Er beschreibt, dass der Geist einen machtvollen Einfluss auf den Körper hat und das wir lernen müssen ihn zu beherrschen, wenn wir sogar den kleinsten Plan in die Tat umsetzen wollen.

Dies ist laut dem Autor nicht einfach. Man kann zum Beispiel glauben, dass Elefanten pink sind. Allerdings wird selbst der überzeugendste Beweis nicht verhindern, dass man an das glaubt, was man gewohnt ist. Das heißt natürlich nicht, dass man sich davon nicht überzeugen ließe.

Er behauptet, dass wir Opfer eines biologischen Prozesses unseres Kopfes wären.

Allerdings sind wir keine Tiere, sondern Menschen. Das heißt, wir haben die Möglichkeit den biologischen Prozess in unsere Richtung zu lenken. (S. 397-398)

Weiterhin stellt er die Frage, ob die physische Welt sich verändert, um die eigene Wahrnehmung zu erfüllen.

Die Antwort beantwortet die Frage damit, dass der Glaube die Macht hat die eigene Realität zu formen. Es ist ein indirektes Formen, welches dazu den eigenen Körper benötigt. So besitzen wir alle die Möglichkeit die physische Welt direkt zu manipulieren, sobald wir das erste Neuron feuern.

Er widerspricht der Meinung einiger Leute, die behaupten, dass wir eine Zusammensetzung von chemischen Elementen sind, die unsere Umwelt beeinflussen, eine Art lebenslanger Autopilot, eine Kette von chemischen Reaktionen, die auf unsere Umgebung basiert. Er wirft dabei folgende Fragen auf: Wie ist es möglich, dass wir sein Buch lesen können? Wie ist es möglich, dass er es schreiben kann? Seine Antwort ist die Freiheit der Wahl, der sogenannte "freie Wille".(S. 399)

Der Autor definiert den Verstand als nicht physischen Teil des Körpers abgegrenzt zum unteren Teil des Körpers und dem Gehirn. Er meint, dass es möglich ist durch Drogen unseren Verstand zu verändern. Dadurch beeinflusst unser Verstand unseren Körper in Richtungen, die wir nicht kontrollieren können.

Nach seiner Meinung gibt es eine Menge Formen und Philosophien. Die Populärste ist, dass negative Gedanken zu negativen Ergebnissen führen und umgekehrt genauso. Er behauptet von sich, dass er ein praktischer Mensch ist, aber er das Vorhandensein einer mystischen Komponente nicht ausschließen würde. (S. 400)

Abschließend meint er, dass unser Bewusstsein und unser Glaube einen positiven oder negativen Einfluss auf unser Leben haben können. Er will uns in seinem Text praktische Beispiele zeigen, wie wir unser Verstand so schärfen können, dass wir die höchste Produktivität für uns heraus holen. (S. 401)

Having the right mental attitude: Rebooting

Im nächsten Unterkapitel wird beschrieben, dass positives Denken ermöglicht, die eigene Lebenszeit zu verlängern, Freundschaften zu entwickeln, ein höheres Einkommen zu haben und körperlich gesund zu sein. Er widerspricht der Behauptung, dass positives Denken destruktiv ist und man realistisch bleiben sollte. Für ihn ist positives Denken die ultimative Form von Realismus, weil der eigene Glaube die Möglichkeit hat, seine Realität zu verändern, so dass man kein Opfer seiner Umstände wird. Er meint mit einer positiven Einstellung lebt man nicht in einer Fantasiewelt, abgetrennt von der Realität, sondern in einer optimalen Welt, wo jeder seine best mögliche Zukunft sieht, und diese real werden lassen will.

Er meint, dass eine Person mit einer positiven Einstellung dazu tendiert, mehr Situationen gut statt schlecht zu betrachten und zwar nicht weil die Situationen objektiv gut oder schlecht sind, sondern weil er erkennt, dass er selbst wählen kann, wie er sie betrachten möchte. (S. 401-403).

Der Autor weiß, dass seine Einstellung seine Performanz beim Arbeiten beeinflusst. Er erkennt dies an seiner eigenen Produktivität. Wenn er eine positive Einstellung hat, ist es für ihn leichter mit Hindernissen umzugehen und Herausforderungen zu akzeptieren, als wenn er mit einer negativen Einstellung umgeben ist.

Er sagt aber auch, dass es nicht reicht, positiv sein zu wollen. Es ist nicht leicht seine Sicht auf die Welt von einer negativen in eine positive zu ändern. (S.405)

Er ist der Meinung, wenn man seine Einstellung ändern will, muss man erstmal seine Gedanken ändern. Und wenn man seine Gedanken ändern will, muss man seinen Ansatz zu Denken ändern. Der Ansatz zu Denken erfolgt durch die eigenen Gewohnheiten und dies führt dazu, dass man manche Wege im Leben signifikant ändern bzw. eine andere Gewohnheit entwickeln sollte. Er sagt, dass es schwierig ist über jedes Ereignis positiv zu denken, aber man selbst die Macht hat positive Gedanken zu erschaffen. Der Schlüssel dafür ist dies aktiv und ernsthaft durch den Tag zu versuchen, sich selbst daran zu erinnern, dass man seine sofortige Reaktion auf eine Situation nicht kontrollieren kann, sondern man kontrollieren sollte, wie man über die entstandene Erfahrung zu denken wählt.

Je mehr man diese Art von Denken übt, desto mehr positive Bilder entwickelt man und lässt dies zur Gewohnheit werden. Nach einiger Zeit ist man eher bereit, einen Unfall oder ein Missgeschick in einer positiven Art und Weise zu beantworten. Man kann sein Gehirn trainieren, die Dinge aus einer positiven Perspektive statt aus einer negativen zu betrachten. Weiterhin erwähnt er, dass manche Studien zeigen, dass Leute, die meditieren, eher bereit sind positive Emotionen zu empfinden. Also sollte man zu meditieren versuchen, um sein positives Mojo wachsen zu lassen. Auch findet er, dass es einfacher ist positiv zu sein, wenn man ab und zu Spaß hat. (S. 406)

Zusammenfassend meint er, dass positives Denken nicht als Chance kommt und nicht etwas ist, was man über Nacht erzwingen kann. Es braucht eine gewisse Anstrengung seinen Verstand in eine positive Richtung zu lenken. Aber es lohnt sich. Man genießt nicht nur das Leben stärker, sondern ermöglicht auch den umgebenden Anderen das Leben zu genießen. (S.407)

Building a positive self-image: Programming your brain

Als nächstes spricht er davon, dass man lernen muss sein Gehirn so zu programmieren, dass man seine Ziele erreichen kann. Die wahre Schlacht richtet sich Mittelmäßigkeit und beginnt im Gehirn. Was man über sich selbst denkt hat die Macht einen zu limitieren oder voranzubringen. Er will zeigen auf welche Weise man ein positives Selbstbild entwickelt, das uns erlaubt im Gehirn ein Autopilot zu entwickeln, um seine Ziele zu erreichen. (S.408)

Seiner Meinung nach ist das Selbstbild die Sicht, die man über sich entwickelt, ohne die Dinge die Andere über einem sagen. Es ist möglich sich seines wirklichen Selbstbildes nicht bewusst zu sein, weil dies sehr weit im eigenen Unterbewusstsein verankert ist. Tief innen haben wir alle ein Bild von uns, welches die ultimative Reflektion der Sicht unseres Gehirns auf unsere Wahrnehmung ist. Dieses Selbstbild ist machtvoll, weil unser Gehirn uns nicht erlaubt, etwas zu tun, was gegen unsere eigene Beurteilung ist. Es ist auch schwierig sich dazu zu überwinden, einfach weil man sich nicht bewusst ist, dass diese Grenzen existieren. Der Autor unterscheidet zwischen Dingen, die in unserer DNA verankert sind, unsere physischen Charakterzüge, und den Dingen, die wir in uns selbst manifestiert haben, so könnte man zum Beispiel faul, nicht gut in Mathe, schlecht im Umgang mit Leuten, schüchtern, reserviert oder aufmerksamkeitssuchend sein.

Er meint auch, wenn einem in der Kindheit eine Charaktereigenschaft nachgesagt wurde, auch wenn sie bis dahin nicht zutreffend war - aber in dem Moment der Erwähnung verinnerlicht wurde. (S.408-409)

Er beschreibt, dass wir die Macht haben unser Selbstbild zu ändern. Das Konzept dahinter ist das ständige Vortäuschen einer gewünschten Eigenschaft bis man sie verinnerlicht. Er sagt, dass es ein einfaches Konzept ist und das wir nur glauben, dass es hart ist. Wir

hätten ein Teil in uns selbst, das krank und sadistisch ist und unsere Schwächen und Grenzen als kritischen Teil ins uns hervorhebt. Er sagt auch, dass unser Unterbewusstsein an unserem Selbstbild festhält und dass wir selbst den Willen haben müssen, unser Selbstbild zu ändern.

Er erzählt auch, dass die Kleidung, die wir tragen, einen Teil unseres Charakters ausmacht. Es fällt ihm auf, dass wir gerne nach der Art unserer Kleidung handeln und dies auch mit in unser Selbstbild einfließen lassen.

Er beschreibt wie er selbst von einer schüchternen und unathletischen Person zu einer athletischen und sozialen Person wurde, indem er die Kontrolle über sein Selbstbild übernommen hat, um so dafür zu sorgen, dass es in seiner Vorstellung für ihn arbeitet statt gegen ihn. (S. 409-411)

Im weiteren Kontext schildert er wie man sein Gehirn darauf trainiert sich Ziele zu setzen und vorzutäuschen, welche Person man sein möchte, um diese schließlich zu werden. Man soll seine Aufmerksamkeit auf sich selbst richten und nicht auf das was andere Leute von einem wollen. Man soll positive Bestätigungen in seinem Alltag suchen und mental an sich selbst glauben. Man soll aufpassen, was man sagt, weil das Unterbewusstsein immer noch ein kleines Kind ist, das auf die eigene Stimme hört. (S.412-413)

Love and relationships: Computers can't hold your hand

Im nächsten Unterkapitel schreibt der Autor über Liebe und Beziehungen. Er beschreibt ein typisches Stereotyp von Softwareentwickler, die nerdig, allein und einsam sind. Er behauptet, dass Liebe und Beziehungen komisch sind. Er sieht die Liebe als Katz und Maus Spiel. Es gibt eine Person, die jagt und eine andere, die gejagt wird. Dies ist kein Problem, solange es abwechselnd und nicht einseitig ist. Weiterhin erwähnt er, dass wenn jemand verbissen versucht eine Beziehung zu finden, er in Verzweiflung versinkt und es hart ist, aus dieser herauszukommen. Auch neigen Leute dazu ihre Gefühle der Verzweiflung und der Einsamkeit dem Rest der Welt über soziale Medien mitzuteilen. Dies macht sie aber laut dem Autor unattraktiv und andere fangen an sie zu meiden. (S.414-415)

Er kann den Gedanken verstehen, dass man ehrlich über sich selbst und seine Gefühle sein will, er aber fragt sich, ob dies auch funktioniert. Er meint, dass man realisieren sollte, dass man ein Spiel spielt und eine Spielstrategie zum Gewinnen finden sollte. Er begründet dies damit, dass viele Leute nur das haben wollen, was sie nicht haben können und nicht das was leicht verfügbar ist. Das heißt je einfacher man zu haben ist, desto weniger wird man gewollt. Er sagt, dass das Leben ein großer Spielplatz ist - je mehr man jemanden jagt, desto mehr rennt er weg.

Auch hält er es nicht für eine gute Strategie zuhause rumzusitzen und auf die große Liebe zu warten. Seiner Meinung nach ist das beste sich so zu verhalten, dass man auf eine

Person selbstsicher zugeht, ihr zeigt, dass man sich gut fühlt, dass man niemanden braucht um glücklich zu sein, aber dennoch Interesse an dieser Person hat.

Nach seiner Meinung soll man erkennen lassen, dass es ein Vorteil für die Person ist, wenn sie mit einem zusammenkommt. Aber man sollte auch nicht sich selbst als Geschenk Gottes ausgeben. Man soll genug Respekt für sich selbst zeigen und dort auftauchen wo man gewollt ist und sich nur mit Leuten abgeben, die einen mögen. Das heißt nicht, dass Erfolg garantiert ist, aber man hat eine bessere Chance seine Liebe zu finden, wenn man die Psychologie des Wegrennen und Jagens versteht und anwendet. Das gleiche gilt für ihn auch bei einem Jobinterview. (S.415-417)

Er hält es für ein Nummernspiel. Es gibt laut ihm viele Arten von Leuten, mit vielen verschiedenen Vorlieben. Er sagt, dass es viele potentielle Partner für einen geben wird. Man soll auf eine Person zugehen, die mit einem zusammen sein möchte und nicht auf eine, welche es nicht will. Es gibt auch genug andere. (S.417-418)

Facing failure head-on

Im letzten Unterkapitel schreibt er über die seiner Meinung nach wichtigste Fähigkeit: Die Ausdauer. Er hält alle anderen Fähigkeiten für wertlos, wenn es an der Ausdauer scheitert. Software zu entwickeln ist schwierig und so ist es laut ihm wichtig Ausdauer zu haben.

Er sagt, dass die Angst vor Fehlern ein innerer Instinkt vieler Leute ist. Diese Angst ist gut, wenn unser Leben bedroht ist, aber schlecht, wenn ein Fehler nur harmlos ist. Wir versuchen das zu tun, was wir können und vermeiden Dinge für die wir inkompetent sind oder nicht die entsprechende Fähigkeit zu haben. Er hält es für einen Schutz des eigenen fragilen Ego. Er glaubt, dass wir denken, dass dies eine Reflektion unseres persönlichen Versagens ist. Der Autor meint, dass das Verletzen des eigenen Egos ein Missverständnis der Natur eines Fehlers ist, weil wir nicht dazu trainiert sind Fehler als Weg, in vielen Fällen als einzigen Weg, zum Erfolg zu sehen. (S.424-425)

Er beschreibt, dass Fehler nicht dasselbe wie Niederlagen sind. Laut ihm sind Fehler temporär, Niederlagen sind permanent. Eine Niederlage ist etwas was man wählt, wenn man einen Fehler permanent akzeptiert. Er hält das Leben für schwierig, man wird niedergeschlagen, aber es ist an einem selbst zu entscheiden, ob man liegen bleibt oder wieder aufsteht. Es liegt an einem selbst die Freude und das Vergnügen zu realisieren, welche zum Großteil dann kommt wenn man es trotz aller Schwierigkeiten und Kämpfe geschafft hat. (S.426)

Statt Fehler zu fürchten soll man sie akzeptieren, sie erwarten und bereit sein ihnen zu begegnen. Es ist für ihn ein notwendiger Schritt um erfolgreich zu sein. Nur wenige Dinge im Leben werden ohne Fehler getan.

Er sieht es als Problem, dass wir lernen Fehler im negativen Licht zu sehen. Als Beispiel nennt er einen nicht bestandenen Test in der Schule den man nicht als Lernfortschritt sieht,

um seine Ziele zu erreichen. Stattdessen sieht man das Ganze negativ.

Er will damit sagen, dass ein Fehlschlag im echten Leben für gewöhnlich ein notwendiger Meilenstein ist, welcher uns immer näher an den eventuellen Erfolg bringt. Mit einem Fehlschlag lernt man dazu und wächst an der Erfahrung. Laut ihm ist unser Gehirn darauf trainiert auf solche Weise zu arbeiten. Das Gehirn macht über die Zeit geringe Korrekturen bei sich wiederholenden Fehlern, die man erlebt, ohne sich dessen bewusst zu sein. Alles was man laut ihm versuchen muss, ist es weiter zu machen und keine Angst vor Fehlern zu haben. (S.426-427)

Weiterhin sagt er, dass es nicht reicht die Angst vor Fehlern zu verlieren, sondern dass man Fehler suchen soll. Man soll sich selbst in Situationen bringen, wo es naheliegt, dass man fehlschlägt. Er behauptet, dass wir oft stagnieren und aufhören Dinge zu tun, die für uns gefährlich oder herausfordernd sind. Wir suchen uns einen komfortablen Platz in unserem Leben.

Manchmal sollte man gewillt sein in eine unkomfortable Situation zu geraten, welche einem zwingt daran zu wachsen. Manchmal soll man versuchen diese Situationen herbeizuführen, weil je mehr man fehlschlägt, die anschließenden Erfolge umso größer erscheinen. Man soll das Akzeptieren von Fehlern zum Teil seines Lebens machen und hinnehmen, dass manchmal Fehler unvermeidlich sind. Man kann seiner Meinung nach nicht alles beim ersten mal perfekt machen, Fehler sind für ihn vorprogrammiert. Wenn man dies akzeptiert hat, dann hört man auf sie zu fürchten. (S.427-428)

Trusted Web 4.0 - Konzepte einer digitalen Gesellschaft

Zusammenfassung des Buches:

Titel: Trusted Web 4.0 - Konzepte einer digitalen Gesellschaft - Konzepte der Dezentralisierung und Anonymisierung

Verfasser: Olaf Berberich

Verlag: Springer-Verlag

Jahr: 2016

ISBN: 978-3-662-49189-8

Zusammenfassung von: Benjamin Schmidt

2 Rechtliche und organisatorische Grundlagen

Mit der Trusted Web 4.0 müssen nur einige der bestehenden Gesetze verändert werden, da der Bürger viele Aufgaben anonym und dezentral erledigt. Die Vorratsdatenspeicherung erfolgt in einem Stufenmodell. Durch dieses Stufenmodell ist der Bürger grundsätzlich anonym, da aus den dezentral gespeicherten Daten kaum Rückschlüsse auf einen bestimmten Bürger vorhanden sind. Wenn der Bürger aber eine Straftat begangen hat, dann kann über eine nachweisliche Notwendigkeit die Daten ermittelt werden. Dieses Verfahren ist vereinbart mit dem Datenschutz und der Strafverfolgung. Damit Daten von außen nicht manipuliert werden und man einzelne Bürger abhört, werden die Daten in dem Trusted Web 4.0 dezentral zweckbestimmt gespeichert. Alle Daten des Trusted Web 4.0 werden in 1000 verschiedenen Kategorien abgelegt. Diese Kategorien werden nach Allgemeinen Oberbegriffe angegeben. Die Begriffe ergeben sich aus dem Suchverhalten der Bürger. Jeder Kategorie kann eine Aufgabe und eine Straftat zugeordnet werden. Jede Kategorie hat für jeden einzelnen Bürger eine feste IP-Adresse. Eine Straftat findet in einer Kategorie, in einer Region, in einem Sprachraum und zu einer bestimmten Zeit statt. Ein Richter kann die passenden Daten für die Strafverfolgung über die Kategorie, die Zeit und die IP-Adresse anfordern, da die IP-Adresse zu einer bestimmten Region und zu einem bestimmten Sprachraum gehört.

Gerade bei Big Data gibt es viele gesammelte Daten bei großen Unternehmen. Wenn man Daten nur noch Kategorien zuordnet, dann wird der gläserne Kunde dadurch verhindert. Das erhöht die Datensicherheit deutlich. Sollte sich aus den gespeicherten Daten einer Kategorie eine Straftat ergeben, so können die personenbezogenen Daten rechtlich angefordert werden. Den digitalen Schlüssel für diese personenbezogenen Daten hat nur der Bürger selbst, sonst niemand anderes. Daraus folgt, dass die personenbezogenen Daten nicht gespeichert werden, sondern der Bürger über seinen digitalen Schlüssel identifiziert wird. Dieser Vorgang führt zu einer Anonymisierung des Bürgers und seiner Daten. Die Daten werden auch dezentral gespeichert. Die Speicherung der Daten findet an unterschiedlichen Orten statt. Diese Schritte führen zu einer besseren Sicherheit gegenüber Datenmissbrauch.

Eine Strafverfolgung findet nur innerhalb eines Rechtsraums statt und nicht außerhalb des Rechtsraums. Der Bürger bleibt nach außen unentdeckt. Um das zu erreichen muss ein weltweiter Kategorienstandard etabliert werden. Für jede Kategorie sollte ein anderer Speicherort, Sicherheitsstandard und Authentifizierung verwendet werden. Die Trusted Web 4.0 nutzt dafür eine eigene Domain. Durch die Benutzung einer eigenen Domain kann bei

einer richterlichen Anordnung der Benutzer mit Verlaufsprotokollen verfolgt werden und auch sein Rechner kann geortet werden. Alle anderen Benutzer sind anonym unterwegs. Optimal wäre, wenn man für jede Kategorie einen Name-Server benutzt.

Das Urheberrecht schützt ein Werk, welches von einem kreativen Menschen geschaffen wurde. Viele Werke werden heute in der digitalen Welt einfach kopiert und weiterverbreitet. Die Reichweite der Werke können Eigentümer kaum noch verfolgen. Die Werke können heute kaum noch geschützt werden davor. Es gibt Möglichkeiten das geistige Eigentum in der digitalen Welt zu schützen, wenn man Dezentralisierung und Anonymisierung benutzt. Der Eigentümer sollte für seine Daten einen eigenen individuellen Schlüssel haben und er sollte Tools haben, die ihn ermöglichen zu entscheiden, wer seine Werke ansehen darf. Von diesen Werken dürfen keine Kopien gemacht werden, dass muss technisch verboten sein.

3 Konzepte der Zukunft

Ein persönliches digitales System (PDS) ist ein System, welches den Benutzer im Alltag unterstützt. Es speichert die Daten des Benutzers und kann mit vielen anderen Diensten Kommunizieren. Denn die Daten sind für Unternehmen sehr wichtig. So sind die rechtlichen Hürden des Urheberrechts und des Datenschutzes leichter zunehmen und der Nutzer bewegt sich im System mit seinen Daten anonym. Aber die Unternehmen können mit diesen Daten wiederum neue Geschäftsmodelle entwickeln. Denn um Big Data zu benutzen, muss das Unternehmen den Kunden nicht kennen. Das Unternehmen muss nur wissen, wie oft wird ein Produkt gekauft und was die Kunden noch dazu kaufen. Mit einer PDS lässt sich E-Government auch schneller beschleunigen. Die digitale Transformation lässt sich mit diesen Konzepten umsetzen.

Derzeit scheitert E-Health an den Datenschutz und der Datensicherheit. Mit einer PDS hätte man diese rechtlichen Hürden nicht, da die Daten anonym vorhanden sind. Die Gesundheitsdaten würde der Benutzer selber speichern und würde den passenden Arzt, die gewünschten Daten zur Verfügung stellen. Wenn der Benutzer den Arzt wechselt, so kann er seine Daten zum nächsten Arzt mitnehmen. Man würde die Daten der Kategorie „Gesundheit“ zuordnen abhängig von der Region. Der Benutzer wäre Eigentümer seiner Daten.

Das Auto ist ein Hilfsmittel für die Mobilität des Menschen und daher im Alltag unverzichtbar. Das muss auch so bleiben, der Benutzer benutzt das Auto als zusätzliche Unterstützung im Alltag und darf damit nicht überwacht werden. Beim Auto gibt es einen Trend zur Digitalisierung, da die Menschen sicherer unterwegs sein wollen. Technik hilft diese Sicherheit zu schaffen. Dies ermöglicht aber auch Angreifern von außen das Auto zu manipulieren, um das Lenkrad oder die Bremse zu aktivieren und fernzusteuern. Die Daten sollten anonym, dezentral gespeichert und standardisiert werden. Man kann auch hier wieder Kategorien definieren und diesen Daten den Kategorien zuordnen. Dadurch ergeben sich auch neue Geschäftsmodelle. Das Auto kann mit der PDS verbunden werden. Die Daten vom Auto kann der Nutzer dann selber kontrollieren. Selbstfahrende Autos ermöglichen in der Zukunft neue Geschäftsmodelle und neue Dienstleistungen, wenn die rechtlichen und gesellschaftlichen Probleme beseitigt sind. Die Verkehrssysteme müssen sicher sein. Dazu müssen alle Teilnehmer im Straßenverkehr mit dem Auto vernetzt sein, der Mensch muss immer in das Geschehen des Autos eingreifen können und alle menschlichen Verkehrsteilnehmer müssen anonym untereinander kommunizieren. Wichtig ist auch, dass die Daten nicht zentral gespeichert werden, sondern nur die

Verkehrsteilnehmer die Daten bekommen. Dafür braucht es ein intelligentes autonomes System, welches mit der Umwelt vernetzt ist und auf die individuellen Bedürfnisse der Individuen eingeht.

Die Trusted Web 4.0 kann auch bei Smart Home angewendet werden. Statt die Daten zentral zu speichern werden die Daten dezentral gespeichert. Das Gebäude optimiert sich autonom selbst und muss den Versorger nicht ständig über die aktuelle Situation benachrichtigen. Angriffe von außen sind nicht machbar, da die Geräte keinen Zugriff von außen ermöglichen. Das Ziel ist eine dezentrale Energieversorgung. Die Energiekonzerne müssen in der Zukunft darauf umrüsten. Die Automobilindustrie geht diesen Schritt bereits jetzt schon. Ein Homebot verknüpft mit der PDS kann beim Energiekonzern, die benötigte Menge an Energie nachfragen und speichert die Daten beim Kunden dann vor Ort ab. So besitzt nur der Kunde seine Daten. Damit der Homebot eine Kommunikation zum Energiekonzern aufbauen kann, muss dieser erst den Kunden fragen, ob er das darf. Das führt zu einer besseren Sicherheit des Systems.

Das E-Government kann dezentralisiert und anonym mit der PDS durchgeführt werden. Die Daten werden derzeit zentral an einer Stelle gespeichert. Angreifer und Geheimdienste können sich diese Daten beschaffen, wenn sie diese zentralen Stellen angreifen. Wenn die Kommunen die Daten dezentralisiert speichern, dann ist es für Angreifer schwieriger an die Daten zukommen. Auch die Public-Key-Infrastruktur beruht auf einer zentralen Stelle, dass können sich Angreifer zunutze machen. Hier müsste man den Schlüssel auch dezentralisiert speichern. Die Schlüssel werden beim Personalausweis verwendet und die zentrale Stelle ist das BSI.

Die Industrie 4.0 kann die dezentralisierende Globalisierung einführen, also weg von der zentralisierenden Globalisierung. Die Industrie 4.0 ermöglicht individuelle Kundenanfertigungen. Durch Dezentralisierung können Produktionen wieder in Deutschland stattfinden. Man würde näher an die Geschäfte produzieren und auch näher an den Endkunden sein. Dafür müssen widerspruchsfreie und offene Standards her. Über diese Standards kann der Informationsaustausch stattfinden. Ein Standard wäre das Internet nach Kategorien und Regionen aufzuteilen, also das Internet zu dezentralisieren. Die Latenzzeiten wären dann dadurch bei der Kommunikation sehr gering. Die Industrie 4.0 muss aus intelligenten autonomen Systemen bestehen, die dezentralisiert sind. Die Kommunikation muss nach menschlichen Regeln funktionieren. Man kann eine anonymisierte Kommunikation über eine eindeutige Auftragsnummer machen. Diese Auftragsnummer ist dezentral und weltweit nur einmal verfügbar. Über diese Auftragsnummer können alle weiteren Aktionen eines Bestellvorgangs abgewickelt werden.

Der Logistiker 4.0 kann die Aufträge bündeln und dadurch den Einzelhändler ersetzen. Er kann eine Preispolitik anbieten abhängig von den Lieferzeiten. Wenn jemand seine Ware schnell will, dann muss er zusätzlich drauf zahlen. Wenn jemand bereit ist für seine Ware

länger zuwarten, dann zahlt er für seine Ware weniger. Der Logistiker kann für mehrere Branchen ausliefern, wenn die Logistikanforderungen für die Branchen gleich sind.

Bei der Dezentralisierung des Finanzwesens ist die Projektfinanzierung sehr wichtig. Bis jetzt investieren Investoren in junge und große Firmen. Die Firmen haben meist ein skalierbares Geschäftsmodell. Es gibt aber auch neu Ansätze wie das Crowdfunding. Beim Crowdfunding investieren viele Kleinanleger für ein Projektziel. Es gibt dezentralisierte Währungen wie Bitcoin, diese beruht auf Blockchaining. Wenn man das Blockchaining mit dem PDS kombiniert, dann können Transaktionen anonym durchgeführt werden.

In der Zukunft sind für die meisten Unternehmen Datensicherheit durch Dezentralisierung und die digitale Transformation wichtige Punkte. Das Ziel muss es sein eine dezentralisierte und anonymisierte IT zu etablieren. Dadurch sind Angriffe von außen für Angreifer unwichtig, da der Aufwand sehr groß ist. Der Zeit nutzt man für die Sicherheit Überwachungssysteme. Der Schaden für Cyberkriminalität ist sehr hoch und kann Unternehmen in die Insolvenz führen. Für die Unternehmen gibt es 3 verschiedene Wege. Der erste Weg ist für die Unternehmen die mit digitaler Transformation nichts zu tun haben wollen. Sie verschlafen die Entwicklung und merken durch Cyberangriffe von Seiten der Konkurrenz, dass sie den Trend verschlafen haben. Irgendwann durch massiven Stellenabbau und Verlusten geht das Unternehmen dann in die Insolvenz. Den anderen Weg gehen Unternehmen, die mit der digitalen Transformation wachsen. Sie kaufen zusätzliches Know-how am Markt ein. Auch hier kommen irgendwann Cyberangriffe und dann steht das Unternehmen genauso dar wie beim ersten Weg. Den letzten Weg gehen Unternehmen, den Weg der Dezentralisierung und der Anonymisierung. Das Unternehmen wächst hier langsamer als beim zweiten Weg, man ist aber gegen Cyberangriffe von außen sicher.

Für ein sicheres Internet müssen die Politik und die Wirtschaft handeln. Die IT-Sicherheitsbranche ist noch eng mit den Geheimdiensten vernetzt. Um ein sicheres Internet zu schaffen, müssen diese voneinander abgekapselt werden. Derzeit gibt es immer noch zentrale Portale, die massiv überwacht werden, damit sie vor Angriffen sicher sind. Derzeit setzt sich die GISAD für ein sicheres Internet ein. Für die Einführung einer PDS braucht es noch Zeit, technisch ist es aber möglich. Wichtig ist, dass die personenbezogenen Daten beim Nutzer bleiben. Der Nutzer muss anonymisiert im Netz unterwegs sein und muss sich mit seinem PDS als einzige Identifikations- und Authentifikationsmethode anmelden. Der Kopierschutz muss kopieren im Internet verhindern und der Zugriff auf Dateien muss erst gestattet sein. Smartphones Apps dürfen keinen Zugriff mehr auf personenbezogenen Daten bekommen. Die Hardwarefunktionen müssen vom Kern getrennt werden, damit Angreifer diese nicht mehr nutzen können. Es müssen dezentralisierte symmetrische Schlüssel erzeugt und gespeichert werden. Davon werden drei gleiche Schlüssel generiert

und an verschiedene Punkte gespeichert. Dezentralisierung kann auch in der Cloud benutzt werden, aber personenbezogenen Daten und Schlüssel haben in der Cloud nichts verloren. Diese Daten müssen bei dem Nutzer selber bleiben.

Die Gemeinwohl-Ökonomie

Zusammenfassung des Buches:

Titel: Die Gemeinwohl-Ökonomie - Eine demokratische Alternative wächst

Verfasser: Christian Felber

Verlag: Deuticke

Jahr: 2014

ISBN: 978-3-552-06291-7

Zusammenfassung von: Oliver Nagel, Jonathan Jansen und Sven Schirmer

1 Kurzanalyse

Werte bilden die Grundorientierung unseres Lebens. Sie bilden einen Leitstern in unserem Leben und prägen unsere zwischenmenschlichen Beziehungen. Zu diesen Werten gehören z.B.

Vertrauensbildung, Ehrlichkeit, Wertschätzung, Respekt, Kooperation, gegenseitige Hilfe und Teilen. Auch die Wirtschaft besitzt Werte, die jedoch bei genauerer Betrachtung konträr zu den zuvor genannten Werten sind. In der freien Marktwirtschaft gelten Prinzipien wie Gewinnstreben und Konkurrenz. Diese Prinzipien fördern jedoch Egoismus, Gier, Geiz, Neid, Rücksichtlosigkeit.

Die freie Marktwirtschaft bildet also einen weiteren Leitstern in unserem Leben. Da wir an den Werten unser Handeln orientieren, haben diese Werte fatale Folgen auf unsere Gesellschaft.

Die Gesellschaft kommt in einen Konflikt, da die Leitsterne in gegensätzliche Richtungen zeigen. Soll sich die Gesellschaft solidarisch und kooperativ verhalten oder die eigenen Vorteile vorzugsweise im Blick haben? Die Werte der Marktwirtschaft werden jedoch durch die Legislative in Form von Gesetzen und Regulieren oder durch Abkommen zwischen den Nationalstaaten weiter unterstützt.

Das Gemeinwohl in der Wirtschaft sollte durch Konkurrenz und durch die persönliche Gewinnmaximierung entstehen. Der Nationalökonom Adam Smith begründete dies vor 250 Jahren wörtlich mit:

„Nicht vom Wohlwollen des Metzgers, Bäckers, Brauers erwarten wir unsere tägliche Mahlzeit, sondern davon, dass sie ihre eigenen Interessen wahrnehmen.“ (Deuticke 2014, S. 19) Jedoch waren Unternehmen

vor 250 Jahren überwiegend klein, besaßen weniger Macht und agierten primär lokal. Oft waren die Unternehmer Gründer oder Eigentümer und bildeten mit Arbeitnehmer eine Personalunion.

In der heutigen Zeit sind jedoch immer mehr anonyme, global agierende Unternehmen zu finden. Durch die globale Aktivität erfahren diese Unternehmen mehr Konkurrenz. Die Unternehmen stehen

dadurch stärker im Wettbewerb hinsichtlich der Preisgestaltung und der Qualität ihrer Produkte. Die Konkurrenz sorgt auf der einen Seite für stärkere Leistungsanreize, jedoch hat sie

Auswirkungen auf die zwischenmenschlichen Beziehungen. Das oberste Ziel ist den eigenen Vorteil anzustreben und gegeneinander zu agieren. Das Übervorteilen wird so zur Normalität. Obwohl

die Würde der höchste aller Werte und im Grundgesetz verankert ist, sorgt die Konkurrenz dafür, dass wir Menschen nicht gleichwertig behandeln. Der Begriff Würde steht für den

"gleichen, bedingungslosen, unveräußerlichen Wert aller Menschen" (Deuticke 2014, S. 21). Daraus resultiert die Gleichheit aller Menschen. In der freien Marktwirtschaft ist jedoch üblich andere Menschen zu instrumentalisieren und übervorteilen und somit die Würde des Einzelnen zu verletzen. Wenn der eigene Vorteil unser höchstes Ziel ist, werden wir zwangsläufig Mittel für unsere Zwecke benutzen und andere übervorteilen.

Die freie Marktwirtschaft schränkt die Freiheit der Teilnehmer ein, da z.B. bei einem Tauschgeschäft eine Partei stärker abhängig ist wie die andere Partei. Derartige Tauschgeschäfte sind z.B.

das Einkaufen von Nahrungsmitteln, das Anmieten einer Wohnung oder die Aufnahme eines Kredits. Dies hat zur Folge, dass z.B. ein Weltkonzern stärkeren Einfluss auf die Bedingungen eines

Liefervertrags hat als der Zulieferer. Das Ausnutzen dieser Macht sorgt erst dafür, dass die freie Marktwirtschaft effizient wird. Jedoch kann eine freie Marktwirtschaft, die durch Gewinnmaximierung und Konkurrenz gekennzeichnet ist, nicht als frei bezeichnet werden. Die ständige Angst, dass jemand von dem Nächsten übervorteilt werden kann, zerstört systematisch das Vertrauen.

Jedoch ist Vertrauen notwendig, um die Gesellschaft zusammen zu halten.

Der Wirtschaftsnobelpreisträger Friedrich August von Hayek schreibt: „Wettbewerb stellt in den meisten Fällen die effizienteste Methode dar, die wir kennen“ (Deuticke 2014, S. 24). Jedoch gibt es keine Studie, die das beweist.

Allerdings gibt es viele Studien, die untersuchen, ob Wettbewerb stärker motiviert als jede andere Methode. Eine große Mehrheit von 87% ist zu dem Entschluss gekommen, dass nicht Wettbewerb, sondern

Kooperation die effizienteste Methode ist. Anders als bei Wettbewerb motiviert Kooperation über gelingende Beziehungen, Anerkennung, Wertschätzung und gemeinsame Zielerreichung. Wettbewerb motiviert über Angst,

da viele um ihren Job, ihr Einkommen oder Status fürchten. Ein weiterer Motivationsfaktor von Wettbewerb ist die Siegeslust, also den Wunsch besser zu sein als jemand anders. Aus psychologischer Sicht spricht man

bei Menschen, die ihren Selbstwert darüber definieren, dass sie sich besser fühlen, wenn es anderen schlechter geht, von pathologischen Narzissmus. Wenn es jedoch mein Ziel ist, gute Leistungen zu erbringen

ohne das mich die Leistungen des anderen kümmern, dann brauche ich den Wettbewerb nicht. Der Wettbewerb ist aber notwendig, damit die Menschen Leistungsanreise erhalten und somit motiviert sind.

Grundsätzlich kann man zwischen der intrinsischen und der extrinsischen Motivation unterscheiden. Die intrinsische Motivation kommt von innen und wirkt stärker als die extrinsische Motivation (z.B. Wettbewerb).

Durch diese Art der Motivation entsteht die Leistung z.B. durch die persönliche Leidenschaft für eine Sache. Eine effiziente Marktwirtschaft sollte also auf einer intrinsischen Motivation aufbauen.

Durch das Verfolgen der eigenen Interessen als höchstens Ziel hat folgende Auswirkungen auf die Marktwirtschaft:

- Auf Grund des Wachstumszwangs entstehen zunehmend Großkonzerne („Global Player“), die ihre Machtposition ausspielen und Konkurrenten aufkaufen.
- Wenn im Markt nur wenige Konkurrenten vorhanden sind, werden strategische Kooperationen eingegangen, deren Ausprägung zum Teil in Form von Kartellen zu erkennen sind, da dies noch effizienter ist.
- Durch verbesserte Standortbedingungen versuchen Staaten Unternehmen anzulocken, um die Bedingungen für Gewinnmaximierung zu verbessern. Dazu zählen z.B. Lohn-, Sozial-, Steuer- und Umweltdumping.
- Die Preisgestaltung orientiert sich an der Angebot- und Nachfragemacht und spiegelt die Interessen des Mächtigen wieder.
- Je globaler der freie Wettbewerb ist, desto größer ist das Machtgefälle zwischen den Marktbeteiligen und führt zu Ungleichheiten und einer Kluft zwischen Arm und Reich.
- Das primäre Ziel des Kapitalismus ist nicht Befriedigung der Grundbedürfnisse, sondern die Vergrößerung des Kapitals. Es werden strategisch neue Bedürfnisse geweckt, hinter denen eine höhere Kaufkraft steht.
- Der Umweltschutz wird vernachlässigt, da er nicht zu der Vermehrung des Kapitals beiträgt.
- Die Anhäufung von materiellen Werten rückt in den Vordergrund und unterwirft andere Werte wie z.B. Beziehungs- und Umweltqualität. Der Konsumzwang wird zur Kaufsucht.
- Die Wirtschaft wird geprägt von Egoismus, da sie diesen durch Konkurrenzverhalten (z.B. Karriere) belohnt. Dieser Egoismus färbt auf andere Bereiche wie Politik und Medien als auch auf unsere zwischenmenschlichen Beziehungen ab.
- Die Demokratie wird schrittweise ausgeschaltet, da Wirtschaftakteure durch Lobbying, Medienbesitz oder Parteifinanzierung ihrer Interessen durchsetzen.

2 Die Gemeinwohl-Ökonomie - der Kern

Ziel des Wirtschaftens

Bei der Beschreibung der Ziele und Ausrichtungen eines Wirtschaftsunternehmens fallen häufig Begriffe wie „Geld“, „Gewinn“ oder auch „Profit“. Tatsächlich wird konträr hierzu in diversen Verfassungen wörtlich

„Die gesamte wirtschaftliche Tätigkeit dient dem Gemeinwohl“ (Felber 2014, S. 32)

verlangt.

Durch die Gemeinwohl-Ökonomie soll das verfassungsmäßige Ziel in der Wirtschaft Einzug erhalten. Das zur Zeit höchste Ziel des finanziellen Wohlstandes soll dem des maximalen Beitrags zum Gemeinwohl weichen. Um dies zu erreichen muss ein Austausch des falschen Leitsterns „Eigennutzenmaximierung“ durch den Leitstern „Gemeinwohl“ erfolgen. Dies würde dazu führen, dass das Ziel aller Unternehmen die Maximierung des eigenen Allgemeinwohl-Beitrags zur Gesellschaft darstellt. Zusätzlich muss die Erfolgsmessung auch auf diesen neuen Leitstern angepasst werden. Eine Messung des Erfolgs eines Unternehmens alleine durch das messen finanzieller Faktoren des Unternehmens ist nicht ausreichend.

„Geld ist [...] nicht das Ziel des Wirtschaftens, sondern nur das Mittel.“ (Felber 2014, S. 33)

Da das zu erreichende Ziel in keinem Zusammenhang zur Mehrung von Kapital steht, besteht für ein Unternehmen auch kein Zwang sich auf diese Mehrung zu fokussieren. Der Erfolg eines Unternehmens muss direkt am Ziel - dem Erbringen von Nutzen für das Allgemeinwohl - gemessen werden.

Die Gesellschaft wie auch einzelne Menschen benötigen kein Geld sondern Nutzwerte (zum Beispiel Nahrung, Kleidung, Wohnung, Beziehungen, usw.). Geld ist nur das Tauschmittel um Nutzwerte zu erlangen. Die Finanzen eines Landes, eines Unternehmens oder einzelner Personen geben wenig Aufschluss über deren Nutzwert-Lage. Dies sieht man am Beispiel des Brutto-Inlands-Produktes - dieses gibt wenig Aufschluss über die politische Lage eines Landes oder die gerechte Verteilung der Güter innerhalb eines Landes. Das BIP muss einem aussagekräftigeren Mittel weichen. (vgl. Felber 2014, S. 34)

Die Suche nach einer entsprechenden Alternative wurde bereits 1970 begonnen. Hierzu gibt es Versuche verschiedener Institutionen und Länder wie dem "Better Live Index" des OECD oder den "W3-Indikatoren" - "Wachstum, Wohlstand, Lebensqualität" des deutschen

Bundestages. Der Zwerghaat Butan hat mit dem "Bruttoinlands Glück" einen weniger auf mathematischen Modellen basierenden Ansatz gewählt. Hier werden jährlich alle 6000 Haushalte zu ihrer Gefühlslage befragt. Viele Ökonomen sind der Meinung, dass das Messen von "Glück" nicht möglich ist. Ein Messen des Glücks durch das Einbeziehen vieler Faktoren kommt dem tatsächlichen Glückswert allerdings wesentlich näher als die Darstellung über den alleinstehenden Faktor "Brutto-Inlands-Produkt". (vgl. Felber 2014, S. 35)

Gemeinwohl als Unternehmensbilanz messen

In der Gemeinwohl-Ökonomie wird als neue wichtigste Unternehmensbilanz die "Gemeinwohlbilanz" eingeführt. Die Finanzen eines Unternehmens erhalten einen geringeren Stellenwert. Dennoch gilt weiterhin - ein Unternehmen soll keine finanziellen Verluste machen, da hieran auch direkt die Möglichkeit der Erbringung des Produktes "Gemeinwohl" geknüpft ist. Vermieden werden sollen

"Gewinne um der Gewinne willen". (Felber 2014, S. 37)

Die Gemeinwohl-Bilanz gibt Aussage darüber, wie die fünf häufigsten Verfassungswerte demokratischer Staaten in Unternehmen umgesetzt werden: "Menschenwürde, Solidarität, Gerechtigkeit, ökologische Nachhaltigkeit und Demokratie" (Felber 2014, S. 37). Hierbei wird genauer die Erfüllung dieser Werte im Umgang mit den "Berührungsgruppen" (Intern & Extern) eines Unternehmens gemessen. Hierzu zählen Mitarbeiter und Zulieferer genau so wie zukünftige Generationen und die Umwelt.

Um die Gemeinwohl-Bilanz zu messen bzw. darzustellen werden für jede Berührungsgruppe derzeit 17 Gemeinwohl-Indikatoren aufgenommen:

Sinnhaftigkeit des Produktes, Arbeitsbedingungen, ökologische Produktion, ethischer Vertrieb der Produkte, Kooperation und Solidarität zu anderen Unternehmen, Verteilung der Erträge, Gleichbehandlung der Frauen und Männer und wie demokratisch Entscheidungen getroffen werden. (vgl. Felber 2014, S. 37,38)

Gemeinwohl definieren

Der Begriff Gemeinwohl kann nur durch einen demokratischen Entscheidungsprozess definiert werden. Eine Definition die durch die Regierung (Beispiel Diktatur) vorgegeben wird, kann zu einem Missbrauch des Begriffes selbst führen. In Diktaturen werden häufig Begriffe wie "Allgemeinwohl" oder auch "Frieden" zur Rechtfertigung der Politik eingesetzt.

"Theoretisch könnte ein Diktator oder ein totalitäres Regime behaupten, sie wüssten am besten, was für alle gut sei, und ihre Politik mit einem so verstandenen »Gemeinwohl« begründen." (Felber 2014, S. 39)

Universalbilanz, Markttransparenz und Gemeinwohl-Audit

Standards und Normen sind Richtlinien, zu denen kein Hersteller verpflichtet ist - dennoch halten sich viele Unternehmen an diese. Standards gibt es zu den verschiedensten Bereichen - auch in Bereichen die das Gemeinwohl betreffen (z.B. "Biolandbau, "Fairer Handel", usw.). Bei der Einführung von Gemeinwohl-Bilanzen auf eben dieser Freiwilligenbasis wie es bei Normen und Standards der Fall ist ergibt sich folgendes Problem:

"Sobald sie in Widerstreit mit der Hauptbilanz – der Finanzbilanz – geraten, sind sie plötzlich nichts mehr wert, denn das würde den Lebensnerv des Unternehmens angreifen und in der heutigen Systemdynamik schädigen: Wer zugunsten einer unverbindlichen Nebenbilanz den Finanzgewinn schmälert, katapultiert sich selbst aus dem Rennen." (Felber 2014, S. 41)

Eine Gemeinwohl-Bilanz kann nicht auf Freiwilligenbasis umgesetzt werden. Sie muss folgende Kriterien erfüllen:

Verbindlichkeit, Ganzheitlichkeit, Messbarkeit, Vergleichbarkeit, Verständlichkeit, Öffentlichkeit, externe Prüfung, Rechtsfolgen (vgl. Felber 2014, S. 44)

Jedes Unternehmen muss eine Gemeinwohl-Bilanz aufstellen. Innerhalb dieser Bilanz kann ein Unternehmen eine Punktestufe erreichen. Das Ergebnis dieser Bilanz könnte dann an den Produkten des Unternehmens farblich neben einem QR-Code markiert werden. Über den QR-Code kann der Käufer sich genauer über die Bilanz erkundigen - so hätten auch die Konsumenten direkte Einsicht in die Gemeinwohl-Bilanz des Unternehmens.

Die Gemeinwohlbilanz soll in erster Version durch das Unternehmen selbst erstellt werden. Hierbei werden optimaler Weise alle Mitarbeiter und Begleitunternehmen, Zulieferer und sonstige Schnittstellengruppen befragt. Anschließend erfolgt eine Prüfung durch externe AuditorInnen. Die Kritik, dass dies zu einem schiefen Bild führen wird, Unternehmen sich besser darstellen oder AuditorInnen bestechen werden, lässt sich genau so auf die aktuelle Situation anwenden. Unternehmen erstellen ihre Finanz-Bilanzen selbst. Diese werden durch externe Audits geprüft. Außerdem erfolgt eine stichprobenartige Kontrolle durch den Staat - diese wird es für die Gemeinwohl-Bilanz ebenfalls geben. Bestechungen bzw. Täuschungen und deren Versuche fließen in die Gemeinwohl-Bilanz mit ein.

Gemeinwohlstreben belohnen

Um das Gemeinwohlstreben der Unternehmen zu steigern wird ein Belohnungssystem eingeführt. Diejenigen Unternehmen, die dem Gemeinwohl nützen, werden mit Steuernachlässen oder verringerten Zollgebühren belohnt. Umgekehrt genauso - wer dem Gemeinwohl schadet zahlt einen erhöhten Steuersatz, muss mehr Zollgebühren zahlen, usw. . Ein Ausgleich der Produkt- und Unternehmenswelt findet statt. Die Unternehmen sind gezwungen gemeinnützig zu Arbeiten, gemeinnützige Produkte herzustellen, gute Arbeitsbedingungen zu schaffen und Gleichberechtigung der Mitarbeiter einzuführen.

Gewinn als Mittel

Natürlich erwirtschaften Unternehmen auch innerhalb der Gemeinwohl-Ökonomie einen Gewinn. Dieser soll allerdings nicht zu hoch sein und dessen Verwendung unterliegt gewissen Vorschriften. Gewinne können dem Gemeinwohl schaden, wie auch das Gemeinwohl mehren. Aus diesem Grund wird bei der Verwendung von Gewinnen zwischen erlaubten und nicht erlaubten Verwendungen unterschieden. Um diese im Vorfeld zu erkennen wird bei jeder größeren Investition eine Investitionsanalyse erstellt, die den "Gemeinwohlfaktor" dieser Investition darstellt.

Erlaubte Verwendungen stellen zum Beispiel Investitionen in Produkte dar, die einen hohen "Gemeinwohl-Faktor" haben (Zum Beispiel erneuerbare Energien im Gegensatz zu umweltunfreundlichen). Eine weitere Möglichkeit stellt das Rückzahlen von Krediten oder das Ausschütten der Gewinne an Mitarbeiter dar. Denkbar wäre ebenfalls das zur Verfügung stellen von zinslosen Darlehen an Mitbewerber, Zulieferer, etc. .

Nicht erlaubte Verwendungen von Gewinnen stellen jene Nutzungen dar, die dem Gemeinwohl schaden - also feindliche Übernahmen oder Finanzinvestments. Unternehmen sollen durch das erbringen ihrer Dienstleistungen oder das Verkaufen ihrer Produkte die Werte schaffen, nicht durch Finanzgeschäfte. Eine weitere nicht erlaubte Verwendung von Gewinnen ist die Ausschüttung an nicht mitarbeitende Gesellschafter.

"Durch die Trennung von entscheidungsmächtigen EigentümerInnen und Beschäftigten im Unternehmen wird Verantwortungslosigkeit bis hin zur totalen Skrupellosigkeit enthemmt" (Felber 2014, S. 54)

Parteispenden durch Unternehmen werden komplett verboten.

Ende des Wachstumszwangs

In den Erfolg eines Unternehmens spielen viele Faktoren ein wie Produktqualität, Innovationskraft, Effizienz, Größe, Flexibilität - wirklich entscheidend ist zur Zeit aber nur der Finanzgewinn. Das führt dazu, dass Unternehmen teilweise dazu gezwungen sind skrupellos, unethisch zu handeln. Es existiert ein Wachstumsdruck: keinen Finanzgewinn zu erzeugen bedeutet nicht erfolgreich zu sein. Wenn der Finanzgewinn nicht der einzige auschlaggebende Faktor wäre, könnten Unternehmen gelassen ihre "optimale Größe" ermitteln, würden keinem Wachstumsdruck mehr unterliegen.

Wachstumsdruck ist ein Faktor, der dazu führt, dass Unternehmen weniger für das Allgemeinwohl handeln. Die Quantität steigt auf Kosten der Qualität. Ein Mensch wächst physisch auch nur bis zu einem gewissen Alter, anschließend wachsen nur noch die inneren Werte - Charakter, der Umgang mit Menschen und Problemen, die fachbezogenen Fähigkeiten und emotionalen Kompetenzen. (vgl. Felber 2014, S. 61)

Das Ende des Wachstumsdrucks für Unternehmen würde zu besseren Arbeitsbedingungen, einer besseren Unternehmensausrichtung und besseren Produkten führen.

Strukturelle Kooperation

Der Übergang vom Gegeneinander zum Miteinander stellt eine der größten zu überwindenden Hürden dar. Zur Zeit gibt es verschiedenste Unternehmen die in den gleichen Branchen nach Lösungen für die gleichen Probleme suchen - gegeneinander.

"Lieg es nicht auf der Hand, dass »gegeneinander suchen« nicht effizient sein kann?"
(Felber 2014, S 62)

In der Gemeinwohl-Ökonomie wird das Konzept der Konkurrenz nicht verboten oder abgeschafft. Unternehmen dürfen weiterhin mit unterschiedlichsten Zielen und Produktideen gegründet werden. Es gibt generell die Möglichkeit, sich gegen die Gesellschaft oder das Gemeinwohl zu stellen - der Unterschied: Es bietet keine Vorteile mehr.

Das Zusammenarbeiten im Sinne einer friedlichen Koexistenz wird gefördert. Im kapitalistischen Denkmuster erinnert dies an ein Kartell. Doch ein Kartell als solches bringt den Unternehmen in der Gemeinwohl-Ökonomie keinen Mehrwert.

"Heute sind Kartelle kein Selbstzweck, sondern ein Mittel, um den Gewinn zu steigern. Wenn Gewinne begrenzt und als Mittel für die Mehrung des Gemeinwohls eingesetzt werden, dann verliert auch Kartellbildung als Mittel dazu ihren Sinn." (Felber 2014, S. 63)

Konkurs

Die Möglichkeit des Konkurses besteht weiterhin - allerdings ist ihre Eintrittswahrscheinlichkeit aus verschiedenen Gründen weniger wahrscheinlich. Die Gemeinwohlwirtschaft führt automatisch dazu, dass üblicherweise nur noch sinnvolle Unternehmen gegründet werden. Profit alleine ist kein Grund zur Unternehmensgründung. Des Weiteren gibt es statt einer feindlichen Konkurrenz eine friedliche Koexistenz der Unternehmen. Zu guter Letzt sind die Mitarbeiter eines "demokratisch geführten" Unternehmens eher motiviert und ziehen an einem Strang, sodass ein Konkurs nicht eintritt oder effektiv verhindert werden kann. (Vgl. Felber 2014, S. 64)

Kooperative Marktsteuerung

Kommt es doch so weit, dass ein Unternehmen oder ein ganzer Markt dem Konkurs unterliegt, gelten wieder die Ziele des Gemeinwohls. Unternehmen eines dem Konkurs unterliegenden Marktes können einen "Krisen- oder Kooperationsausschuss" einberufen, durch den gemeinsam das weiter Vorgehen und die sinnvollste Reaktion auf die zu überwindenden Problem erarbeitet wird. Es könnte ein koordiniertes Verkürzen der Arbeitszeiten in allen betroffenen Unternehmen erfolgen, ein Arbeitsplatzabbau mit entsprechenden Umschulungen der zu entlassenen Mitarbeiter, die Umspezialisierung weniger Unternehmen auf neue Themengebiete um den betroffenen Markt wieder zu öffnen oder, als letzte Möglichkeit, das Schließen von Betrieben mit einer Umschulung der Mitarbeiter auf Märkte mit Arbeitskräfte-Mangel. (Vgl. Felber 2014, S. 65)

Gemeinwohl und Globalisierung

Ein häufiger Kritikpunkt an der Gemeinwohl-Ökonomie ist, dass deren Einführung laut Kritikern das Zusammenspiel der gesamten Welt bedarf. Tatsächlich ist es so, dass im Konkurrenzkampf zweier Unternehmen der Unethischere / Skrupellosere gewinnt - also im globalen Kontext: Ein Land mit einer Gemeinwohl-Ökonomie hat geringe Chancen gegen Länder mit der Freihandels-Ökonomie. Genau das ist auch der Fehler im System. Das unethische und skrupellose Verhalten wird belohnt und damit weiter ausgebaut. Das direkte Einführen der Gemeinwohl-Ökonomie weltweit ist allerdings auch unrealistisch.

Zur Lösung werden zwei Vorschläge geliefert (vgl. Felber 2014, S. 67):

1. Der globale ordnungspolitische Ansatz: Es werden gemeinsame Rahmenbedingungen bestimmt (Arbeitsschutz, Sozialschutz, Umweltschutz, etc.). Die UNO wird hier als Ort der Regulierung vorgeschlagen. Die Rahmenbedingungen werden zur Umsetzung der Gemeinwohl-Ökonomie in einer ersten Länder-Zone (z.B. der EU) eingehalten. Für alle Länder die mit einem Land der Gemeinwohl-Zone handeln wollen, werden bei nicht Erfüllung einer der Standards (Arbeitsschutz, Sozialschutz, etc.) erhöhte Zollgebühren

verlangt. Hier entsteht eine Staffelung: jede nicht eingehaltene Rahmenbedingung erhöht die Zollgebühren weiter.

2. Der anreizpolitische Ansatz der Gemeinwohl-Ökonomie: Die Erstellung und Offenlegung einer Gemeinwohlbilanz für Unternehmen wird verpflichtet. Eine bessere Bilanz führt zu einem "freieren" Marktzugang, eine schlechtere zu einem schlechten Marktzugang (Erhöhen des "ethischen Schutzzolls").

Soziale Sicherheit, Freijahre, Solidaritätseinkommen und Rente

In einer Wirtschaft in der Unternehmen dem Konkurs unterliegen können, gibt es auch die Möglichkeit der Arbeitslosigkeit. In der Gemeinwohl-Ökonomie wird pro Arbeitnehmer und Pro Dekade Berufstätigkeit ein "Freijahr" zur eigenen Verfügung eingeführt. Der Arbeitnehmer erhält in dieser Zeit den gesetzlichen Mindestlohn. Da dieses Jahr für jede Person gleichermaßen gilt, ist eine Diskussion über Ungleichbehandlung oder die steuerliche Finanzierung überflüssig.

Die Freijahre hätten den Effekt, dass die Arbeitslosigkeit gesenkt werden würde und jede Person sich innerhalb des Jahres anderen Themen wie der "Weiterbildung, der Familie [...] oder anderen Passionen" (Felber 2014, S. 69) widmen könnte. Das Arbeitsklima würde hierdurch entzerrt und die Mitarbeiter motivierter.

"Aufgrund dieser geänderten Umstände erscheint es mir »systemwidrig«, dass Leistungen wie Arbeitslosen-, Notstands-, Sozialhilfe oder Hartz IV noch nötig sein werden." (Felber 2014, S. 69)

Ein Konzept zum Umgang mit Arbeitslosen muss dennoch angedacht werden. In welcher Form eine Hilfe umgesetzt wird muss noch erdacht werden. Möglich wäre ein Teil des Mindestlohns als Sozialhilfe - auch ein Bedingungsloses Grundeinkommen wäre im Sinne der Gemeinwohl-Ökonomie möglich.

"Für Menschen mit besonderen Bedürfnissen oder Einschränkungen, die sich nicht oder nur teilweise an der Erwerbsarbeit beteiligen können, soll es jedenfalls ein bedingungsloses Solidaritätseinkommen geben: zum Beispiel in der Höhe des Durchschnittseinkommens" (Felber 2014, S. 69)

Als Rente wird das derzeitige System und dessen Kopplung zu den Finanzmärkten abgeschafft.

"Die Privatisierung der Renten macht diese weder sicherer noch sozialer, noch billiger – in allen drei Kriterien tritt das Gegenteil ein."

Aus diesem Grund wird das ehemalige System des Generationenvertrages wieder eingesetzt. Die Rentensicherung stellte vor der Umstellung nie ein wirkliches Problem dar. Die Annahme, die Kopplung der Rente an die Finanzmärkte würde diese sichern ist ein Irrglaube, von dem die private Versicherungswirtschaft eigene Gewinne erwirtschaftet. Doch Gewinne und Profite sind in der Gemeinwohl-Wirtschaft keines der Ziele. Banken und Versicherungen, das gesamte Finanzsystem wird zu einem Mittel, das Gemeinwohl zu erreichen und nicht Profit zu schlagen.

3 Die Demokratische Bank

Durch die Liberalisierung und Globalisierung der Finanzmärkte entstehen zunehmend Global-Player-Banken, die nicht der Gesellschaft und dem Gemeinwohl dienen, sondern das primäre Ziel von Profit verfolgen. Die Gemeinwohl-Ökonomie benötigt daher ein anderes Finanzsystem. Die Hauptaufgabe der Banken war die Umwandlung von Spargeldern in zugängliche Kredite für regionale Unternehmen und Privathaushalte. Dieser Aufgabe kommen Sie jedoch teilweise oder unzureichend nach.

Der liberalisierte Markt fördert die globale wettbewerbsfähige Größe von Banken. Dies ist jedoch das explizite Ziel des EU-Finanzbinnenmarktes und des Weltmarktes für Finanzdienstleistungen durch die WTO. Die Banken werden dadurch ökonomisch und politisch systemrelevant. Durch die steigende Macht können sie sich zusätzlich gegen Zerteilung, Regulierungen und Besteuerung wehren.

Bei einer Gemeinwohl-Ökonomie würden in einem Finanzsystem Geld in Form von Krediten dem öffentlichen Zwecke dienen und die Finanzmärkte geschlossen werden.

Ziel und Leistungen

Die Demokratische Bank verfolgt die Ziele und Werte der Gemeinwohl-Ökonomie. Sie ist daher nicht gewinnorientiert und fördert insbesondere regionale Wirtschaftskreisläufe. Zu ihren Kernleistungen zählen u.A. die „unbeschränkte Garantie der Spareinlagen“, „kostenlose Girokonten für alle WohnsitzbürgerInnen“, ein „flächendeckendes Filialnetz mit wertschätzender persönlicher Betreuung“, „kostenlose Ergänzungskredite“ und der „Wechsel von Währungen“ (Deuticke 2014, S. 74). Die Leistungen und Ziele der demokratischen Bank sind in der Verfassung niedergeschrieben und können nur per Volksabstimmung geändert werden.

Transparenz und Sicherheit

Die Demokratische Bank nimmt lediglich die Rolle des Geldvermittlers zwischen SparerInnen und KreditnehmerInnen ein. Zusätzlich muss sie alle Geschäfte in der Bankbilanz vermerken und gesetzliche Eigenkapitalvorschriften einhalten.

Finanzierung, Refinanzierung, Konkurs

Die Mitarbeiter der Demokratischen Bank „genießen hohe soziale Sicherheit und umfassende Mitbestimmungsrechte“ und erhalten ein „menschenwürdiges Einkommen“ (Deuticke 2014, S. 75). Kreditvergabe der Bank erfolgt aus den Einlagen von Privatpersonen, Unternehmen und Staat. Diese Finanzvermögen wachsen in Relation mit der realen Wirtschaftsleistung (BIP), sodass ein ausreichendes Kreditkapital für Refinanzierungen zur Verfügung steht. Im Fall, dass die Spareinlagen in einer Gemeinde, Region oder Bundesland nicht ausreichen, um die Kreditanfragen zu decken, verteilen andere Banken diese um. Dabei haftet die Zentralbank für dieses Umverteilungsrisiko. Die Wahrscheinlichkeit des Konkurses einer Zweigstelle der Demokratischen Bank ist gering, da sie ähnlich wie andere staatliche Einrichtungen (z.B. Schulen, Krankenhäuser), welche auch nicht Konkurs gehen können, behandelt wird.

Zinsen und Inflation

KreditnehmerInnen zahlen eine Kreditgebühr, die sich nach den entstehenden Kosten der Bank durch die Kreditvergabe orientiert und der Bank keinen Gewinn bringt. Kredit- und Sparzinsen existieren nicht mehr. Die Möglichkeit der Inflation wird verringert, da es kein Wachstumszwang in der Gemeinwohl-Ökonomie gibt.

Soziale und ökologische Kreditprüfung

Die Kreditvergabe wird beeinflusst durch die Kenntnis der lokalen Situation und der Wirtschaftsakteure. Die Kredite werden nicht mehr nach ökonomischer Rentabilität vergeben, sondern nach sozialen und ökologischen Mehrwert. Die Investitionen, die einen besonders hohen sozialen und ökologischen Mehrwert bringen, werden durch kostenlose Kredite oder teilweise Kredite mit negativen Zinssatz unterstützt. Auf der anderen Seite erhalten Investitionsvorhaben, die einen sozialen oder ökologischen Minderwert schaffen, keinen Kredit.

Ökosoziales Risikokapital und Gemeinwohl-Börsen

Am Risikokapitalmarkt (z.B. Börse) herrscht die Hoffnung, dass Projekte mit ungewisser Rentabilität finanziert werden. Diese Projekte schaffen jedoch oft keinen sozialen und ökologischen Mehrwert. Die Demokratische Bank könnte dieses Prinzip jedoch aufgreifen und eine Risiko-Abteilung schaffen, die Innovationen mit sozialen und ökologischen Mehrwert finanziert. Dazu wird ein kleiner Prozentsatz der Spareinlagen als ökosoziales Risikokapital bereitgestellt. Eine andere Möglichkeit ist die Einrichtung von regionalen

Gemeinwohl-Börsen, die von den Banken einer Region getragen werden könnten. An der Börse können Kreditanfragen, die zwar die finanzielle Bonitätsprüfung nicht bestehen, jedoch einen sozialen und ökologischen Mehrwert schaffen durch Unternehmen unterstützt werden.

Verhältnis zu Privatbanken

Für einen fairen Wettbewerb müssen Privatbanken in nicht gewinnorientierte Rechtsformen umgewandelt werden und das Investmentbanking abgeschafft werden. Dieser Übergang kann erzielt werden, indem gemeinwohlorientierte Banken staatliche Unterstützungsleistungen erhalten, denen man gewinnorientierte Banken verweigert.

Zentralbank

Die Zentralbank ist Teil des Demokratischen Bankensystems und transparent und demokratisch organisiert. Der Leitungskreis umfasst VertreterInnen aller Gesellschaftsbereiche. Die Zentralbank besitzt das Geldschöpfungsmonopol und stellt dem Staat Geld in begrenztem Maß zur Verfügung. Sie finanziert den Staat über unverzinste Kredite oder durch „Erweiterung der Geldmenge als Geschenk an den Staatshaushalt“ (Deuticke 2014, S. 82).

Die Zentralbank ist an einer globalen Währungskooperation beteiligt. Die Kooperation umfasst die Schaffung einer neutralen Verrechnungseinheit für den internationalen Handel (z.B. „Globo“ oder „Terra“). Die nationalen Währungen bestehen weiterhin. Der „Globo“ stellt daher eine Komplementärwährung auf internationaler Ebene dar. Derartige Komplementärwährungen kann es auch auf regionaler Ebene geben. Die Entscheidung für weitere regionale Währungen tragen dabei die Demokratischen Banken.

4 Eigentum

Derzeit ist das Eigentum ungleich verteilt, sodass einzelne Personen und Unternehmen Medien und Politik kontrollieren können. Das Privateigentum gefährdet die Demokratie und die Freiheit und Gleichheit der Menschen. Die Gemeinwohl-Ökonomie möchte diese Problematiken umgehen, sodass eine ethische und trotzdem liberale Marktwirtschaft entsteht.

Negative Rückkopplung

Der Kapitalismus ist ein positiv rückgekoppeltes System, sodass die reiche Elite immer reicher wird. Diese Rückkopplung der Bildung einer finanziellen Elite könnte durch verschiedene Maßnahmen unterbunden werden. Es könnte die Einkommensungleichheit, das Privatvermögen und das Unternehmensvermögen in privater Hand begrenzt werden. Des Weiteren besteht die Möglichkeit das Erbrecht neuzugestalten und die mögliche Erbmasse zu limitieren.

Relative Begrenzung der Einkommensungleichheit

Eine sehr starke Einkommensungleichheit wird weder Leistung noch Verantwortung fördern. Die Reichen werden hierdurch nicht glücklich, sondern gierig und die Armen fühlen sich minderwertig. Zudem wird Aggression und Kriminalität ansteigen, Stress und Krankheiten werden gefördert. In der Gemeinwohl-Ökonomie soll ein demokratischer Wirtschaftskonvent mögliche Grenzen einer Einkommensungleichheit erarbeiten. Hierbei wird nur das Arbeitseinkommen und die Einnahmen aus Vermietung betrachtet, da es kein Kapitaleinkommen mehr gibt. Hierbei wird auch ein Mindestlohn definiert, der ein menschenwürdiges Leben ermöglichen soll.

Begrenzung der Privatvermögen

Der Wirtschaftskonvent soll nicht nur die Begrenzung des Einkommens, sondern auch eine Obergrenze des Privatvermögens festlegen.

Demokratisierung von Großunternehmen

Global agierende Unternehmen haben heutzutage mehr Einfluss als manche Regierungen. Die Unternehmen werden von einer kleinen Zahl von Privatpersonen gelenkt, ohne demokratischen Einfluss. Unternehmen sollen anhand ihrer Größe eine Demokratisierung erfahren. So könnten z.B. kleine Unternehmen bis 250 Beschäftigten weiterhin in privater Hand bleiben. Ab 250 Beschäftigten erhalten die Mitarbeiter 25% der Stimmrechte. Bei steigender Größe der Belegschaft wird das Unternehmen weitergehend demokratisiert. Zudem könnte große Unternehmen durch ein regierungsunabhängiges regionales Wirtschaftsparlament geführt werden, welches direktdemokratisch gewählt wird.

Mitarbeiterbeteiligung

Es wird angestrebt, dass möglichst viele Menschen Eigentum an den Unternehmen gelangen und so die Unternehmen steuern können und zwangsläufig das Verlustrisiko gemeinsam tragen. Z.B. können den Mitarbeitern Mitarbeiterfonds übertragen werden.

Gewinnbindung an das Unternehmen

Die Gewinnausschüttung des Unternehmens an den Gründer soll mit steigenden Beitrag anderer Menschen am Unternehmenserfolg abnehmen. Somit wird nach einiger Zeit jeder Gewinn in Investitionen fließen müssen.

Begrenzung des Erbrechts, Generationenfonds und demokratische Mitgift

Das uneingeschränkte Erbrecht führte dazu, dass das Startkapital in privater Hand sehr ungleich verteilt war. Die Gemeinwohl-Ökonomie sieht eine Erbmasse bis zu einem bestimmten Betrag als steuerfrei an. Darüber hinaus soll die Erbmasse in einen öffentlichen Generationenfonds eingespeist werden und zu gleichen Teilen an die nachfolgenden Generationen verteilt werden. Somit wird ein großes konzentriertes Vermögen zerkleinert auf die breite Masse.

Immobilien

Immobilien können innerhalb dieses Freibetrages auch vererbt werden. Wenn die Immobilie einen höheren Wert hat, so könne man diese mit mehreren Erben weiterhin besitzen, ohne dass die Immobilie dem Generationenfonds zugeführt wird.

Vererbung von Unternehmen

In der Bundesrepublik Deutschland können derzeit Unternehmen steuerfrei an die Erben vererbt werden, ungeachtet derer Qualifikation. In der Gemeinwohl-Ökonomie wird es vermieden, dass riesige Unternehmen in einzelnen privaten Händen gehalten werden, sondern auf die breite Masse verteilt werden. So könnten z.B. bei Aktiengesellschaften Anteile bis zu der definierten Größe weiter vererbt werden, darüber hinaus werden die Anteile dem Generationenfonds zugeführt. Bei Familienunternehmen wird dieser mögliche Erbbetrag deutlich angehoben auf z.B. zehn Millionen Euro. Die weiteren Firmenanteile können an die Belegschaft verteilt werden und an ausgewählte Nichtfamilienmitglieder bis zu einer Obergrenze verteilt werden. Die Erben müssten sich verpflichten z.B. mindestens drei Jahre in dem Betrieb zu arbeiten.

Dem Menschen werden heute nur zwei Möglichkeiten geboten. Zum einen der Verkauf der eigenen Arbeitskraft (strukturelle Sklaverei) und zum anderen ein eigenes erfolgreiches Unternehmen zu gründen. Im Kapitalismus verdient der Kapitalgeber an dem Mehrwert der Arbeitsleistung des Arbeitnehmers. Somit ist dies ein unfreies und ausbeuterisches System. Gerechter wäre es, wenn alle Arbeitenden und Geldgebenden betriebliche Entscheidungsgewalt beherrschen. Die Erträge unter allen Arbeitenden aufgeteilt werden, die Firmenanteile an möglichst viele verteilt werden. Somit tragen alle das unternehmerische Risiko und müssen Verantwortung übernehmen. Es müssen aber nicht zwangsläufig alle Arbeitenden diese Verantwortung übernehmen, aber die Möglichkeit sollte bestehen.

Schenkung

Es wird in der Gemeinwohl-Ökonomie einen Schenkungsfreibetrag eingeführt, ähnlich dem Erbfreibetrag, bei dem bis zu dem festgelegten Betrag Finanzmittel übertragen werden können.

Demokratische Allmenden

Die dritte Kategorie des Eigentums in der Gemeinwohl-Ökonomie ist das öffentliche Gemeinschaftseigentum. Dies könnten z.B. die Bahn, die Post, Bildungseinrichtungen, Stadtwerke, Kindergärten und Banken sein. Dieses Eigentum könnte durch ein direkt gewähltes Leistungsgremium aus Vertretern des Staates, der Belegschaft, der Nutzer, Gender-Beauftragten und einem Zukunftsanwalt verwaltet werden. Klassische Staatsunternehmen, welche ausschließlich durch den Staat kontrolliert werden, wird es somit nicht mehr geben. Das Daseinsvorsorgekonvent definiert das öffentliche Gemeinschaftseigentum und bestimmt die Spielregeln für die Organisation.

Eigentum an Natur

In der Gemeinwohl-Ökonomie soll der Bürger ein Eigentum an Grund und Boden besitzen. Die Bürger, welche es für konkrete Nutzungszwecke benötigt, wird ein Anteil kostenlos zur Bewirtschaftung überlassen. Demzufolge entfällt die Grundsteuer für den primären Sektor. Bei einer besonders ökologischen Nutzung der Fläche wird dem Bauern mehr Fläche zugeteilt werden.

Für den privaten Wohnraum könnte ein Anteil an die einzelnen Parteien durch die Gemeinden zugeteilt werden. Jedem Mensch steht grundsätzlich das Recht auf Wohnfläche zu bis zu einer Obergrenze an qm. Diese Fläche wird dann zu einem festgelegten qm-Preis erworben.

Eine extreme Ungleichverteilung an Immobilien wird hiermit umgangen.

Freiheit und Gleichheit

Das Eigentumsrecht wird auf ein für ein gutes Leben nötiges Eigentum begrenzt, da sonst die Freiheiten anderer Menschen gefährdet werden.

5 Motivation und Sinn

Motivation

Bedenken hinsichtlich fehlender Motivation der einzelnen Marktteilnehmer in der Wirtschaft sollen mit den folgenden Punkten beseitigt werden:

1. In der Gemeinwohl-Ökonomie besteht weiterhin ein Erwerbszwang und die durchschnittliche Arbeitszeit soll reduziert werden. Zusätzlich wird es für nicht Erwerbstätige ein Solidaritätseinkommen geben, welches ein Überleben in Würde sichern soll. Zudem gibt es vier Freijahre.
2. Das maximale Einkommen in privaten Unternehmen wird gedeckelt und an den Mindestlohn gebunden. Mit welchem Hebel wird von dem Souverän beschlossen.
3. Höhere Einkommen als 20000 US-Dollar machen nicht mehr glücklicher.
4. Forschungen zeigen, dass Menschen durch das Streben nach Autonomie, Identität, Kompetenz, Beitrag, Gemeinschaft und Beziehung stärker motiviert werden als durch das Einkommen.
5. Autonomie. Menschen, die frei in ihren Gefühlen, Bedürfnissen und Gedanken sind.
6. Identität. Es ist nicht wichtig besser zu sein als andere, sondern anders.
7. Kompetenz. Kompetenzen lassen sich genauso in kooperativen Strukturen entwickeln, wie auch in Konkurrenzstrukturen.
8. Beitrag. Jeder Mensch will und kann seinen Beitrag leisten.
9. Gemeinschaft. Jeder Mensch sehnt sich nach einer Gemeinschaft und erfährt dort Geborgenheit, Sicherheit, Wertschätzung und Anerkennung.
10. Beziehung. Menschen sind am glücklichsten in Momenten von Verbundenheit mit sich selbst, mit anderen Mensch, mit der Natur oder mit dem großen Ganzen.

Diese Punkte könnten in der Wirtschaft als Ziel bestimmt werden.

Das Kapital und die Stimmrechte haben in der Gemeinwohl – Ökonomie eine Mehrzahl an Menschen inne, sodass eine höhere Risikobereitschaft der Unternehmen gewährleistet ist.

Sinn

In der Gemeinwohl-Ökonomie ist der Sinn des Arbeitens eine Bedürfnisbefriedigung, Stärkung des Gemeinwohls und sinnvolle Arbeit und nicht primär das Einkommen.

Intrinsische Motivation ist zudem viel stärker als extrinsische. Die meisten Kinder werden zur Leistungserbringung durch extrinsische Motivation erzogen. Dieses Erlernte setzen sie später auch im Berufsleben um und substituieren die elterliche Liebe durch Geld. Menschen sollen mehr auf ein gutes Zusammenleben achten und die Gefühle und Bedürfnisse anderer wahren.

Erziehung und Bildung

Durch das Umdenken der Bevölkerung auf das Gemeinwohl, sind die folgenden Unterrichtsfächer viel relevanter für den Schulbetrieb, als die derzeit Üblichen.

1. Gefühlskunde. Hier wird gelehrt wie Kinder ihre eigenen Gefühle wahrzunehmen haben, sowie die Kommunikation dieser.
2. Wertekunde. Unbewusste Wertvorstellungen der Kinder werden aufgezeigt. Außerdem werden die Auswirkungen eines konkurrierenden oder kooperativen Handelns der Kinder zueinander aufgezeigt.
3. Kommunikationskunde. In der Kommunikation wird eine sachliche Diskussion geschult, bei der auf persönliche Werte und Beleidigungen verzichtet wird.
4. Demokratiekunde. Hier werden demokratische Prozesse vermittelt und das Grundkonzept einer Demokratie aufgezeigt,
5. Naturerfahrenskunde. Hier werden nicht nur biologische Sachverhalte gelehrt, sondern auch eine gute Beziehung zur Natur. Durch diese Beziehung wird die Lebensqualität gesteigert.
6. Kunsthandwerk. Die Kinder sollen selber kleine Kunsthandsarbeiten erledigen, da dieses glücklich macht und außerdem die Befassung mit Materialien und Werkzeugen zu einem ganzheitlichen Leben dazu gehört.
7. Körpersensibilisierung. Dem Kindern wird gelehrt auf den eigenen Körper zu hören und ihn zu achten.

6 Weiterentwicklung der Demokratie

Die gewählten Repräsentanten der Demokratie entfernen sich weiter von den Bürgerinteressen, deren Stimme sie erhalten haben. Diese Distanz wird mit folgend erklärt.

1. Die Wahlversprechen aus dem Parteiprogramm werden meist gebrochen und der Stimmberchtige hat kein Recht auf Erfüllung der Versprechen.
2. Die Führungsriege mancher Konzerne ist zu sehr in politische Prozesse involviert. Des Weiteren werden Politiker häufig in Management Positionen eingesetzt.
3. Das Management der Betriebe und Politiker haben einen starken Einfluss auf die Medien
4. Die Bildungseinrichtungen sind abhängig von dem finanziellen Hahn der Wirtschaft.
5. Thinktanks arbeiten für den Auftraggeber, von dem sie auch eine Bezahlung beziehen.
6. Parteien werden durch Geldflüsse gelenkt

Die demokratischen Prozesse müssen offener gestaltet werden, sodass eine Vielzahl der Menschen daran teilhaben kann und sie mitgestalten können. Die gemeine Bevölkerung benötigt mehr Mitspracherechte und der Lobbyismus muss bekämpft werden und auch die Medien müssen ihre Unabhängigkeit zurück erlangen.

Gewaltenteilung & Souverän

Die Macht darf im Staate nicht konzentriert sein, sodass diese geteilt werden muss – Gewaltenteilung. Außerdem muss dem Volke eine Möglichkeit innerhalb der Legislaturperiode gegeben werden, um die gewählte Vertretung ggf. korrigieren zu können.

Dreistufige direkte Demokratie

Erste Stufe: Das Volk kann für ein gewünschtes Gesetz Unterstützungserklärungen sammeln

Zweite Stufe: Wenn eine bestimmte Anzahl an Unterstützungserklärungen gesammelt wurde, wird ein Volksbegehren eingeleitet.

Dritte Stufe: Volksabstimmung mit rechtlich bindendem Gesetz, wie in der Schweizerischen Eidgenossenschaft.

Die direkte Demokratie birgt auch Nachteile, da das Volk nicht in jedem Sachverhalt so aufgeklärt ist wie die parlamentarischen Vertreter. Doch es gibt auch viele Vorteile. Die direkte Demokratie soll die alte Form nicht ergänzen, sondern erweitern. Das Volk kann einzelne Sachfragen herausgreifen, selbstständig entscheiden und nicht auf die Erfüllung der Wahlversprechen hoffen. Für ethische Entscheidungen können alle Menschen eine hohe Kompetenz aufweisen. Wenn die Bevölkerung in verschiedenen Fragen auch berechtigt ist eine Entscheidung zu treffen, wird das Interesse auch an den Sachverhalten ansteigen. Dennoch müsste die Macht der Medien wie z.B. der populistischen „Bild Zeitung“ reglementiert werden. Zudem sind viele politische Vertreter nicht qualifizierter als die Vertretenden.

Trennung von verfassungsgebender und verfasster Gewalt

Es muss nicht nur eine Gewaltenteilung im Staate herrschen, sondern die Gewalt muss auch geteilt werden, welche die Verfassung schreibt und die Verfassung einsetzt. Z.B. wurde die EU geründet von den Regierungen der Mitgliedsstaaten, welche auch die Verfassungen schrieben. Dies ist somit ein Bruch der zweiten Gewaltenteilung.

Wirtschaftskonvent

Der demokratische Wirtschaftskonvent hat die Aufgabe, die zuvor in der Verfassung definierten Ziele in Gesetze zu verwandeln.

Bildungskonvent

Ein demokratischer Bildungskonvent wird von den betroffenen (Schüler, Studenten, Lehrer etc.) gewählt und ist dafür zuständig, Ziele und Inhalte der Bildungsweges der Betroffenen festzulegen.

Daseinsvorsorgekonvent

Der Daseinsvorsorgekonvent entscheidet welche Betriebe z.B. Schulen, Bahn, etc. in der Hand der Öffentlichkeit verbleibt bzw. neu dazu kommt.

Medienkonvent

Ein Medienkonvent stellt die Unabhängigkeit der Medien sicher.

Demokratiekonvent

Der Demokratiekonvent ist für die Ausarbeitung einer neuen Strategie zur Erfüllung einer wirklich demokratischen Gesellschaft verantwortlich.

7 Beispiele, Verwandte und Vorbilder

Dass die Gemeinwohl-Ökonomie keine Utopie ist, belegen viele Unternehmen, die das Prinzip der Kooperation verfolgen. Vielen Unternehmen ist gemeinsam, dass Kapital nur ein Mittel ist, um höhere und vielfältigere Ziele zu verfolgen. Die folgenden Beispiele zeigen, dass Unternehmen viele Aspekte der Gemeinwohl-Ökonomie bereits leben.

Bio- und Fair-Trade Betriebe – Diese Betriebe produzieren z.B. Bio-Lebensmittel, Gesundheitsprodukte und Textilien aus ökologischen Anbau. Des Weiteren legen diese einen hohen Wert auf faire Arbeits- und Handelsbedingungen, sodass alle Beteiligten der Wertschöpfungskette zufrieden sind. Einige Betriebe achten z.B. darauf, dass die Mitarbeiter kranken- und unfallversichert sind oder grüne Energie für Produktionsprozesse verwendet wird. In einer Gemeinwohlökonomie würden die Produkte, die nicht fair gehandelt werden z.B. durch Zollaufschläge benachteiligt werden, sodass nach einer mehrjährigen Übergangszeit nur noch faire Produkte produziert werden würden.

Community Supported Agriculture – Bei dem Konzept der „gemeinschaftsgestützen Landwirtschaft“ versorgt ein Hof sein Umfeld mit Lebensmitteln. Im Gegensatz stellt dieses Umfeld dem Hof die notwendigen Finanzmittel bereit und gibt eine Abnahmegarantie von sechs bis zwölf Monaten.

Banken – Ethik-Banken arbeiten nach sozial-ökologischen Grundsätzen, indem sie Unternehmen in den Bereichen „freie Schulen und Kindergärten, regenerative Energien, Behinderteneinrichtungen, Wohnen und nachhaltiges Bauen und Leben im Alter“ (Deuticke 2014, S. 155) finanzieren. Einige Ethik-Banken streben keinen Gewinn an, sondern agieren nach dem Kostendeckungsprinzip. Die Kredit- und Sparzinsen sind gering. Die Sparda-Bank agiert nach dem Prinzip der Gemeinwohl-Ökonomie, indem sie kostenlose Girokonten bereitstellen. Alle Konto-Kunden sind gleichzeitig stimmberechtigte Eigentümer, die den Aufsichtsrat und Vorstand wählen können. Die VertreterInnen entscheiden dann über die Verteilung der Gewinne.

Open-source und Free-software-Bewegung – Die Gemeinwohl-Ökonomie fördert die Weitergabe und nicht die kommerzielle Verwertung von Wissen. Die Aktivisten der Open-source und Free-software-Bewegung verfolgen das Ziel, dass Software nicht patentiert werden, sondern offen und kooperativ entwickelt werden soll. Aus dieser Forderung sind freie Hightech-Produkte wie z.B. das Betriebssystem Linux, der Webbrowser Firefox oder das Mailprogramm Thunderbird entstanden.

8 Umsetzungsstrategie

Im Jahr 2010 wurde die erste Fassung der Gemeinwohl-Ökonomie mit 15 sogenannten "Attac-UnternehmerInnen" entwickelt. Ein erster Versuch der Umsetzung wurde gestartet. Hieran waren und sind verschiedene Personen- und Wirkungsgruppen Beteiltigt.

Auf der Website scheinen außerdem ideelle UnterstützerInnen in fünf Kategorien auf. Bis Ende 2014 waren es rund 1750 Unternehmen, 6000 Privatpersonen, 65 PolitikerInnen (zahlloser Parteien) sowie sieben Gemeinden/Regionen. (Felber 2014, S. 167)

Im Folgenden werden die beteiligten Personen-Gruppen und -Kreise kurz erläutert:

Pionierinnen-Gruppen

1. *Wirtschaftliche PionierInnen* sind Pionier-Unternehmen und Organisationen die bereits die Gemeinwohl-Ökonomie auf erster Basis umsetzen. Sie erstellen die Gemeinwohl-Bilanz, kooperieren mit anderen Unternehmen statt zu konkurrieren und verbreiten die Idee der Gemeinwohl-Ökonomie regional.
2. *Politische PionierInnen* sind Gemeinwohl-Gemeinden, die Gemeinwohl-Bilanzen kommunaler Betriebe erstellen, diese Betriebe zu Versammlungen einlädt in denen das Verständnis und die Akzeptanz der Gemeinwohl-Ökonomie gesteigert wird und BürgerInnen bei der Entstehung des Gemeinwohl-Index (Gemeinwohl-Faktoren) beteiligt und einbezieht.
3. *Kulturelle PionierInnen* sind Lehrerkräfte (sowohl Schule als auch Universität) die das Verständnis für die Gemeinwohl-Ökonomie in den entsprechenden Institutionen stärken. Zusätzlich werden Überlegungen zu einem "Gemeinwohl-Ökonomie"-Studiengang angestellt.

AkteurInnen-Kreise

1. *RedakteurInnen* arbeiten an den Gemeinwohl-Kriterien durch einbeziehen des Feedbacks durch weitere PionierInnen.
2. *BeraterInnen* unterstützen die Pionier-Unternehmen bei der Umsetzung der Gemeinwohl-Ökonomie (Hilfe beim erstellen der Gemeinwohl-Bilanz, Verbessern der Prozesse, usw.).
3. *AuditorInnen* überprüfen die eingereichten Gemeinwohl-Bilanzen der Pionier-Unternehmen. Außerdem führen sie regelmäßige Kontrollen in den auditierten

Unternehmen durch.

4. *ReferentInnen* halten Vorträge zur Gemeinwohl-Ökonomie in Unternehmen, Gemeinden, Lehrinstitutionen und politischen Institutionen
5. *BotschafterInnen* sind Personen mit einem großen Wirkungskreis (zum Beispiel Helmut Lind, CEO der Sparda-Bank München) die den Bekanntheitsgrad und die Akzeptanz der Gemeinwohl-Ökonomie steigern.
6. *WissenschaftlerInnen* unterschiedlichster Disziplinen entwickeln gemeinsam an der Gemeinwohl-Ökonomie weiter.
7. *KonsumentInnen* fragen bei Unternehmen bei denen sie Kunden sind nach, ob diese Unternehmen bereits eine Gemeinwohl-Bilanz erstellt haben - als Grundlage für eine Kaufentscheidung.

EnergieFelder

Energiefelder sind lokale oder regionale Gruppen, welche die GWÖ in den Gemeinden, Städten, Bezirken, Landkreisen und Großregionen verwirklichen. (Felber 2014, S. 177)

Energiefelder unterstützen die einzelnen PionierInnen-Gruppen. Sie fördern das Verständnis und die Akzeptanz durch öffentliche Veranstaltungen und unterstützen die Weiterentwicklung der Gemeinwohl-Ökonomie. Mehrere Energiefelder vernetzen sich zu einem Größeren.

Positive Rückkopplungen

Positive Rückkopplungen sind Reaktionen auf Verhalten einzelner "Systemmitglieder" um die Gemeinwohl-Ökonomie zu stärken. Sie sind wie in einem Regelkreis die Rückwirkung auf das Systemverhalten und dienen der richtigen Aussteuerung der Eingangsgrößen.

Es werden einige Beispiele genannt (vgl. Felber 2014, S. 178/179):

- günstige Kredite für Unternehmen mit hoher Gemeinwohl-Bilanz
- das Nutzen von Gemeinwohl-Banken erhöht die Gemeinwohl-Bilanz
- das Kooperieren mit anderen Unternehmen erhöht die Gemeinwohl-Bilanz
- usw.

Die positive Rückkopplung führt zum Verstärken des gemeinwohl-ökonomischen Verhaltens und zur Schwächung des Entgegengesetzten.

Strategische Vernetzung

Die strategische Vernetzung beschreibt die Notwendigkeit der Vernetzung ähnlicher oder verwandter Ansätze untereinander, sodass jeder einzelne dieser Ansätze von der erhöhten Verbreitung profitieren kann. Die Einzel-Akzeptanz vieler dieser Ansätze kann zu einer vergrößerten Akzeptanz und damit zu einem Paradigmenwechsel innerhalb der Gesellschaft führen. Außerdem sollen die vernetzten Ansätze sich gegenseitig helfen, infragestellen und damit verbessern.

Zusammenfassung der vorbereiteten Kapitel.

1 Gesellschaft und Veränderung

1.1 Kurzanalyse

Werte bilden die Grundorientierung unseres Lebens. Sie bilden einen Leitstern in unserem Leben und prägen unsere zwischenmenschlichen Beziehungen. Zu diesen Werten gehören z.B.

Vertrauensbildung, Ehrlichkeit, Wertschätzung, Respekt, Kooperation, gegenseitige Hilfe und Teilen. Auch die Wirtschaft besitzt Werte, die jedoch bei genauerer Betrachtung konträr zu den zuvor genannten Werten sind. In der freien Marktwirtschaft gelten Prinzipien wie Gewinnstreben und Konkurrenz. Diese Prinzipien fördern jedoch Egoismus, Gier, Geiz, Neid, Rücksichtlosigkeit.

Die freie Marktwirtschaft bildet also einen weiteren Leitstern in unserem Leben. Da wir an den Werten unser Handeln orientieren, haben diese Werte fatale Folgen auf unsere Gesellschaft.

Die Gesellschaft kommt in einen Konflikt, da die Leitsterne in gegensätzliche Richtungen zeigen. Soll sich die Gesellschaft solidarisch und kooperativ verhalten oder die eigenen Vorteile vorzugsweise im Blick haben? Die Werte der Marktwirtschaft werden jedoch durch die Legislative in Form von Gesetzen und Regulieren oder durch Abkommen zwischen den Nationalstaaten weiter unterstützt.

Das Gemeinwohl in der Wirtschaft sollte durch Konkurrenz und durch die persönliche Gewinnmaximierung entstehen. Der Nationalökonom Adam Smith begründete dies vor 250 Jahren wörtlich mit:

„Nicht vom Wohlwollen des Metzgers, Bäckers, Brauers erwarten wir unsere tägliche Mahlzeit, sondern davon, dass sie ihre eigenen Interessen wahrnehmen.“ (Deuticke 2014, S. 19) Jedoch waren Unternehmen

vor 250 Jahren überwiegend klein, besaßen weniger Macht und agierten primär lokal. Oft waren die Unternehmer Gründer oder Eigentümer und bildeten mit Arbeitnehmer eine Personalunion.

In der heutigen Zeit sind jedoch immer mehr anonyme, global agierende Unternehmen zu finden. Durch die globale Aktivität erfahren diese Unternehmen mehr Konkurrenz. Die Unternehmen stehen

dadurch stärker im Wettbewerb hinsichtlich der Preisgestaltung und der Qualität ihrer Produkte. Die Konkurrenz sorgt auf der einen Seite für stärkere Leistungsanreize, jedoch hat sie

Auswirkungen auf die zwischenmenschlichen Beziehungen. Das oberste Ziel ist den eigenen Vorteil anzustreben und gegeneinander zu agieren. Das Übervorteilen wird so zur Normalität. Obwohl

die Würde der höchste aller Werte und im Grundgesetz verankert ist, sorgt die Konkurrenz dafür, dass wir Menschen nicht gleichwertig behandeln. Der Begriff Würde steht für den

"gleichen, bedingungslosen, unveräußerlichen Wert aller Menschen" (Deuticke 2014, S. 21). Daraus resultiert die Gleichheit aller Menschen. In der freien Marktwirtschaft ist jedoch üblich andere Menschen zu instrumentalisieren und übervorteilen und somit die Würde des Einzelnen zu verletzen. Wenn der eigene Vorteil unser höchstes Ziel ist, werden wir zwangsläufig Mittel für unsere Zwecke benutzen und andere übervorteilen.

Die freie Marktwirtschaft schränkt die Freiheit der Teilnehmer ein, da z.B. bei einem Tauschgeschäft eine Partei stärker abhängig ist wie die andere Partei. Derartige Tauschgeschäfte sind z.B.

das Einkaufen von Nahrungsmitteln, das Anmieten einer Wohnung oder die Aufnahme eines Kredits. Dies hat zur Folge, dass z.B. ein Weltkonzern stärkeren Einfluss auf die Bedingungen eines

Liefervertrags hat als der Zulieferer. Das Ausnutzen dieser Macht sorgt erst dafür, dass die freie Marktwirtschaft effizient wird. Jedoch kann eine freie Marktwirtschaft, die durch Gewinnmaximierung und Konkurrenz gekennzeichnet ist, nicht als frei bezeichnet werden. Die ständige Angst, dass jemand von dem Nächsten übervorteilt werden kann, zerstört systematisch das Vertrauen.

Jedoch ist Vertrauen notwendig, um die Gesellschaft zusammen zu halten.

Der Wirtschaftsnobelpreisträger Friedrich August von Hayek schreibt: „Wettbewerb stellt in den meisten Fällen die effizienteste Methode dar, die wir kennen“ (Deuticke 2014, S. 24). Jedoch gibt es keine Studie, die das beweist.

Allerdings gibt es viele Studien, die untersuchen, ob Wettbewerb stärker motiviert als jede andere Methode. Eine große Mehrheit von 87% ist zu dem Entschluss gekommen, dass nicht Wettbewerb, sondern

Kooperation die effizienteste Methode ist. Anders als bei Wettbewerb motiviert Kooperation über gelingende Beziehungen, Anerkennung, Wertschätzung und gemeinsame Zielerreichung. Wettbewerb motiviert über Angst,

da viele um ihren Job, ihr Einkommen oder Status fürchten. Ein weiterer Motivationsfaktor von Wettbewerb ist die Siegeslust, also den Wunsch besser zu sein als jemand anders. Aus psychologischer Sicht spricht man

bei Menschen, die ihren Selbstwert darüber definieren, dass sie sich besser fühlen, wenn es anderen schlechter geht, von pathologischen Narzissmus. Wenn es jedoch mein Ziel ist, gute Leistungen zu erbringen

ohne das mich die Leistungen des anderen kümmern, dann brauche ich den Wettbewerb nicht. Der Wettbewerb ist aber notwendig, damit die Menschen Leistungsanreise erhalten und somit motiviert sind.

Grundsätzlich kann man zwischen der intrinsischen und der extrinsischen Motivation unterscheiden. Die intrinsische Motivation kommt von innen und wirkt stärker als die extrinsische Motivation (z.B. Wettbewerb).

Durch diese Art der Motivation entsteht die Leistung z.B. durch die persönliche Leidenschaft für eine Sache. Eine effiziente Marktwirtschaft sollte also auf einer intrinsischen Motivation aufbauen.

Durch das Verfolgen der eigenen Interessen als höchstens Ziel hat folgende Auswirkungen auf die Marktwirtschaft:

- Auf Grund des Wachstumszwangs entstehen zunehmend Großkonzerne („Global Player“), die ihre Machtposition ausspielen und Konkurrenten aufkaufen.
- Wenn im Markt nur wenige Konkurrenten vorhanden sind, werden strategische Kooperationen eingegangen, deren Ausprägung zum Teil in Form von Kartellen zu erkennen sind, da dies noch effizienter ist.
- Durch verbesserte Standortbedingungen versuchen Staaten Unternehmen anzulocken, um die Bedingungen für Gewinnmaximierung zu verbessern. Dazu zählen z.B. Lohn-, Sozial-, Steuer- und Umweltdumping.
- Die Preisgestaltung orientiert sich an der Angebot- und Nachfragemacht und spiegelt die Interessen des Mächtigen wieder.
- Je globaler der freie Wettbewerb ist, desto größer ist das Machtgefälle zwischen den Marktbeteiligen und führt zu Ungleichheiten und einer Kluft zwischen Arm und Reich.
- Das primäre Ziel des Kapitalismus ist nicht Befriedigung der Grundbedürfnisse, sondern die Vergrößerung des Kapitals. Es werden strategisch neue Bedürfnisse geweckt, hinter denen eine höhere Kaufkraft steht.
- Der Umweltschutz wird vernachlässigt, da er nicht zu der Vermehrung des Kapitals beiträgt.
- Die Anhäufung von materiellen Werten rückt in den Vordergrund und unterwirft andere Werte wie z.B. Beziehungs- und Umweltqualität. Der Konsumzwang wird zur Kaufsucht.
- Die Wirtschaft wird geprägt von Egoismus, da sie diesen durch Konkurrenzverhalten (z.B. Karriere) belohnt. Dieser Egoismus färbt auf andere Bereiche wie Politik und Medien als auch auf unsere zwischenmenschlichen Beziehungen ab.
- Die Demokratie wird schrittweise ausgeschaltet, da Wirtschaftakteure durch Lobbying, Medienbesitz oder Parteifinanzierung ihrer Interessen durchsetzen.

1.1.1 Die Gemeinwohl-Ökonomie - der Kern

Ziel des Wirtschaftens

Bei der Beschreibung der Ziele und Ausrichtungen eines Wirtschaftsunternehmens fallen häufig Begriffe wie „Geld“, „Gewinn“ oder auch „Profit“. Tatsächlich wird konträr hierzu in diversen Verfassungen wörtlich

„Die gesamte wirtschaftliche Tätigkeit dient dem Gemeinwohl“ (Felber 2014, S. 32)
verlangt.

Durch die Gemeinwohl-Ökonomie soll das verfassungsmäßige Ziel in der Wirtschaft Einzug erhalten. Das zur Zeit höchste Ziel des finanziellen Wohlstandes soll dem des maximalen Beitrags zum Gemeinwohl weichen. Um dies zu erreichen muss ein Austausch des falschen Leitsterns „Eigennutzenmaximierung“ durch den Leitstern „Gemeinwohl“ erfolgen. Dies würde dazu führen, dass das Ziel aller Unternehmen die Maximierung des eigenen Allgemeinwohl-Beitrags zur Gesellschaft darstellt. Zusätzlich muss die Erfolgsmessung auch auf diesen neuen Leitstern angepasst werden. Eine Messung des Erfolgs eines Unternehmens alleine durch das messen finanzieller Faktoren des Unternehmens ist nicht ausreichend.

„Geld ist [...] nicht das Ziel des Wirtschaftens, sondern nur das Mittel.“ (Felber 2014, S. 33)

Da das zu erreichende Ziel in keinem Zusammenhang zur Mehrung von Kapital steht, besteht für ein Unternehmen auch kein Zwang sich auf diese Mehrung zu fokussieren. Der Erfolg eines Unternehmens muss direkt am Ziel - dem Erbringen von Nutzen für das Allgemeinwohl - gemessen werden.

Die Gesellschaft wie auch einzelne Menschen benötigen kein Geld sondern Nutzwerte (zum Beispiel Nahrung, Kleidung, Wohnung, Beziehungen, usw.). Geld ist nur das Tauschmittel um Nutzwerte zu erlangen. Die Finanzen eines Landes, eines Unternehmens oder einzelner Personen geben wenig Aufschluss über deren Nutzwert-Lage. Dies sieht man am Beispiel des Brutto-Inlands-Produktes - dieses gibt wenig Aufschluss über die politische Lage eines Landes oder die gerechte Verteilung der Güter innerhalb eines Landes. Das BIP muss einem aussagekräftigeren Mittel weichen. (vgl. Felber 2014, S. 34)

Die Suche nach einer entsprechenden Alternative wurde bereits 1970 begonnen. Hierzu gibt es Versuche verschiedener Institutionen und Länder wie dem "Better Live Index" des OECD oder den "W3-Indikatoren" - "Wachstum, Wohlstand, Lebensqualität" des deutschen Bundestages. Der Zwerghaat Butan hat mit dem "Bruttoinlands Glück" einen weniger auf mathematischen Modellen basierenden Ansatz gewählt. Hier werden jährlich alle 6000

Haushalte zu ihrer Gefühlslage befragt. Viele Ökonomen sind der Meinung, dass das Messen von "Glück" nicht möglich ist. Ein Messen des Glücks durch das Einbeziehen vieler Faktoren kommt dem tatsächlichen Glückswert allerdings wesentlich näher als die Darstellung über den alleinstehenden Faktor "Brutto-Inlands-Produkt". (vgl. Felber 2014, S. 35)

Gemeinwohl als Unternehmensbilanz messen

In der Gemeinwohl-Ökonomie wird als neue wichtigste Unternehmensbilanz die "Gemeinwohlbilanz" eingeführt. Die Finanzen eines Unternehmens erhalten einen geringeren Stellenwert. Dennoch gilt weiterhin - ein Unternehmen soll keine finanziellen Verluste machen, da hieran auch direkt die Möglichkeit der Erbringung des Produktes "Gemeinwohl" geknüpft ist. Vermieden werden sollen

"Gewinne um der Gewinne willen". (Felber 2014, S. 37)

Die Gemeinwohl-Bilanz gibt Aussage darüber, wie die fünf häufigsten Verfassungswerte demokratischer Staaten in Unternehmen umgesetzt werden: "Menschenwürde, Solidarität, Gerechtigkeit, ökologische Nachhaltigkeit und Demokratie" (Felber 2014, S. 37). Hierbei wird genauer die Erfüllung dieser Werte im Umgang mit den "Berührungsgruppen" (Intern & Extern) eines Unternehmens gemessen. Hierzu zählen Mitarbeiter und Zulieferer genau so wie zukünftige Generationen und die Umwelt.

Um die Gemeinwohl-Bilanz zu messen bzw. darzustellen werden für jede Berührungsgruppe derzeit 17 Gemeinwohl-Indikatoren aufgenommen:

Sinnhaftigkeit des Produktes, Arbeitsbedingungen, ökologische Produktion, ethischer Vertrieb der Produkte, Kooperation und Solidarität zu anderen Unternehmen, Verteilung der Erträge, Gleichbehandlung der Frauen und Männer und wie demokratisch Entscheidungen getroffen werden. (vgl. Felber 2014, S. 37,38)

Gemeinwohl definieren

Der Begriff Gemeinwohl kann nur durch einen demokratischen Entscheidungsprozess definiert werden. Eine Definition die durch die Regierung (Beispiel Diktatur) vorgegeben wird, kann zu einem Missbrauch des Begriffes selbst führen. In Diktaturen werden häufig Begriffe wie "Allgemeinwohl" oder auch "Frieden" zur Rechtfertigung der Politik eingesetzt.

"Theoretisch könnte ein Diktator oder ein totalitäres Regime behaupten, sie wüssten am besten, was für alle gut sei, und ihre Politik mit einem so verstandenen »Gemeinwohl« begründen." (Felber 2014, S. 39)

Universalbilanz, Markttransparenz und Gemeinwohl-Audit

Standards und Normen sind Richtlinien, zu denen kein Hersteller verpflichtet ist - dennoch halten sich viele Unternehmen an diese. Standards gibt es zu den verschiedensten Bereichen - auch in Bereichen die das Gemeinwohl betreffen (z.B. "Biolandbau, "Fairer Handel", usw.). Bei der Einführung von Gemeinwohl-Bilanzen auf eben dieser Freiwilligenbasis wie es bei Normen und Standards der Fall ist ergibt sich folgendes Problem:

"Sobald sie in Widerstreit mit der Hauptbilanz – der Finanzbilanz – geraten, sind sie plötzlich nichts mehr wert, denn das würde den Lebensnerv des Unternehmens angreifen und in der heutigen Systemdynamik schädigen: Wer zugunsten einer unverbindlichen Nebenbilanz den Finanzgewinn schmälert, katapultiert sich selbst aus dem Rennen." (Felber 2014, S. 41)

Eine Gemeinwohl-Bilanz kann nicht auf Freiwilligenbasis umgesetzt werden. Sie muss folgende Kriterien erfüllen:

Verbindlichkeit, Ganzheitlichkeit, Messbarkeit, Vergleichbarkeit, Verständlichkeit, Öffentlichkeit, externe Prüfung, Rechtsfolgen (vgl. Felber 2014, S. 44)

Jedes Unternehmen muss eine Gemeinwohl-Bilanz aufstellen. Innerhalb dieser Bilanz kann ein Unternehmen eine Punktestufe erreichen. Das Ergebnis dieser Bilanz könnte dann an den Produkten des Unternehmens farblich neben einem QR-Code markiert werden. Über den QR-Code kann der Käufer sich genauer über die Bilanz erkundigen - so hätten auch die Konsumenten direkte Einsicht in die Gemeinwohl-Bilanz des Unternehmens.

Die Gemeinwohlabilanz soll in erster Version durch das Unternehmen selbst erstellt werden. Hierbei werden optimaler Weise alle Mitarbeiter und Begleitunternehmen, Zulieferer und sonstige Schnittstellengruppen befragt. Anschließend erfolgt eine Prüfung durch externe AuditorInnen. Die Kritik, dass dies zu einem schiefen Bild führen wird, Unternehmen sich besser darstellen oder AuditorInnen bestechen werden, lässt sich genau so auf die aktuelle Situation anwenden. Unternehmen erstellen ihre Finanz-Bilanzen selbst. Diese werden durch externe Audits geprüft. Außerdem erfolgt eine stichprobenartige Kontrolle durch den Staat - diese wird es für die Gemeinwohl-Bilanz ebenfalls geben. Bestechungen bzw. Täuschungen und deren Versuche fließen in die Gemeinwohl-Bilanz mit ein.

Gemeinwohlstreben belohnen

Um das Gemeinwohlstreben der Unternehmen zu steigern wird ein Belohnungssystem eingeführt. Diejenigen Unternehmen, die dem Gemeinwohl nützen, werden mit Steuernachlässen oder verringerten Zollgebühren belohnt. Umgekehrt genauso - wer dem Gemeinwohl schadet zahlt einen erhöhten Steuersatz, muss mehr Zollgebühren zahlen, usw. . Ein Ausgleich der Produkt- und Unternehmenswelt findet statt. Die Unternehmen sind gezwungen gemeinnützig zu Arbeiten, gemeinnützige Produkte herzustellen, gute Arbeitsbedingungen zu schaffen und Gleichberechtigung der Mitarbeiter einzuführen.

Gewinn als Mittel

Natürlich erwirtschaften Unternehmen auch innerhalb der Gemeinwohl-Ökonomie einen Gewinn. Dieser soll allerdings nicht zu hoch sein und dessen Verwendung unterliegt gewissen Vorschriften. Gewinne können dem Gemeinwohl schaden, wie auch das Gemeinwohl mehren. Aus diesem Grund wird bei der Verwendung von Gewinnen zwischen erlaubten und nicht erlaubten Verwendungen unterschieden. Um diese im Vorfeld zu erkennen wird bei jeder größeren Investition eine Investitionsanalyse erstellt, die den "Gemeinwohlfaktor" dieser Investition darstellt.

Erlaubte Verwendungen stellen zum Beispiel Investitionen in Produkte dar, die einen hohen "Gemeinwohl-Faktor" haben (Zum Beispiel erneuerbare Energien im Gegensatz zu umweltunfreundlichen). Eine weitere Möglichkeit stellt das Rückzahlen von Krediten oder das Ausschütten der Gewinne an Mitarbeiter dar. Denkbar wäre ebenfalls das zur Verfügung stellen von zinslosen Darlehen an Mitbewerber, Zulieferer, etc. .

Nicht erlaubte Verwendungen von Gewinnen stellen jene Nutzungen dar, die dem Gemeinwohl schaden - also feindliche Übernahmen oder Finanzinvestments. Unternehmen sollen durch das erbringen ihrer Dienstleistungen oder das Verkaufen ihrer Produkte die Werte schaffen, nicht durch Finanzgeschäfte. Eine weitere nicht erlaubte Verwendung von Gewinnen ist die Ausschüttung an nicht mitarbeitende Gesellschafter.

"Durch die Trennung von entscheidungsmächtigen EigentümerInnen und Beschäftigten im Unternehmen wird Verantwortungslosigkeit bis hin zur totalen Skrupellosigkeit enthemmt" (Felber 2014, S. 54)

Parteispenden durch Unternehmen werden komplett verboten.

Ende des Wachstumszwangs

In den Erfolg eines Unternehmens spielen viele Faktoren ein wie Produktqualität, Innovationskraft, Effizienz, Größe, Flexibilität - wirklich entscheidend ist zur Zeit aber nur der Finanzgewinn. Das führt dazu, dass Unternehmen teilweise dazu gezwungen sind skrupellos, unethisch zu handeln. Es existiert ein Wachstumsdruck: keinen Finanzgewinn zu erzeugen bedeutet nicht erfolgreich zu sein. Wenn der Finanzgewinn nicht der einzige auschlaggebende Faktor wäre, könnten Unternehmen gelassen ihre "optimale Größe" ermitteln, würden keinem Wachstumsdruck mehr unterliegen.

Wachstumsdruck ist ein Faktor, der dazu führt, dass Unternehmen weniger für das Allgemeinwohl handeln. Die Quantität steigt auf Kosten der Qualität. Ein Mensch wächst physisch auch nur bis zu einem gewissen Alter, anschließend wachsen nur noch die inneren Werte - Charakter, der Umgang mit Menschen und Problemen, die fachbezogenen Fähigkeiten und emotionalen Kompetenzen. (vgl. Felber 2014, S. 61)

Das Ende des Wachstumsdrucks für Unternehmen würde zu besseren Arbeitsbedingungen, einer besseren Unternehmensausrichtung und besseren Produkten führen.

Strukturelle Kooperation

Der Übergang vom Gegeneinander zum Miteinander stellt eine der größten zu überwindenden Hürden dar. Zur Zeit gibt es verschiedenste Unternehmen die in den gleichen Branchen nach Lösungen für die gleichen Probleme suchen - gegeneinander.

"Lieg es nicht auf der Hand, dass »gegeneinander suchen« nicht effizient sein kann?"
(Felber 2014, S 62)

In der Gemeinwohl-Ökonomie wird das Konzept der Konkurrenz nicht verboten oder abgeschafft. Unternehmen dürfen weiterhin mit unterschiedlichsten Zielen und Produktideen gegründet werden. Es gibt generell die Möglichkeit, sich gegen die Gesellschaft oder das Gemeinwohl zu stellen - der Unterschied: Es bietet keine Vorteile mehr.

Das Zusammenarbeiten im Sinne einer friedlichen Koexistenz wird gefördert. Im kapitalistischen Denkmuster erinnert dies an ein Kartell. Doch ein Kartell als solches bringt den Unternehmen in der Gemeinwohl-Ökonomie keinen Mehrwert.

"Heute sind Kartelle kein Selbstzweck, sondern ein Mittel, um den Gewinn zu steigern. Wenn Gewinne begrenzt und als Mittel für die Mehrung des Gemeinwohls eingesetzt werden, dann verliert auch Kartellbildung als Mittel dazu ihren Sinn." (Felber 2014, S. 63)

Konkurs

Die Möglichkeit des Konkurses besteht weiterhin - allerdings ist ihre Eintrittswahrscheinlichkeit aus verschiedenen Gründen weniger wahrscheinlich. Die Gemeinwohlwirtschaft führt automatisch dazu, dass üblicherweise nur noch sinnvolle Unternehmen gegründet werden. Profit alleine ist kein Grund zur Unternehmensgründung. Des Weiteren gibt es statt einer feindlichen Konkurrenz eine friedliche Koexistenz der Unternehmen. Zu guter Letzt sind die Mitarbeiter eines "demokratisch geführten" Unternehmens eher motiviert und ziehen an einem Strang, sodass ein Konkurs nicht eintritt oder effektiv verhindert werden kann. (Vgl. Felber 2014, S. 64)

Kooperative Marktsteuerung

Kommt es doch so weit, dass ein Unternehmen oder ein ganzer Markt dem Konkurs unterliegt, gelten wieder die Ziele des Gemeinwohls. Unternehmen eines dem Konkurs unterliegenden Marktes können einen "Krisen- oder Kooperationsausschuss" einberufen, durch den gemeinsam das weiter Vorgehen und die sinnvollste Reaktion auf die zu überwindenden Problem erarbeitet wird. Es könnte ein koordiniertes Verkürzen der Arbeitszeiten in allen betroffenen Unternehmen erfolgen, ein Arbeitsplatzabbau mit

entsprechenden Umschulungen der zu entlassenen Mitarbeiter, die Umspezialisierung weniger Unternehmen auf neue Themengebiete um den betroffenen Markt wieder zu öffnen oder, als letzte Möglichkeit, das Schließen von Betrieben mit einer Umschulung der Mitarbeiter auf Märkte mit Arbeitskräfte-Mangel. (Vgl. Felber 2014, S. 65)

Gemeinwohl und Globalisierung

Ein häufiger Kritikpunkt an der Gemeinwohl-Ökonomie ist, dass deren Einführung laut Kritikern das Zusammenspiel der gesamten Welt bedarf. Tatsächlich ist es so, dass im Konkurrenzkampf zweier Unternehmen der Unethischere / Skrupellosere gewinnt - also im globalen Kontext: Ein Land mit einer Gemeinwohl-Ökonomie hat geringe Chancen gegen Länder mit der Freihandels-Ökonomie. Genau das ist auch der Fehler im System. Das unethische und skrupellose Verhalten wird belohnt und damit weiter ausgebaut. Das direkte Einführen der Gemeinwohl-Ökonomie weltweit ist allerdings auch unrealistisch.

Zur Lösung werden zwei Vorschläge geliefert (vgl. Felber 2014, S. 67):

1. Der globale ordnungspolitische Ansatz: Es werden gemeinsame Rahmenbedingungen bestimmt (Arbeitsschutz, Sozialschutz, Umweltschutz, etc.). Die UNO wird hier als Ort der Regulierung vorgeschlagen. Die Rahmenbedingungen werden zur Umsetzung der Gemeinwohl-Ökonomie in einer ersten Länder-Zone (z.B. der EU) eingehalten. Für alle Länder die mit einem Land der Gemeinwohl-Zone handeln wollen, werden bei nicht Erfüllung einer der Standards (Arbeitsschutz, Sozialschutz, etc.) erhöhte Zollgebühren verlangt. Hier entsteht eine Staffelung: jede nicht eingehaltene Rahmenbedingung erhöht die Zollgebühren weiter.
2. Der anreizpolitische Ansatz der Gemeinwohl-Ökonomie: Die Erstellung und Offenlegung einer Gemeinwohlbilanz für Unternehmen wird verpflichtet. Eine bessere Bilanz führt zu einem "freieren" Marktzugang, eine schlechtere zu einem schlechten Marktzugang (Erhöhen des "ethischen Schutzzolls").

Soziale Sicherheit, Freijahre, Solidaritätseinkommen und Rente

In einer Wirtschaft in der Unternehmen dem Konkurs unterliegen können, gibt es auch die Möglichkeit der Arbeitslosigkeit. In der Gemeinwohl-Ökonomie wird pro Arbeitnehmer und Pro Dekade Berufstätigkeit ein "Freijahr" zur eigenen Verfügung eingeführt. Der Arbeitnehmer erhält in dieser Zeit den gesetzlichen Mindestlohn. Da dieses Jahr für jede Person gleichermaßen gilt, ist eine Diskussion über Ungleichbehandlung oder die steuerliche Finanzierung überflüssig.

Die Freijahre hätten den Effekt, dass die Arbeitslosigkeit gesenkt werden würde und jede Person sich innerhalb des Jahres anderen Themen wie der "Weiterbildung, der Familie [...] oder anderen Passionen" (Felber 2014, S. 69) widmen könnte. Das Arbeitsklima würde hierdurch entzerrt und die Mitarbeiter motivierter.

"Aufgrund dieser geänderten Umstände erscheint es mir »systemwidrig«, dass Leistungen wie Arbeitslosen-, Notstands-, Sozialhilfe oder Hartz IV noch nötig sein werden." (Felber 2014, S. 69)

Ein Konzept zum Umgang mit Arbeitslosen muss dennoch angedacht werden. In welcher Form eine Hilfe umgesetzt wird muss noch erdacht werden. Möglich wäre ein Teil des Mindestlohns als Sozialhilfe - auch ein Bedingungsloses Grundeinkommen wäre im Sinne der Gemeinwohl-Ökonomie möglich.

"Für Menschen mit besonderen Bedürfnissen oder Einschränkungen, die sich nicht oder nur teilweise an der Erwerbsarbeit beteiligen können, soll es jedenfalls ein bedingungsloses Solidaritätseinkommen geben: zum Beispiel in der Höhe des Durchschnittseinkommens" (Felber 2014, S. 69)

Als Rente wird das derzeitige System und dessen Kopplung zu den Finanzmärkten abgeschafft.

"Die Privatisierung der Renten macht diese weder sicherer noch sozialer, noch billiger – in allen drei Kriterien tritt das Gegenteil ein."

Aus diesem Grund wird das ehemalige System des Generationenvertrages wieder eingesetzt. Die Rentensicherung stellte vor der Umstellung nie ein wirkliches Problem dar. Die Annahme, die Kopplung der Rente an die Finanzmärkte würde diese sichern ist ein Irrglaube, von dem die private Versicherungswirtschaft eigene Gewinne erwirtschaftet. Doch Gewinne und Profite sind in der Gemeinwohl-Wirtschaft keines der Ziele. Banken und Versicherungen, das gesamte Finanzsystem wird zu einem Mittel, das Gemeinwohl zu erreichen und nicht Profit zu schlagen.

1.1.2 Die Demokratische Bank

Durch die Liberalisierung und Globalisierung der Finanzmärkte entstehen zunehmend Global-Player-Banken, die nicht der Gesellschaft und dem Gemeinwohl dienen, sondern das primäre Ziel von Profit verfolgen. Die Gemeinwohl-Ökonomie benötigt daher ein anderes Finanzsystem. Die Hauptaufgabe der Banken war die Umwandlung von Spargeldern in zugängliche Kredite für regionale Unternehmen und Privathaushalte. Dieser Aufgabe kommen Sie jedoch teilweise oder unzureichend nach.

Der liberalisierte Markt fördert die globale wettbewerbsfähige Größe von Banken. Dies ist jedoch das explizite Ziel des EU-Finanzbinnenmarktes und des Weltmarktes für Finanzdienstleistungen durch die WTO. Die Banken werden dadurch ökonomisch und politisch systemrelevant. Durch die steigende Macht können sie sich zusätzlich gegen Zerteilung, Regulierungen und Besteuerung wehren.

Bei einer Gemeinwohl-Ökonomie würden in einem Finanzsystem Geld in Form von Krediten dem öffentlichen Zwecke dienen und die Finanzmärkte geschlossen werden.

Ziel und Leistungen

Die Demokratische Bank verfolgt die Ziele und Werte der Gemeinwohl-Ökonomie. Sie ist daher nicht gewinnorientiert und fördert insbesondere regionale Wirtschaftskreisläufe. Zu ihren Kernleistungen zählen u.A. die „unbeschränkte Garantie der Spareinlagen“, „kostenlose Girokonten für alle WohnsitzbürgerInnen“, ein „flächendeckendes Filialnetz mit wertschätzender persönlicher Betreuung“, „kostenlose Ergänzungskredite“ und der „Wechsel von Währungen“ (Deuticke 2014, S. 74). Die Leistungen und Ziele der demokratischen Bank sind in der Verfassung niedergeschrieben und können nur per Volksabstimmung geändert werden.

Transparenz und Sicherheit

Die Demokratische Bank nimmt lediglich die Rolle des Geldvermittlers zwischen SparerInnen und KreditnehmerInnen ein. Zusätzlich muss sie alle Geschäfte in der Bankbilanz vermerken und gesetzliche Eigenkapitalvorschriften einhalten.

Finanzierung, Refinanzierung, Konkurs

Die Mitarbeiter der Demokratischen Bank „genießen hohe soziale Sicherheit und umfassende Mitbestimmungsrechte“ und erhalten ein „menschenwürdiges Einkommen“ (Deuticke 2014, S. 75). Kreditvergabe der Bank erfolgt aus den Einlagen von Privatpersonen, Unternehmen und Staat. Diese Finanzvermögen wachsen in Relation mit der realen Wirtschaftsleistung (BIP), sodass ein ausreichendes Kreditkapitel für Refinanzierungen zur Verfügung steht. Im Fall, dass die Spareinlagen in einer Gemeinde, Region oder Bundesland nicht ausreichen, um die Kreditanfragen zu decken, verteilen

andere Banken diese um. Dabei haftet die Zentralbank für dieses Umverteilungsrisiko. Die Wahrscheinlichkeit des Konkurses einer Zweigstelle der Demokratischen Bank ist gering, da sie ähnlich wie andere staatliche Einrichtungen (z.B. Schulen, Krankenhäuser), welche auch nicht Konkurs gehen können, behandelt wird.

Zinsen und Inflation

KreditnehmerInnen zahlen eine Kreditgebühr, die sich nach den entstehenden Kosten der Bank durch die Kreditvergabe orientiert und der Bank keinen Gewinn bringt. Kredit- und Sparzinsen existieren nicht mehr. Die Möglichkeit der Inflation wird verringert, da es kein Wachstumszwang in der Gemeinwohl-Ökonomie gibt.

Soziale und ökologische Kreditprüfung

Die Kreditvergabe wird beeinflusst durch die Kenntnis der lokalen Situation und der Wirtschaftsakteure. Die Kredite werden nicht mehr nach ökonomischer Rentabilität vergeben, sondern nach sozialen und ökologischen Mehrwert. Die Investitionen, die einen besonders hohen sozialen und ökologischen Mehrwert bringen, werden durch kostenlose Kredite oder teilweise Kredite mit negativen Zinssatz unterstützt. Auf der anderen Seite erhalten Investitionsvorhaben, die einen sozialen oder ökologischen Minderwert schaffen, keinen Kredit.

Ökosoziales Risikokapital und Gemeinwohl-Börsen

Am Risikokapitalmarkt (z.B. Börse) herrscht die Hoffnung, dass Projekte mit ungewisser Rentabilität finanziert werden. Diese Projekte schaffen jedoch oft keinen sozialen und ökologischen Mehrwert. Die Demokratische Bank könnte dieses Prinzip jedoch aufgreifen und eine Risiko-Abteilung schaffen, die Innovationen mit sozialen und ökologischen Mehrwert finanziert. Dazu wird ein kleiner Prozentsatz der Spareinlagen als ökosoziales Risikokapital bereitgestellt. Eine andere Möglichkeit ist die Einrichtung von regionalen Gemeinwohl-Börsen, die von den Banken einer Region getragen werden könnten. An der Börse können Kreditanfragen, die zwar die finanzielle Bonitätsprüfung nicht bestehen, jedoch einen sozialen und ökologischen Mehrwert schaffen durch Unternehmen unterstützt werden.

Verhältnis zu Privatbanken

Für einen fairen Wettbewerb müssen Privatbanken in nicht gewinnorientierte Rechtsformen umgewandelt werden und das Investmentbanking abgeschafft werden. Dieser Übergang kann erzielt werden, indem gemeinwohlorientierte Banken staatliche Unterstützungsleistungen erhalten, denen man gewinnorientierte Banken verweigert.

Zentralbank

Die Zentralbank ist Teil des Demokratischen Bankensystems und transparent und demokratisch organisiert. Der Leitungskreis umfasst VertreterInnen aller Gesellschaftsbereiche. Die Zentralbank besitzt das Geldschöpfungsmonopol und stellt dem Staat Geld in begrenztem Maß zur Verfügung. Sie finanziert den Staat über unverzinste Kredite oder durch „Erweiterung der Geldmenge als Geschenk an den Staatshaushalt“ (Deuticke 2014, S. 82).

Die Zentralbank ist an einer globalen Währungskooperation beteiligt. Die Kooperation umfasst die Schaffung einer neutralen Verrechnungseinheit für den internationalen Handel (z.B. „Globo“ oder „Terra“). Die nationalen Währungen bestehen weiterhin. Der „Globo“ stellt daher eine Komplementärwährung auf internationaler Ebene dar. Derartige Komplementärwährungen kann es auch auf regionaler Ebene geben. Die Entscheidung für weitere regionale Währungen tragen dabei die Demokratischen Banken.

1.1.3 Eigentum

Derzeit ist das Eigentum ungleich verteilt, sodass einzelne Personen und Unternehmen Medien und Politik kontrollieren können. Das Privateigentum gefährdet die Demokratie und die Freiheit und Gleichheit der Menschen. Die Gemeinwohl-Ökonomie möchte diese Problematiken umgehen, sodass eine ethische und trotzdem liberale Marktwirtschaft entsteht.

Negative Rückkopplung

Der Kapitalismus ist ein positiv rückgekoppeltes System, sodass die reiche Elite immer reicher wird. Diese Rückkopplung der Bildung einer finanziellen Elite könnte durch verschiedene Maßnahmen unterbunden werden. Es könnte die Einkommensungleichheit, das Privatvermögen und das Unternehmensvermögen in privater Hand begrenzt werden. Des Weiteren besteht die Möglichkeit das Erbrecht neuzugestalten und die mögliche Erbmasse zu limitieren.

Relative Begrenzung der Einkommensungleichheit

Eine sehr starke Einkommensungleichheit wird weder Leistung noch Verantwortung fördern. Die Reichen werden hierdurch nicht glücklich, sondern gierig und die Armen fühlen sich minderwertig. Zudem wird Aggression und Kriminalität ansteigen, Stress und Krankheiten werden gefördert. In der Gemeinwohl-Ökonomie soll ein demokratischer Wirtschaftskonvent mögliche Grenzen einer Einkommensungleichheit erarbeiten. Hierbei wird nur das Arbeitseinkommen und die Einnahmen aus Vermietung betrachtet, da es kein Kapitaleinkommen mehr gibt. Hierbei wird auch ein Mindestlohn definiert, der ein menschenwürdiges Leben ermöglichen soll.

Begrenzung der Privatvermögen

Der Wirtschaftskonvent soll nicht nur die Begrenzung des Einkommens, sondern auch eine Obergrenze des Privatvermögens festlegen.

Demokratisierung von Großunternehmen

Global agierende Unternehmen haben heutzutage mehr Einfluss als manche Regierungen. Die Unternehmen werden von einer kleinen Zahl von Privatpersonen gelenkt, ohne demokratischen Einfluss. Unternehmen sollen anhand ihrer Größe eine Demokratisierung erfahren. So könnten z.B. kleine Unternehmen bis 250 Beschäftigten weiterhin in privater Hand bleiben. Ab 250 Beschäftigten erhalten die Mitarbeiter 25% der Stimmrechte. Bei steigender Größe der Belegschaft wird das Unternehmen weitergehend demokratisiert. Zudem könnte große Unternehmen durch ein regierungsunabhängiges regionales Wirtschaftsparlament geführt werden, welches direktdemokratisch gewählt wird.

Mitarbeiterbeteiligung

Es wird angestrebt, dass möglichst viele Menschen Eigentum an den Unternehmen gelangen und so die Unternehmen steuern können und zwangsläufig das Verlustrisiko gemeinsam tragen. Z.B. können den Mitarbeitern Mitarbeiterfonds übertragen werden.

Gewinnbindung an das Unternehmen

Die Gewinnausschüttung des Unternehmens an den Gründer soll mit steigenden Beitrag anderer Menschen am Unternehmenserfolg abnehmen. Somit wird nach einiger Zeit jeder Gewinn in Investitionen fließen müssen.

Begrenzung des Erbrechts, Generationenfonds und demokratische Mitgift

Das uneingeschränkte Erbrecht führte dazu, dass das Startkapital in privater Hand sehr ungleich verteilt war. Die Gemeinwohl-Ökonomie sieht eine Erbmasse bis zu einem bestimmten Betrag als steuerfrei an. Darüber hinaus soll die Erbmasse in einen öffentlichen Generationenfonds eingespeist werden und zu gleichen Teilen an die nachfolgenden Generationen verteilt werden. Somit wird ein großes konzentriertes Vermögen zerkleinert auf die breite Masse.

Immobilien

Immobilien können innerhalb dieses Freibetrages auch vererbt werden. Wenn die Immobilie einen höheren Wert hat, so könne man diese mit mehreren Erben weiterhin besitzen, ohne dass die Immobilie dem Generationenfonds zugeführt wird.

Vererbung von Unternehmen

In der Bundesrepublik Deutschland können derzeit Unternehmen steuerfrei an die Erben vererbt werden, ungeachtet derer Qualifikation. In der Gemeinwohl-Ökonomie wird es vermieden, dass riesige Unternehmen in einzelnen privaten Händen gehalten werden, sondern auf die breite Masse verteilt werden. So könnten z.B. bei Aktiengesellschaften Anteile bis zu der definierten Größe weiter vererbt werden, darüber hinaus werden die Anteile dem Generationenfonds zugeführt. Bei Familienunternehmen wird dieser mögliche Erbbetrag deutlich angehoben auf z.B. zehn Millionen Euro. Die weiteren Firmenanteile können an die Belegschaft verteilt werden und an ausgewählte Nichtfamilienmitglieder bis zu einer Obergrenze verteilt werden. Die Erben müssten sich verpflichten z.B. mindestens drei Jahre in dem Betrieb zu arbeiten.

Dem Menschen werden heute nur zwei Möglichkeiten geboten. Zum einen der Verkauf der eigenen Arbeitskraft (strukturelle Sklaverei) und zum anderen ein eigenes erfolgreiches Unternehmen zu gründen. Im Kapitalismus verdient der Kapitalgeber an dem Mehrwert der Arbeitsleistung des Arbeitnehmers. Somit ist dies ein unfreies und ausbeuterisches System. Gerechter wäre es, wenn alle Arbeitenden und Geldgebenden betriebliche

Entscheidungsgewalt beherrschen. Die Erträge unter allen Arbeitenden aufgeteilt werden, die Firmenanteile an möglichst viele verteilt werden. Somit tragen alle das unternehmerische Risiko und müssen Verantwortung übernehmen. Es müssen aber nicht zwangsläufig alle Arbeitenden diese Verantwortung übernehmen, aber die Möglichkeit sollte bestehen.

Schenkung

Es wird in der Gemeinwohl-Ökonomie einen Schenkungsfreibetrag eingeführt, ähnlich dem Erbfreibetrag, bei dem bis zu dem festgelegten Betrag Finanzmittel übertragen werden können.

Demokratische Allmenden

Die dritte Kategorie des Eigentums in der Gemeinwohl-Ökonomie ist das öffentliche Gemeinschaftseigentum. Dies könnten z.B. die Bahn, die Post, Bildungseinrichtungen, Stadtwerke, Kindergärten und Banken sein. Dieses Eigentum könnte durch ein direkt gewähltes Leistungsgremium aus Vertretern des Staates, der Belegschaft, der Nutzer, Gender-Beauftragten und einem Zukunftsanwalt verwaltet werden. Klassische Staatsunternehmen, welche ausschließlich durch den Staat kontrolliert werden, wird es somit nicht mehr geben. Das Daseinsvorsorgekonvent definiert das öffentliche Gemeinschaftseigentum und bestimmt die Spielregeln für die Organisation.

Eigentum an Natur

In der Gemeinwohl-Ökonomie soll der Bürger ein Eigentum an Grund und Boden besitzen. Die Bürger, welche es für konkrete Nutzungszwecke benötigt, wird ein Anteil kostenlos zur Bewirtschaftung überlassen. Demzufolge entfällt die Grundsteuer für den primären Sektor. Bei einer besonders ökologischen Nutzung der Fläche wird dem Bauern mehr Fläche zugeteilt werden.

Für den privaten Wohnraum könnte ein Anteil an die einzelnen Parteien durch die Gemeinden zugeteilt werden. Jedem Mensch steht grundsätzlich das Recht auf Wohnfläche zu bis zu einer Obergrenze an qm. Diese Fläche wird dann zu einem festgelegten qm-Preis erworben.

Eine extreme Ungleichverteilung an Immobilien wird hiermit umgangen.

Freiheit und Gleichheit

Das Eigentumsrecht wird auf ein für ein gutes Lebens nötiges Eigentum begrenzt, da sonst die Freiheiten anderer Menschen gefährdet werden.

1.1.4 Motivation und Sinn

Motivation

Bedenken hinsichtlich fehlender Motivation der einzelnen Marktteilnehmer in der Wirtschaft sollen mit den folgenden Punkten beseitigt werden:

1. In der Gemeinwohl-Ökonomie besteht weiterhin ein Erwerbszwang und die durchschnittliche Arbeitszeit soll reduziert werden. Zusätzlich wird es für nicht Erwerbstätige ein Solidaritätseinkommen geben, welches ein Überleben in Würde sichern soll. Zudem gibt es vier Freijahre.
2. Das maximale Einkommen in privaten Unternehmen wird gedeckelt und an den Mindestlohn gebunden. Mit welchem Hebel wird von dem Souverän beschlossen.
3. Höhere Einkommen als 20000 US-Dollar machen nicht mehr glücklicher.
4. Forschungen zeigen, dass Menschen durch das Streben nach Autonomie, Identität, Kompetenz, Beitrag, Gemeinschaft und Beziehung stärker motiviert werden als durch das Einkommen.
5. Autonomie. Menschen, die frei in ihren Gefühlen, Bedürfnissen und Gedanken sind.
6. Identität. Es ist nicht wichtig besser zu sein als andere, sondern anders.
7. Kompetenz. Kompetenzen lassen sich genauso in kooperativen Strukturen entwickeln, wie auch in Konkurrenzstrukturen.
8. Beitrag. Jeder Mensch will und kann seinen Beitrag leisten.
9. Gemeinschaft. Jeder Mensch sehnt sich nach einer Gemeinschaft und erfährt dort Geborgenheit, Sicherheit, Wertschätzung und Anerkennung.
10. Beziehung. Menschen sind am glücklichsten in Momenten von Verbundenheit mit sich selbst, mit anderen Mensch, mit der Natur oder mit dem großen Ganzen.

Diese Punkte könnten in der Wirtschaft als Ziel bestimmt werden.

Das Kapital und die Stimmrechte haben in der Gemeinwohl – Ökonomie eine Mehrzahl an Menschen inne, sodass eine höhere Risikobereitschaft der Unternehmen gewährleistet ist.

Sinn

In der Gemeinwohl-Ökonomie ist der Sinn des Arbeitens eine Bedürfnisbefriedigung, Stärkung des Gemeinwohls und sinnvolle Arbeit und nicht primär das Einkommen.

Intrinsische Motivation ist zudem viel stärker als extrinsische. Die meisten Kinder werden zur Leistungserbringung durch extrinsische Motivation erzogen. Dieses Erlernte setzen sie später auch im Berufsleben um und substituieren die elterliche Liebe durch Geld. Menschen sollen mehr auf ein gutes Zusammenleben achten und die Gefühle und Bedürfnisse anderer wahren.

Erziehung und Bildung

Durch das Umdenken der Bevölkerung auf das Gemeinwohl, sind die folgenden Unterrichtsfächer viel relevanter für den Schulbetrieb, als die derzeit Üblichen.

1. Gefühlskunde. Hier wird gelehrt wie Kinder ihre eigenen Gefühle wahrzunehmen haben, sowie die Kommunikation dieser.
2. Wertekunde. Unbewusste Wertvorstellungen der Kinder werden aufgezeigt. Außerdem werden die Auswirkungen eines konkurrierenden oder kooperativen Handelns der Kinder zueinander aufgezeigt.
3. Kommunikationskunde. In der Kommunikation wird eine sachliche Diskussion geschult, bei der auf persönliche Werte und Beleidigungen verzichtet wird.
4. Demokratiekunde. Hier werden demokratische Prozesse vermittelt und das Grundkonzept einer Demokratie aufgezeigt,
5. Naturerfahrenskunde. Hier werden nicht nur biologische Sachverhalte gelehrt, sondern auch eine gute Beziehung zur Natur. Durch diese Beziehung wird die Lebensqualität gesteigert.
6. Kunsthandwerk. Die Kinder sollen selber kleine Kunsthandsarbeiten erledigen, da dieses glücklich macht und außerdem die Befassung mit Materialien und Werkzeugen zu einem ganzheitlichen Leben dazu gehört.
7. Körpersensibilisierung. Dem Kindern wird gelehrt auf den eigenen Körper zu hören und ihn zu achten.

1.1.5 Weiterentwicklung der Demokratie

Die gewählten Repräsentanten der Demokratie entfernen sich weiter von den Bürgerinteressen, deren Stimme sie erhalten haben. Diese Distanz wird mit folgend erklärt.

1. Die Wahlversprechen aus dem Parteiprogramm werden meist gebrochen und der Stimmberchtige hat kein Recht auf Erfüllung der Versprechen.
2. Die Führungsriege mancher Konzerne ist zu sehr in politische Prozesse involviert. Des Weiteren werden Politiker häufig in Management Positionen eingesetzt.
3. Das Management der Betriebe und Politiker haben einen starken Einfluss auf die Medien
4. Die Bildungseinrichtungen sind abhängig von dem finanziellen Hahn der Wirtschaft.
5. Thinktanks arbeiten für den Auftraggeber, von dem sie auch eine Bezahlung beziehen.
6. Parteien werden durch Geldflüsse gelenkt

Die demokratischen Prozesse müssen offener gestaltet werden, sodass eine Vielzahl der Menschen daran teilhaben kann und sie mitgestalten können. Die gemeine Bevölkerung benötigt mehr Mitspracherechte und der Lobbyismus muss bekämpft werden und auch die Medien müssen ihre Unabhängigkeit zurück erlangen.

Gewaltenteilung & Souverän

Die Macht darf im Staate nicht konzentriert sein, sodass diese geteilt werden muss – Gewaltenteilung. Außerdem muss dem Volke eine Möglichkeit innerhalb der Legislaturperiode gegeben werden, um die gewählte Vertretung ggf. korrigieren zu können.

Dreistufige direkte Demokratie

Erste Stufe: Das Volk kann für ein gewünschtes Gesetz Unterstützungserklärungen sammeln

Zweite Stufe: Wenn eine bestimmte Anzahl an Unterstützungserklärungen gesammelt wurde, wird ein Volksbegehren eingeleitet.

Dritte Stufe: Volksabstimmung mit rechtlich bindendem Gesetz, wie in der Schweizerischen Eidgenossenschaft.

Die direkte Demokratie birgt auch Nachteile, da das Volk nicht in jedem Sachverhalt so aufgeklärt ist wie die parlamentarischen Vertreter. Doch es gibt auch viele Vorteile. Die direkte Demokratie soll die alte Form nicht ergänzen, sondern erweitern. Das Volk kann einzelne Sachfragen herausgreifen, selbstständig entscheiden und nicht auf die Erfüllung der Wahlversprechen hoffen. Für ethische Entscheidungen können alle Menschen eine hohe Kompetenz aufweisen. Wenn die Bevölkerung in verschiedenen Fragen auch berechtigt ist eine Entscheidung zu treffen, wird das Interesse auch an den Sachverhalten ansteigen.

Dennoch müsste die Macht der Medien wie z.B. der populistischen „Bild Zeitung“ reglementiert werden. Zudem sind viele politische Vertreter nicht qualifizierter als die Vertretenden.

Trennung von verfassungsgebender und verfasster Gewalt

Es muss nicht nur eine Gewaltenteilung im Staate herrschen, sondern die Gewalt muss auch geteilt werden, welche die Verfassung schreibt und die Verfassung einsetzt. Z.B. wurde die EU geründet von den Regierungen der Mitgliedsstaaten, welche auch die Verfassungen schrieben. Dies ist somit ein Bruch der zweiten Gewaltenteilung.

Wirtschaftskonvent

Der demokratische Wirtschaftskonvent hat die Aufgabe, die zuvor in der Verfassung definierten Ziele in Gesetze zu verwandeln.

Bildungskonvent

Ein demokratischer Bildungskonvent wird von den betroffenen (Schüler, Studenten, Lehrer etc.) gewählt und ist dafür zuständig, Ziele und Inhalte der Bildungsweges der Betroffenen festzulegen.

Daseinsvorsorgekonvent

Der Daseinsvorsorgekonvent entscheidet welche Betriebe z.B. Schulen, Bahn, etc. in der Hand der Öffentlichkeit verbleibt bzw. neu dazu kommt.

Medienkonvent

Ein Medienkonvent stellt die Unabhängigkeit der Medien sicher.

Demokratiekonvent

Der Demokratiekonvent ist für die Ausarbeitung einer neuen Strategie zur Erfüllung einer wirklich demokratischen Gesellschaft verantwortlich.

1.1.6 Beispiele, Verwandte und Vorbilder

Dass die Gemeinwohl-Ökonomie keine Utopie ist, belegen viele Unternehmen, die das Prinzip der Kooperation verfolgen. Vielen Unternehmen ist gemeinsam, dass Kapital nur ein Mittel ist, um höhere und vielfältigere Ziele zu verfolgen. Die folgenden Beispiele zeigen, dass Unternehmen viele Aspekte der Gemeinwohl-Ökonomie bereits leben.

Bio- und Fair-Trade Betriebe – Diese Betriebe produzieren z.B. Bio-Lebensmittel, Gesundheitsprodukte und Textilien aus ökologischen Anbau. Des Weiteren legen diese einen hohen Wert auf faire Arbeits- und Handelsbedingungen, sodass alle Beteiligten der Wertschöpfungskette zufrieden sind. Einige Betriebe achten z.B. darauf, dass die Mitarbeiter kranken- und unfallversichert sind oder grüne Energie für Produktionsprozesse verwendet wird. In einer Gemeinwohlökonomie würden die Produkte, die nicht fair gehandelt werden z.B. durch Zollaufschläge benachteiligt werden, sodass nach einer mehrjährigen Übergangszeit nur noch faire Produkte produziert werden würden.

Community Supported Agriculture – Bei dem Konzept der „gemeinschaftsgestützen Landwirtschaft“ versorgt ein Hof sein Umfeld mit Lebensmitteln. Im Gegensatz stellt dieses Umfeld dem Hof die notwendigen Finanzmittel bereit und gibt eine Abnahmegarantie von sechs bis zwölf Monaten.

Banken – Ethik-Banken arbeiten nach sozial-ökologischen Grundsätzen, indem sie Unternehmen in den Bereichen „freie Schulen und Kindergärten, regenerative Energien, Behinderteneinrichtungen, Wohnen und nachhaltiges Bauen und Leben im Alter“ (Deuticke 2014, S. 155) finanzieren. Einige Ethik-Banken streben keinen Gewinn an, sondern agieren nach dem Kostendeckungsprinzip. Die Kredit- und Sparzinsen sind gering. Die Sparda-Bank agiert nach dem Prinzip der Gemeinwohl-Ökonomie, indem sie kostenlose Girokonten bereitstellen. Alle Konto-Kunden sind gleichzeitig stimmberechtigte Eigentümer, die den Aufsichtsrat und Vorstand wählen können. Die VertreterInnen entscheiden dann über die Verteilung der Gewinne.

Open-source und Free-software-Bewegung – Die Gemeinwohl-Ökonomie fördert die Weitergabe und nicht die kommerzielle Verwertung von Wissen. Die Aktivisten der Open-source und Free-software-Bewegung verfolgen das Ziel, dass Software nicht patentiert werden, sondern offen und kooperativ entwickelt werden soll. Aus dieser Forderung sind freie Hightech-Produkte wie z.B. das Betriebssystem Linux, der Webbrowser Firefox oder das Mailprogramm Thunderbird entstanden.

1.1.7 Umsetzungsstrategie

Im Jahr 2010 wurde die erste Fassung der Gemeinwohl-Ökonomie mit 15 sogenannten "Attac-UnternehmerInnen" entwickelt. Ein erster Versuch der Umsetzung wurde gestartet. Hieran waren und sind verschiedene Personen- und Wirkungsgruppen Beteiltigt.

Auf der Website scheinen außerdem ideelle UnterstützerInnen in fünf Kategorien auf. Bis Ende 2014 waren es rund 1750 Unternehmen, 6000 Privatpersonen, 65 PolitikerInnen (zahlloser Parteien) sowie sieben Gemeinden/Regionen. (Felber 2014, S. 167)

Im Folgenden werden die beteiligten Personen-Gruppen und -Kreise kurz erläutert:

Pionierinnen-Gruppen

1. *Wirtschaftliche PionierInnen* sind Pionier-Unternehmen und Organisationen die bereits die Gemeinwohl-Ökonomie auf erster Basis umsetzen. Sie erstellen die Gemeinwohl-Bilanz, kooperieren mit anderen Unternehmen statt zu konkurrieren und verbreiten die Idee der Gemeinwohl-Ökonomie regional.
2. *Politische PionierInnen* sind Gemeinwohl-Gemeinden, die Gemeinwohl-Bilanzen kommunaler Betriebe erstellen, diese Betriebe zu Versammlungen einlädt in denen das Verständnis und die Akzeptanz der Gemeinwohl-Ökonomie gesteigert wird und BürgerInnen bei der Entstehung des Gemeinwohl-Index (Gemeinwohl-Faktoren) beteiligt und einbezieht.
3. *Kulturelle PionierInnen* sind Lehrerkräfte (sowohl Schule als auch Universität) die das Verständnis für die Gemeinwohl-Ökonomie in den entsprechenden Institutionen stärken. Zusätzlich werden Überlegungen zu einem "Gemeinwohl-Ökonomie"-Studiengang angestellt.

AkteurlInnen-Kreise

1. *RedakteurInnen* arbeiten an den Gemeinwohl-Kriterien durch einbeziehen des Feedbacks durch weitere PionierInnen.
2. *BeraterInnen* unterstützen die Pionier-Unternehmen bei der Umsetzung der Gemeinwohl-Ökonomie (Hilfe beim erstellen der Gemeinwohl-Bilanz, Verbessern der Prozesse, usw.).
3. *AuditorInnen* überprüfen die eingereichten Gemeinwohl-Bilanzen der Pionier-Unternehmen. Außerdem führen sie regelmäßige Kontrollen in den auditierten Unternehmen durch.
4. *ReferentInnen* halten Vorträge zur Gemeinwohl-Ökonomie in Unternehmen, Gemeinden, Lehrinstitutionen und politischen Institutionen
5. *BotschafterInnen* sind Personen mit einem großen Wirkungskreis (zum Beispiel Helmut

- Lind, CEO der Sparda-Bank München) die den Bekanntheitsgrad und die Akzeptanz der Gemeinwohl-Ökonomie steigern.
6. *WissenschaftlerInnen* unterschiedlichster Disziplinen entwickeln gemeinsam an der Gemeinwohl-Ökonomie weiter.
 7. *KonsumentInnen* fragen bei Unternehmen bei denen sie Kunden sind nach, ob diese Unternehmen bereits eine Gemeinwohl-Bilanz erstellt haben - als Grundlage für eine Kaufentscheidung.

EnergieFelder

Energiefelder sind lokale oder regionale Gruppen, welche die GWÖ in den Gemeinden, Städten, Bezirken, Landkreisen und Großregionen verwirklichen. (Felber 2014, S. 177)

Energiefelder unterstützen die einzelnen PionierInnen-Gruppen. Sie fördern das Verständnis und die Akzeptanz durch öffentliche Veranstaltungen und unterstützen die Weiterentwicklung der Gemeinwohl-Ökonomie. Mehrere Energiefelder vernetzen sich zu einem Größeren.

Positive Rückkopplungen

Positive Rückkopplungen sind Reaktionen auf Verhalten einzelner "Systemmitglieder" um die Gemeinwohl-Ökonomie zu stärken. Sie sind wie in einem Regelkreis die Rückwirkung auf das Systemverhalten und dienen der richtigen Aussteuerung der Eingangsgrößen.

Es werden einige Beispiele genannt (vgl. Felber 2014, S. 178/179):

- günstige Kredite für Unternehmen mit hoher Gemeinwohl-Bilanz
- das Nutzen von Gemeinwohl-Banken erhöht die Gemeinwohl-Bilanz
- das Kooperieren mit anderen Unternehmen erhöht die Gemeinwohl-Bilanz
- usw.

Die positive Rückkopplung führt zum Verstärken des gemeinwohl-ökonomischen Verhaltens und zur Schwächung des Entgegengesetzten.

Strategische Vernetzung

Die strategische Vernetzung beschreibt die Notwendigkeit der Vernetzung ähnlicher oder verwandter Ansätze untereinander, sodass jeder einzelne dieser Ansätze von der erhöhten Verbreitung profitieren kann. Die Einzel-Akzeptanz vieler dieser Ansätze kann zu einer vergrößerten Akzeptanz und damit zu einem Paradigmenwechsel innerhalb der Gesellschaft führen. Außerdem sollen die vernetzten Ansätze sich gegenseitig helfen, infragestellen und damit verbessern.

1.2 Sharing Economy

Bei der Sharing Economy (auch Shareconomy oder Collaborative Consumption) rückt die Nutzung von Ressourcen in den Vordergrund. Um für Umweltschutz, Effizienz und Nachhaltigkeit zu sorgen wird immer öfter nicht der Besitz eines Guts sondern nur die Nutzung dieses angestrebt. Nur selten ist ein permanenter Zugriff auf eine Ressource, welche sich als Maschine, Örtlichkeit oder etwas abstraktes darstellen lässt, nötig. In diesem Fall soll durch das Teilen der Ressource für mehr Nachhaltigkeit und Kosteneffizienz gesorgt werden.

Der Gedanke der Shareconomy wurde bereits in den 70er Jahren gefordert, passte allerdings nicht zu den Gesellschaftsbedürfnissen, welche auf den Besitz von Gütern ausgelegt waren. Heutzutage hingegen wird bereits vieles in sozialen Netzwerken geteilt und somit ist der Gedanke des Teilens bereits in der Gesellschaft verankert.

In einer Statistik von 2010 aus den USA werden folgende Werte für geteilte Güter aufgezählt:

- Wohnraum (58%)
- Arbeitsraum (57%)
- Nahrungszubereitung (57%)
- Haushaltszubehör (53%)
- Kleidung (50%)

Zu den geteilten Gütern zählt auch der Kauf oder Verkauf von gebrauchten Gütern. Durch das Internet und die damit einhergehenden Möglichkeiten ist das Teilen von Ressourcen einfacher geworden. So werden öfter Immobilien, Verkehrsmittel oder selten genutzte Dinge wie Gartengeräte vermietet und geteilt.

Das momentan bekannteste Beispiel für das Verleihen von Gütern ist in der Autobranche zu finden. Vor allem in Städten wird ein Auto oftmals nichtmehr als Statussymbol, sondern als Belastung empfunden. Durch das Mieten und Teilen von Autos ist ein riesiger, neuer Markt entstanden, welcher sich Dank des Internets und mobiler Applikationen immer stärker verbreitet.

Der Grundgedanke hinter der Shareconomy ist der, ein Gut günstig zu leihen, anstatt es teuer zu kaufen. Dieser Gedanke findet sich auch immer öfter in großen Firmen und nicht nur bei Privatpersonen wieder. Um Ressourcen wie zum

Beispiel Speicherplatz zu sparen, werden eBooks, Musik und andere Medien digital oftmals zum Leihen angeboten. Der Konsument hat geringere Kosten für das Leihen und beansprucht nur für kurze Zeit seinen Speicherplatz.

Die Sorge, dass mit verliehenen Gütern nicht ordnungsgemäß umgegangen wird oder dass die Güter gestohlen werden können, ist momentan der stärkste Gegner der Shareconomy. Allerdings gab es ähnliche Sorgen bereits bei der Einführung des Online-Shoppings, welche sich schnell verflüchtigten.

Es gibt kritische Stimmen gegen die Shareconomy, welche behaupten, dass das Teilen ohne den eigentlichen Besitz eines Guts nicht möglich ist, wobei die Anzahl der Güter, welche sich im Umlauf befinden, allerdings verringert würde. Auch wird davon gesprochen, dass nicht die Nachhaltigkeit oder der Umweltschutz sondern lediglich der Kostenfaktor der Antrieb für die Shareconomy sei.

1.2.1 Crowdsourcing als neue Organisationsform

Der Begriff Crowdsourcing beschreibt den Zugriff auf ein gemeinschaftliches Wissen und personelle Ressourcen, um ein Problem zu lösen.

Unterkategorien des Crowdsourcing sind unter anderem:

- Crowdfunding (Finanzierung eines Projekts durch die Community)
- Co-Creation (Erschaffung eines Werkes durch die Community)
- Microworking (Teilaufgaben eines Projekts werden durch die Community erledigt)

Durch das Internet zählt die Crowdsourcing-Branche aktuell zu den am schnellsten wachsenden. Der Zugriff auf eine immer größer werdende Community, in welcher alle Arten von Fähigkeiten abgedeckt sind, wird durch Plattformen im Internet ermöglicht. Teams können Aufgaben zur Fertigstellung eines Projekts durch einfache Kommunikationsmöglichkeiten unabhängig von räumlichen und zeitlichen Gegebenheiten realisieren.

Die hohe Nachfrage an Crowdsourcing Dienstleistungen basiert auf folgenden sechs Grundprinzipien:

1. Die Anforderungen an technische und soziale Kompetenzen steigen.
2. Durch moderne Kommunikationstechnologien sind Personen nicht an Freizeit und Arbeitszeit gebunden.
3. Produkte werden durch Ideen, Innovationen und Informationen ersetzt.
4. Arbeitsszeiten und Löhne können flexibler an die Fähigkeiten der Personen angepasst werden.
5. Wertschöpfungsketten werden transparenter und der Konsument erlangt Eingriff in diese.
6. Durch Crowdfunding werden Produkte von Konsumenten vorfinanziert.

Im Gegensatz zu Outsourcing ist Crowdsourcing nicht an feste Arbeitszeiten und einen festen Arbeitsplatz gebunden. Außerdem sind die einzelnen Teammitglieder einfacher auszutauschen und die Bezahlung erfolgt nach dem Festpreisprinzip anstatt auf Stundenlohnbasis.

Die Motivation für Crowdsourcing erstreckt sich über viele Bereiche, wobei der materielle Ansatz ein sehr geringer ist. Ein Beispiel für Crowdsourcing ist Wikipedia, wobei Leute ihr Wissen teilen um selber von dem Wissen anderer profitieren zu können. Auch Wohltätigkeit, Spaß und Langeweile können motivieren. Geld ist nicht die beste Motivation für Crowdsourcing Projekte, da die Bezahlung oftmals nicht besonders hoch ist und die Teilnehmer dementsprechend auch nur einen Teil ihres Könnens einsetzen.

Für Unternehmen sind gibt es mehrere interessante Vorteile. Im Folgenden werden einige davon aufgelistet:

- Austausch von Wissen
- erweiterter Ideenpool
- Einbeziehung geographischer Besonderheiten
- Ressourcenersparnisse durch Spezialisierung

1.2.2 Beispiel: Vernetzte Arbeits- und Lebensräume

Vor allem in Städten ist der Effekt der Shareconomy deutlich zu spüren. Sei es bei dem Teilen von Büroflächen, bei Wohngemeinschaften oder bei der Autovermietung.

Der Begriff Co-Working beschreibt das Mieten eines Arbeitsplatzes mit Internetanschluss und einer Postadresse. Dieses Konzept ist vor allem bei Startups und Freelancern beliebt, da Kosten minimiert werden können, die Arbeiter nicht zu sehr abgelenkt werden wie wenn sie von zu Hause arbeiten würden und da der Austausch mit anderen Mieter möglich ist, wodurch ein professionelles Netzwerk aufgebaut werden kann.

Unter dem Begriff Co-Living wird das Teilen einer Wohnung verstanden. Vor allem für Städtetrips ist diese Hotelalternative besonders interessant, da Geld gespart werden kann und der Austausch zwischen verschiedenen Kulturen gefördert wird.

1.2.3 Vernetztes Wissen

Sogar für das Wissen an sich lassen sich Unterschiede zwischen alter und neuer Denkweise finden. Früher konnte Wissen als persönlicher Besitz angesehen werden. Oftmals blieben Arbeiter bis zur Rente nur in einer Firma, da sie sich in dieser Branche auskannten und der Umstieg in eine andere Firma eine Art neue Ausbildung mit sich ziehen würde. Heute ist dies zum Teil immer noch so, allerdings wird das Wissen heutzutage vermehrt geteilt. Im Internet findet man leicht Vorlesungen von Professoren oder auch Berichte von Experten, wodurch es einfacher ist, am technologischen Wandel teilzuhaben.

Der Begriff Co-Learning stellt das Lernen geteilten Wissens in einer Gruppe unabhängig vom Ort dar. So ist es beispielsweise möglich, Nischenangebote zu erstellen, welche in einer Universität nur sehr wenige Studenten zum teilhaben bewegen können, allerdings weltweit eine große Anzahl an Interessenten finden.

Foren und Wikis dienen als Wissens- und Datensammlungen, welche auf einer technologischen Organisationskultur aufbauen, ein nötiges Maß an Vertrauen und Transparenz vermitteln und die Beitragileistenden in Kategorien einstufen.

Das Wissen, welches in diesen Sammlungen geteilt werden soll kann entweder explizit oder implizit sein.

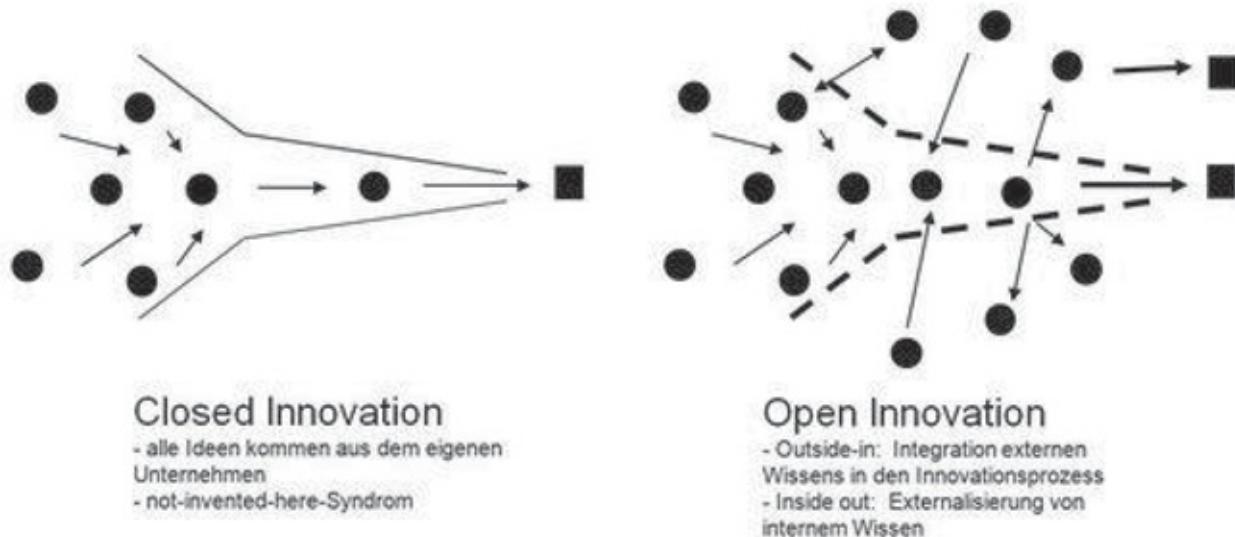
Folgende Punkte treffen für das explizite Wissen zu:

- Der Wissensinhaber kann die Information beschreiben
- Der Wissensempfänger muss Kenntnis über das Vorhandensein der Information haben
- Der Zugang zu den geteilten Informationen muss für den Wissensempfänger möglich sein
- Die Wissenssammlung muss strukturiert werden

Implizites Wissen hingegen kann vom Wissensinhaber nicht artikuliert werden. Hierunter fallen beispielsweise Fähigkeiten wie das Fahrradfahren. Der Austausch von implizitem Wissen kann somit nur in informellen Netzwerken stattfinden.

Auch innerhalb von Unternehmen ist ein Austausch von explizitem und implizitem Wissen oft nötig und kann durch ein Wiki abgewickelt werden, welches unter anderem Produktbeschreibungen, Anwenderhandbücher oder Allgemeines Markt- und Branchenwissen beinhalten kann. Hierdurch wird die Kommunikation oft vereinfacht, da der Zugang für alle betreffenden Personen möglich ist.

Durch Co-Creation und Open Innovation können auch externe Personen am Prozess der Entwicklung eines finalen Produkts in einem Unternehmen teilhaben. Impulse können hierbei von innen und von außen angestoßen und Informationen von innen nach außen weitergegeben werden. Dieser Prozess wird in der folgenden Abbildung dargestellt und wird als Crowdstorm (Brainstorming der Crowd -> Zusammensetzung von Brainstorming und Crowdsourcing) bezeichnet.



Je mehr sich die Unternehmen in diesem Prozess der Crowd öffnen, desto innovativer werden schlussendlich die Produkte, da die internen Mitarbeiter sich ansonsten in einer Art Blase befinden und sich vor externen Anreizen verschließen.

1.2.4 Beispiel: vernetzte Dienstleistungen

Durch die Digitalisierung konnte das Angebot und die Nachfrage von Dienstleistungen ins Internet verschoben werden und somit können Zeit- und Ortsunabhängig Spezialisten für gewisse Problemstellungen gefunden werden. Hierfür sind Online-Dienste wie Auftrags-Datenbanken oder das Micro-Payment unablässlich.

Es gibt verschiedene Tools, welche die ortsunabhängige Zusammenarbeit von Teams unterstützen wie beispielsweise Skype. Soll die Entwicklung von Projekten auch zeitunabhängig sein, so kann auf Dienste wie Google Drive zurückgegriffen werden. Diese Tools umfassen die Hauptaufgaben Online-Projektmanagement, Datentransfer und Kommunikation und werden oftmals gemeinsam eingesetzt.

Nicht nur Dienstleistungen werden vermehrt über das Internet angeboten, sondern auch Produktionsvorgänge werden immer mehr digitalisiert. Durch die Einführung des 3D-Drucks werden viele druckbare Objekte über das Internet erreichbar gemacht und können somit zu Hause erstellt werden. Natürlich ist auch das Teilen eines Objekts mit der Community möglich. Hierdurch werden Innovationen ins heimische Arbeitszimmer verlegt und somit werden Produktionsabläufe grundlegend beeinflusst.

Auch das Finanzwesen ist von der Co-Economy betroffen. Im Internet werden bereits viele Zahlungsmöglichkeiten wie Paypal angeboten. Auch Crowdfunding ist durch das Internet bekannt geworden und somit konnten bereits viele Innovationen umgesetzt werden.

Mittlerweile hat sich sogar digitales Geld in Form von Bitcoins im Finanzwesen durchgesetzt. Der Vorteil liegt hierbei darin, dass Geldbeträge banken- und zeitunabhängig den Besitzer tauschen können. Außerdem sind die Transaktionen sehr sicher und machen Rückbuchungen unmöglich.

Zur Zeit ist die Bitcoin allerdings noch sehr instabil im Kurs und der mögliche Diebstahl durch Hacker wird ebenfalls oft als ein hohes Sicherheitsrisiko angesehen.

Das Prinzip Crowdfunding ist eine Form des Crowdsourcings, in welcher Geld für Projekte gesammelt werden kann. Das Geld kommt hierbei von der Community welche automatisch einen gewissen Einfluss in die Entwicklung des Produkts bekommt.

Beim Crowdfunding kann durch die Analyse der zahlenden Community automatisch die Zielgruppe und der mögliche zu erschließende Markt erforscht werden.

Die zahlende Community kann das Geld entweder spenden, wodurch dieses nicht versteuert werden muss, oder allerdings eine materielle Gegenleistung erwarten.

Die Begriffe Crowdinvesting und Crowdloaning fallen unter den Begriff Crowdfunding.

Das Crowdinvesting stellt hierbei die klassische Finanzierung eines Startups dar, während beim Crowdloaning Geld an eine Privatperson in Form eines Kredits fließt. Die Privatperson zahlt das erhaltene Geld dann verzinst an die geldgebende Community zurück.

2 SoftSkills

2.1 Career

Nur wenige Softwareentwickler gestalten aktiv ihre Karriere. Aber Erfolg ist nie ein Zufall.
Man muss jeden Schritt langfristig planen.

2.1.1 Beginnen mit einem Bang: Mache nicht, was alle anderen machen

Man sollte seine Karriere als ein Geschäft und die Firma, bei der man arbeitet, als den Kunden für sein Geschäft verstehen. Damit fühlt man sich nicht von einer bestimmten Firma abhängig, sondern ist unabhängig und autonom.

Jedes Geschäft muss ein Produkt oder eine Dienstleistung anbieten. Meistens bieten Softwareentwickler Softwareentwicklung als Dienstleistung an. Geschäfte müssen ständig ihre Produkte verbessern. Als Softwareentwickler muss man es genauso machen. Das Produkt ist allein nicht genug, die potentiellen Kunden müssen von dem Produkt Bescheid wissen, um es kaufen zu können. Deswegen ist das Marketing genauso wichtig wie das Produkt. Besseres Marketing bedeutet höhere Preise.

2.1.2 Denk über die Zukunft nach

Als Unternehmen muss man Ziele festlegen. Das ist einfacher gesagt als getan. Man muss sich zuerst ein großes Ziel setzen. Dieses Ziel muss nicht konkret sein, es reicht z.B., wenn man festlegt, wo man sich in 5 oder 10 Jahren sieht. Danach setzt man sich kleinere Ziele (monatlich, wöchentlich, täglich), die dabei helfen, das große Ziel zu erreichen. Man muss die Ziele regelmäßig verfolgen und auf den neuesten Stand bringen, wenn es nötig ist.

2.1.3 People Skills

Das Code-Schreiben ist nur ein kleiner Teil der Verantwortungen eines Softwareentwicklers. Softwareentwickler beschäftigen sich mit anderen Menschen, z.B. schreiben wir unseren Code in High-Level-Programming-Languages, weil wir möchten, dass andere Personen unseren Code verstehen können. Deswegen muss ein guter Softwareentwickler einen Weg finden, um mit Menschen effizient zu kommunizieren.

Jeder Mensch möchte sich wichtig fühlen. Wenn wir möchten, dass die Kollegin unsere Ideen hören, müssen wir zuerst ihre Ideen hören.

Die Belohnung eines positiven Verhaltens ist immer besser als die Bestrafung eines negativen Verhaltens. Wenn man motivieren möchte, muss man nicht die Kritik benutzen, sondern das Lob.

Wenn wir mit Menschen erfolgreicher handeln möchten, müssen wir daran denken, was diese Person will und was für diese Person wichtig ist.

Soweit wie möglich sollte man Streit vermeiden, weil es nicht immer möglich ist eine Person mit konkreten Gründen zu überzeugen. Bei einer Diskussion, muss man überlegen, ob man diese bestimmte Schlacht wirklich gewinnen muss. Wenn das Thema nicht so wichtig ist, kann man akzeptieren, dass man nicht Recht hat. Dann ist man nächstes Mal stärker.

2.1.4 Vorstellungsgespräche

Vorstellungsgespräche sind für alle schwer, weil man nie wissen kann, was man gefragt werden wird. Allerdings gibt es eine Möglichkeit, um im Vorhinein in eine gute Ausgangsposition zu kommen.

Das Wichtigste ist, dass die Person, die das Vorstellungsgespräch führt, den Bewerber mag. Dies kann man erreichen, bevor das Vorstellungsgespräch anfängt.

Man sollte persönlichen Kontakt mit Mitarbeitern der Firma, bei der man sich bewirbt, aufnehmen und sich am besten mit den Mitarbeitern der Personalabteilung anfreunden, bevor man sich bewirbt. Wenn dies nicht möglich ist, sollte man vor dem Vorstellungsgespräch um ein „Vor-Vorstellungsgespräch“ bitten und Interesse an der Firma zeigen.

Während des Vorstellungsgespräches muss man zeigen, dass man Selbstvertrauen hat und weiß, wie man Dinge erledigt.

Selbst wenn man derzeit nicht nach einem Job sucht, ist es wichtig, dass man seine technischen Fähigkeiten auf dem Laufenden hält und sich auf potentielle zukünftige Vorstellungsgespräche vorbereitet.

Resumes are boring - Let's fix that

Im nächsten Unterkapitel behauptet der Autor, dass der durchschnittliche Lebenslauf eines Softwareentwicklers ein fünfseitiges monströses Dokument ist, welches eine Schriftart und zwei Spalten hat, und mit grammatischen Fehlern, Tippfehlern und erbärmlich strukturierten Sätzen voll mit Phrasen, wie "anführend" und "auf Ergebnisse fokussiert" gefüllt ist. Er vergleicht dies mit einem professionellen Lebenslauf und kommt zum Ergebnis, dass man lieber einen professionellen Lebenslauf-Schreiber anheuern sollte. Dies begründet er damit, dass das Schreibenlernen eines professionellen Lebenslauf eine Verschwendug von Zeit und Talent ist und dies besser in professionelle Hände gehört. Bei der Beauftragung eines professionellen Schreibers ist seiner Meinung nach darauf zu achten, dass er technisches Know-how besitzen sollte und auch Beispiele aus dem technischen Bereich zeigen kann. Bei der Beauftragung selbst sollte man beachten, dass man so viele Informationen wie möglich von sich selbst gibt und versucht sich selbst in einen positiven Licht mit den Informationen darstellen zu lassen. Auch ist es wichtig einen Format zu wählen, das für den Leser einfach verständlich ist.

Weiterhin regt der Autor an, dass man seinen Lebenslauf, wenn es fertiggestellt ist, auch online auf einer Homepage und Portalen wie LinkedIn zur Verfügung stellen sollte. Auch dies würde häufig von einer professionellen Fachkraft übernommen.

Wenn man trotzdem seinen Lebenslauf lieber selbst schreiben will, rät der Autor folgendes

zu beachten: Man soll seinen Lebenslauf online stellen, ihn in einzigartiger Weise präsentieren, vorherige Projekte im Lebenslauf präsentieren und er sollte frei von Tipp- und Sprachfehlern sein.

2.1.5 Welche Beschäftigungsmöglichkeiten gibt es?

Als Softwareentwickler hat man verschiedene Berufsmöglichkeiten:

- Arbeitnehmer:
 - Vorteile: Stabilität; einfach; bezahlter Urlaub; wenig Verantwortung
 - Nachteile: Mangel an Freiheit; begrenzte Verdienstmöglichkeiten
- Freelancer:
 - Vorteile: Mehr Freiheit; abwechslungsreiche Arbeit; potentiell: höheres Einkommen
 - Nachteile: Es ist schwer, Arbeit zu finden; hohe Ausgaben; mehrere Chefs, anstelle von einem Chef
- Unternehmer:
 - Vorteile: Freiheit; extrem hohes Einkommenspotential; kein Chef
 - Nachteile: Riskant; man ist auf sich allein gestellt; man braucht mehr als nur technische Fähigkeiten; lange Arbeitszeiten Am Anfang ist es meist einfacher als Arbeitnehmer zu arbeiten.

2.1.6 Welche Art von Softwareentwickler bist du?

Es ist wichtig, dass man sich als Softwareentwickler in einem Bereich spezialisiert. Zu sagen, „ich bin ein C#-Entwickler“ ist nicht genug. Die Regel der Spezialisierung lautet: Je mehr man sich spezialisiert, desto weniger potentielle Arbeitsplätze hat man, aber desto wahrscheinlicher ist es, dass man in einem dieser potentiellen Arbeitsplätze eingestellt wird.

Als Softwareentwickler kann man sich in vielen verschiedenen Bereichen spezialisieren, z.B. Programmiersprachen, Plattformen, Methodologien, oder spezifische Technologien. Bei so vielen Möglichkeiten ist es für manche Softwareentwickler schwer, eine Spezialisierung zu wählen. Man sollte beachten, einen Bereich zu wählen, den nur wenige andere Entwickler wählen, damit man bessere Berufschancen hat. Das wichtigste ist immer, eine Spezialisierung zu wählen. Man kann den Bereich nachher immer noch wechseln. Spezialisierung bedeutet außerdem nicht, dass man nur von einem Bereich Ahnung hat - es ist wichtig, von verschiedenen Dingen etwas zu verstehen.

2.1.7 Nicht alle Firmen sind gleich

Als Softwareentwickler kann man bei verschiedenen Firmen arbeiten: Kleine Firmen oder Startups; Mittlere Firmen; Große Firmen.

Meist sind kleine Firmen Startups. Als Softwareentwickler in einem Startup ist man nicht nur Code-Schreiber. Man muss flexibler sein, weil es in diesen Firmen weniger Mitarbeiter gibt. Der Vorteil am Arbeiten in kleinen Firmen ist, dass man häufig aufregende Arbeit hat, bei der man die Ergebnisse direkt sieht. Der Nachteil sind lange Arbeitszeiten, wenig Stabilität und weniger Lohn.

Die meisten Firmen sind Firmen von mittlerer Größe. Diese Firmen bieten mehr Stabilität als kleine und als große Firmen.

Große Firmen sind meist sehr institutionalisiert, es gibt bestimmte Prozeduren für jede Art von Arbeit. Sie bieten viele Möglichkeiten zur Weiterbildung und man hat die Möglichkeit an großen, bedeutenden Projekten zu arbeiten. Ein Nachteil an großen Firmen ist, dass man meist nicht direkt die Ergebnisse seiner Arbeit sieht, weil man nur in einem kleinen Bereich eines umfassenden Projektes arbeitet. Deswegen wird man in großen Firmen leicht übersehen.

Für Softwareentwickler ist es außerdem wichtig, ob man einer Firma arbeitet, die hauptsächlich Software entwickelt, oder in einer Firma arbeitet, deren Hauptaufgabe nicht Softwareentwicklung ist. Deswegen ist es wichtig, dass man sorgfältig darüber nachdenkt, in welcher Art von Firma man arbeiten möchte.

Working remotely survival strategies

In diesem Unterkapitel diskutiert der Autor über Strategien für den Fall, dass man von zuhause aus arbeitet. Er spricht davon, dass heutzutage eine Menge unabhängiger Software-Entwickler per Remote-Desktop oder über ein virtuelles Büro von zuhause aus arbeiten. Er geht dabei auf die Vor- und Nachteile der Heimarbeit ein. Als Nachteil sieht er zum Beispiel die Isolation, Einsamkeit und mangelnde Selbstmotivation. Er erwähnt, dass er anfangs die Vorteile dabei sah, aus seinem Bett morgens direkt an den Arbeitsplatz zu kommen. Sagt aber auch, dass er die entstandenen Herausforderungen unterschätzt hat. Die erste Herausforderung für ihn ist das Zeitmanagement. Er behauptet, dass es bei der Arbeit zuhause eine Menge Arten von Ablenkung gibt, die einen daran hindern die Arbeit zu erledigen. Auch neigt man angeblich dazu, seine Arbeit später am Abend zu erledigen, wenn man sich die Zeit abseits vom Alltagsstress nehmen kann. Seiner Meinung nach endet das aber in einem Desaster. Er behauptet, wenn man von zuhause aus arbeiten will, muss man ein vernünftiges Konzept für das Zeitmanagement erstellen.

Eine weitere Herausforderung ist die Selbstmotivation. Er empfiehlt Leuten die Probleme mit

Disziplin und Selbstkontrolle zu haben, nicht von zuhause aus zu arbeiten. Er meint, dass wenn man im Büro arbeitet, der Chef beobachten kann, wie man seine Arbeit erledigt. Wenn man allerdings von zuhause aus arbeitet, kann dies der Chef nicht. Das heißt, dass man für seine Disziplin und Motivation selbst verantwortlich ist. Er hält eine Planung und Routine der Zeiten bei der Heimarbeit für sehr wichtig, vor allem für die Phasen bei denen man nicht motiviert ist. Auch sollte man Versuchungen und Ablenkungsmöglichkeiten von seinem Arbeitsplatz entfernen. Wenn man sich absolut unmotiviert fühlt, schlägt er vor eine Zeituhr auf 15 Minuten zu stellen und sich in diesen 15 Minuten zu seiner Arbeit zwingt. Meist ist es der Fall, dass man nach jenen 15 Minuten mehr motiviert ist seine Arbeit fortzusetzen. Ein weiteres Hilfsmittel ist das Nutzen der Kommunikation mit Mitarbeitern per Google Hangouts oder Skype, um nicht sozial von der Arbeit abgehängt zu werden.

2.1.8 In der Firma aufsteigen

Es ist nicht immer einfach in seinem Job aufzusteigen. Die beste Art im Job aufzusteigen ist Verantwortung zu übernehmen. Wenn einem angeboten wird, Verantwortung zu übernehmen, sollte man dies also unbedingt annehmen. Sollte man keine Verantwortung angeboten bekommen, kann man dies selber erzwingen, in dem man sich z.B. auf in der Firma vernachlässigte Bereiche konzentriert.

Außerdem ist es sehr wichtig, dass man für seine Vorgesetzten sichtbar ist. Dies kann man z.B. dadurch erreichen, dass man seinen Vorgesetzten wöchentliche Arbeitsberichte schreibt.

Ein weiterer Schritt, um in der Firma aufzusteigen besteht darin sich permanent weiterzubilden. Dabei sollte man möglichst versuchen seinem Umfeld zu zeigen, was man lernt.

Des Weiteren sollte man versuchen, wie jemand zu wirken, der Lösungen für alle Art von Problemen findet. Problemlöser sind immer beliebt.

2.1.9 Professionell sein

Es ist sehr wichtig ein professioneller Arbeitnehmer zu sein – und kein Amateur. Ein professioneller Arbeitnehmer nimmt seine Karriere und seine Verantwortungen ernst. Professionelle Menschen wissen nicht alles und geben dies auch gerne zu. Aber sie sind stabil und zuverlässig.

Um ein professioneller Arbeitnehmer zu werden muss man sich gewisse Angewohnheiten, wie z.B. effektives Zeitmanagement, aneignen.

Außerdem muss man als Softwareentwickler technisch und ethisch richtige Entscheidungen treffen – selbst wenn dies kurzfristige Nachteile bedeuten. Man muss zu seinen Prinzipien stehen.

Professionell bedeutet außerdem permanent daran zu arbeiten, sich selbst zu verbessern und hohe Qualität zu liefern.

Don't get religious about technology

Im letzten Unterkapitel warnt der Autor davor bei Technologien religiös zu werden. Damit meint er, dass Leute, die eine Technologie, Programmiersprache oder Software kennen, gerne an dieser festhalten ohne offen für etwas anderes zu sein. Er hält dieses Verhalten für destruktiv und limitierend. Wir würden einen Punkt erreichen, an dem wir aufhören zu "wachsen" und meinen auf alles eine Antwort gefunden zu haben. Er sagt von sich, dass er aufgehört hat religiös zu sein und in einem Punkt seiner Karriere verschiedene Betriebssysteme, Programmiersprachen und Texteditoren ausprobiert und gelernt hat, bevor er sich für seine beste Technologie entschieden hat.

Der Autor meint aus seiner Perspektive betrachtet, dass nicht alle Technologien großartig sind, sondern manche nur gut sind. Von Zeit zu Zeit kann sich dies auch ändern. Er sagt, dass es nicht nur die eine gute oder gar die beste Lösung für ein Problem gibt. Man entscheidet danach, was man am besten findet, aber das heißt nicht notwendigerweise, dass es das Beste ist. Der Autor erzählt, dass er in seiner früheren Zeit darüber debattierte, ob Microsoft oder Mac besser ist, C# besser ist als Java oder statische Programmiersprachen besser sind als dynamische. Er sagt, dass seine eigene Ansicht falsch war. Er hat in den letzten Jahren an einem Javaprojekt gearbeitet und gelernt für alles offen zu sein. Er versucht nun die Dinge auszuprobieren, bevor er eine Entscheidung trifft. Sein Punkt dabei ist, dass man in seinen Optionen keine Limits setzen sollte. Man sollte nicht seine Wahl der Technologie für die beste erklären und alles andere ignorieren. Es wird seiner Meinung nach am Ende einem nur weh tun. Andererseits sagt er, dass nur dann wenn man gewillt ist offen zu sein und man etwas für das Beste erklärt, andere einen folgen werden.

2.1.10 Freiheit: Den Job kündigen

Wenn man seine eigene Firma möchte ist es sehr riskant seinen Job bei einer anderen Firma einfach so zu kündigen. Bevor man seinen Job kündigt sollte man anfangen die Firma, die man gründen möchte, aufzubauen und quasi einen Nebenverdienst zu haben.

Die meisten Menschen unterschätzen, wie schwer es ist, sich selbstständig zu machen. Die Arbeitszeiten sind z.B. sehr lang. Wenn man seine eigene Firma als Nebenverdienst aufbaut, bekommt man ein Gefühl dafür, wie schwer es ist, seine eigene Firma aufzubauen. Man sollte außerdem wissen, dass die meisten neugegründeten Firmen pleitegehen.

Wenn man selbstständig ist, muss man viel härter arbeiten. Als normaler Mitarbeiter in einer Firma verschwendet man die Hälfte der Arbeitszeit mit anderen Dingen und arbeitet eigentlich nur vier Stunden am Tag. Als Selbstständiger ist das nicht möglich.

2.1.11 Freelancing

Um als Freelancer zu arbeiten, sollte man freelancing als Nebentätigkeit anfangen und mit der Zeit baut man einen Kundenstamm auf.

Wie bekommt man seinen ersten Kunden? Höchstwahrscheinlich wird der erste Kunde jemand aus dem Bekanntenkreis sein, da Bekannte eher bereit sind einem zu vertrauen. Sollte man keine Bekannten haben, die potentielle Kunden sein könnten, sollte man Inbound Marketing anwenden. Inbound Marketing ist, dass man nicht Kunden sucht, sondern das Kunden selber zu einem kommen. Es gibt verschiedene Wege um das zu schaffen. Z.B. kann man etwas kostenfrei anbieten (bspw. in einem Blog) oder Email-Marketing machen.

Freelancing ist riskanter als Arbeitnehmer zu sein. Es gibt mehr Ausgaben (z.B. Steuern, Sozialabgaben, Nebenkosten etc.). Daher sollte man als Faustregel doppelt so viel Lohn verlangen, wie man als Arbeitnehmer bekommt. Aber man kann den Lohn nur so hoch ansetzen, wie der Markt bereit ist zu zahlen. Deswegen braucht man einen hohen Bekanntheitsgrad.

2.1.12 Das erste Produkt

Jedes Produkt löst ein Problem. Wenn das nicht so ist, hat das Produkt keinen Zweck. Deswegen muss man erst überlegen wer das Produkt kaufen soll, d.h. wer die Zielgruppe sein soll. Manchmal ist es besser, zuerst einige Leute zu finden, die bestimmte Probleme haben und eine Lösung für diese bestimmten Probleme zu finden.

Bevor man sein Produkt entwickelt, sollte man rausfinden, ob die Zielgruppe wirklich für das Produkt Geld ausgeben würden. Dies kann man schaffen indem man z.B. potentiellen Interessenten vorab die Möglichkeit gibt, Geld zu investieren.

Für das erste Produkt ist es außerdem ratsam mit einem kleinem Produkt anzufangen.

Fake it till you make it

Hier spricht der Autor davon, dass man immer wieder auf Herausforderungen und Grenzen stößt, auf die man nicht vorbereitet ist. Er sagt, dass es einige Leute gibt, die der Herausforderung lieber ausweichen und andere die sie annehmen und kämpfen. Er behauptet, dass es nicht an ihrer Selbstsicherheit und Fähigkeit liegt, Erfolg zu haben, sondern dass sie alle in der Lage sind vorzutäuschen, dass sie es schaffen. Er meint, dass das Vortäuschen etwas zu Können bis man es schafft, etwas mit Selbstsicherheit zu tun hat. Man handelt nach einem großen Glauben an sich selbst alle Hürden überwinden zu können. Man muss dazu bereit sein ins große Unbekannte springen zu wollen, auch wenn man dann nicht weiß was man tun muss und Angst dabei empfindet. Er hält es nämlich für unmöglich als Softwareentwickler ein Experte für alles zu sein. So würden auch die meisten Jobinterviews Fähigkeiten erfordern, die man noch nicht besitzt. Er meint, dass das Schlüsselwort hier das "noch nicht" ist, dass es wichtig ist, sein Auge auf die Zukunft zu richten statt auf das, was man noch nicht kann. Daher ist es wichtig eine Aura von Selbstsicherheit auszustrahlen mit dem Wissen, dass man Herausforderungen in der Vergangenheit gemeistert hat und es keinen Grund gibt, dass man sie nicht auch in der Zukunft meistern kann. Weiterhin soll man nach Meinung des Autors kein Lügner sein, sondern ehrlich mit den eigenen Fähigkeiten sein und zeigen, dass man in der Lage ist mit Hindernissen umzugehen.

2.1.13 Möchtest Du ein Startup gründen?

Es gibt zwei Arten von Startups. Die erste Art von Startups versuchen Geld von Investoren zu bekommen. Diese Startups haben die Absicht zu großen Firmen zu werden, können aber leicht scheitern. Die zweite Art von Startups, Bootstrapped Startups werden von ihren Gründern finanziert. Diese Startups bleiben meist kleinere Unternehmen, aber scheitern weniger häufig.

Die meisten Startups haben das Ziel, irgendwann einen großen Profit zu erwirtschaften. Investoren haben fast immer eine Exit-Strategie. Die kann entweder darin bestehen das Startup an eine große Firma zu verkaufen oder das Unternehmen an die Börse zu bringen.

Ein Startup basiert meist auf einer Idee. Ein guter Startup hat a) eine einzigartige Idee oder Erfindung, die sich schwer kopieren lässt und b) das Potential stark zu wachsen.

Eine wichtige Hilfe, um mit einem Startup erfolgreich zu sein, kann man mit einem Startup-Accelerator-Programm bekommen. Accelerator-Programme bieten Hilfe und auch ein bisschen Geld für Startup-Firmen und bekommen im Gegenzug Anteile vom Startups. Viele große Technologieunternehmen waren zu Beginn in einem Accelerator-Programm (z.B. Dropbox).

Nach dem Accelerator-Programm muss das Startup seinen ersten Investor (seed money) finden. Mit diesem Geld baut man sein Business-Modell auf. Wenn man dieses Geld verbraucht hat, muss man neue und große Investoren finden (venture capitalists). Diese Finanzierung wird Series A genannt. Wenn dieses Geld nicht ausreicht um die Firma groß genug zu machen und sie zu verkaufen, muss man erneut große Investoren finden. Dieser Prozess wiederholt sich so lange, bis man scheitert oder die Firma verkauft.

2.2 Marketing yourself

Der Unterschied zwischen einem erfolgreichen Musiker und einem Superstar ist ihre Vermarktung, denn Marketing ist ein Talent-Multiplikator.

Was ist Selbstmarketing?

Selbstmarketing bedeutet, sich mit Menschen zu verbinden, die etwas haben wollen, was du bei dir hast oder in dir trägst. Es bedeutet, für andere einen (Mehr-)Wert zu generieren. Dabei kontrolliert die Person, die sich selbst vermarktet, was sie anderen vermittelt, welche Nachricht sie sendet und welches Bild sie vermitteln möchte.

Warum Selbstmarketing wichtig ist

Selbstmarketing garantiert keinen Erfolg, aber es ist ein sehr wichtiges Element, welches du kontrollieren kannst. Viele Software-Entwickler können ein sehr hohes Kompetenz-Level innerhalb von zehn Jahren in ihrer Karriere erreichen. Danach kann es sehr schwierig sein, diese Kompetenzen weiter auszubauen, weil der intellektuelle Input von anderen Menschen fehlt. Das individuelle Talent wird deutlich an Bedeutung verlieren, weil jeder Software-Entwickler mit allen anderen Software-Entwicklern konkurriert, die ähnliche Kompetenzen besitzen wie das einzelne Talent. Durch Selbstmarketing kannst du herausfinden, was dich auszeichnet und welche deine Kompetenzen sind.

Wie du dich selbst vermarkten kannst

- Alles beginnt damit, eine Markenpersönlichkeit von dir zu kreieren; etwas, was dich widerspiegelt. Treffe bewusste Entscheidungen darüber, wer du sein möchtest und wie du dieses Bild in der Welt darstellen möchtest. Außerdem solltest du eine familiäre Atmosphäre erzeugen, wenn dir jemand entgegentritt oder wenn es sich um ein Produkt handelt, welches du schon viele Male kreiert hast. Benutze viele Kanäle für das Selbstmarketing. Ein Blog kann die Basis für deinen Internet-Auftritt bilden, weil du dort den Informationsfluss vollständig steuern kannst und weil du so nicht von anderen Menschen und ihren Plattformen und Regeln abhängig bist. Dazu gibt es eine Strategie von dem Unternehmer Pat Flynn, die "Be everywhere" genannt wird. Sie besagt, dass du überall dort sein musst, wo du dich selber vermarkten möchtest; dies hat das Ziel, dass du überall dort auch eine gute Chance hast, von deiner gewünschten Zuhörerschaft gehört zu werden. Mögliche Kanäle sind Blog-Posts, Podcasts, Videos, Artikel in Fachzeitschriften, Bücher, Code-Camps und Konferenzen. Dabei solltest du nicht vergessen, dass all dies von deiner Fähigkeit abhängt, (Mehr-)Werte für andere zu generieren.

2.2.1 Erschaffe eine Marke, die auf dich aufmerksam macht

Das Logo eines Unternehmens ist eine visuelle Erinnerung an seine Marke, aber es ist nicht die Marke selbst. Wenn du dieses Logo siehst, hast du Erwartungen an die Dienstleistung oder das Produkt. Eine Marke ist ein Versprechen, einen bestimmten (Mehr-)Wert zu erzeugen, in der Art, in der du es selber erwartest.

Was eine Marke ausmacht

Du brauchst vier Dinge für eine erfolgreiche Marke: eine Nachricht, Visuelles, Konsistenz und eine wiederholte Marken-Darstellung: Eine Nachricht ist das, was du versuchst zu vermitteln und Gefühle, die du mit deiner Marke versuchst, hervorzurufen. Visuelles repräsentiert deine Marke: Ein Logo ist eine einfache Repräsentation, ein Set an Farben und ein Style, der die Marke repräsentiert, machen auf sie aufmerksam. Sei konsistent, sodass ein Kunde keine wechselnden Erfahrungen mit deiner Marke hat. Eine wiederholte Marken-Darstellung ist essentiell, weil ein Kunde sich dadurch an dich und deine Marke erinnert, wenn er das Visuelle deiner Marke einige Male gesehen hat. So kann er diese Erfahrungen mit deiner Dienstleistung oder mit deinem Produkt verknüpfen.

Kreiere deine eigene Marke

Entscheide zunächst, was du repräsentieren möchtest und definiere deine Nachricht dafür. Verkleinere deine Zuhörerschaft und wähle eine Nische aus (Spezialisierung). Der beste Weg für die Entscheidung, welche Nische du wählen sollst, erfolgt von einer rein strategischen Perspektive: Welchen Vorteil hast du von einer bestimmten, spezifischen Nische, die du nutzen kannst? Zögere nicht, deine Nische später zu verändern, wenn dies nötig ist.

Schritte, eine Marke zu kreieren, sind: Definiere deine Nachricht, wähle deine Nische, erschaffe dein Motto, kreiere einen Elevator-Pitch und designe die visuellen Komponenten deiner Marke. Starte mit deinem Motto (Slogan), welches deine Marke in einem einzigen oder in zwei Sätzen darstellt. Ein Elevator-Pitch ist eine kurze Beschreibung dessen, was du machst und des einzigartigen Wertes, welchen du erschaffst (dies kann in der Zeit erklärt werden, die man benötigt, um mit einem Fahrtstuhl zu fahren). Dieser Elevator-Pitch stellt sicher, dass du und deine Marke konsistent seid und dass ihr immer die gleiche Nachricht vermittelst. Visuelle Komponenten sollten dabei helfen, diese Nachricht zu übermitteln und als visuelle Erinnerung dessen dienen, was deine Marke darstellen soll.

2.2.2 Einen sehr erfolgreichen Blog kreieren

Ein Blog kann eine Menge an Informationen über einen Entwickler enthalten; z. B. Programmier-Beispiele und -sammlungen, sowie detaillierte technische Analysen verschiedenster Aspekte der Software-Entwicklung. Ein Blog kann mehr Informationen übermitteln, als jede andere Art der Kommunikation wie Interviews oder Zusammenfassungen. Durch einen Blog kannst du einen besseren Job erlangen, du kannst aber auch ein besserer Software-Entwickler und Kommunikator werden. Ein Blog kann dir Möglichkeiten verschaffen, die du dir vielleicht nie zuvor vorgestellt hast. Wenn du ein Freelancer bist, kannst du herausfinden, dass ein erfolgreicher Blog viele Kunden zu dir führen kann ohne dass du nach ihnen suchen musst. Ein Kunde, der zu dir kommt ist bereit viel mehr Geld zu zahlen. Es wird viel einfacher sein, ihn von deinem Talent und deinen Kompetenzen zu überzeugen, sodass du z. B. einen Job von ihm bekommst. Bei genügend Blog-Traffic kannst du den Blog als Plattform für die Vermarktung deiner Produkte nutzen oder ein Produkt erzeugen, welches sich die Blog-User wünschen.

Kreiere einen Blog

Anfangs kannst du eine kostenlose Software für deine Blog nutzen (z. B. WordPress). Später, oder wenn du einen Shop oder Ähnliches benötigst, ist es ratsam einen kostenpflichtigen Service zu nutzen, da so Vieles einfacher ist. Registriere deine eigene Domain, um SEO-Kriterien zu erfüllen.

Schlüssel zum Erfolg

Der wichtigste Aspekt auf dem Weg zum Erfolg als Blogger ist Konsistenz. Dies ist sehr wichtig, da dich so viele Menschen über Suchmaschinen finden können. Je höhere Qualität dein Inhalt hat, desto wahrscheinlicher ist es, dass Menschen zu deinem Blog zurückkehren oder dort etwas schreiben oder ihn in den sozialen Medien teilen, sodass dein Blog mit anderen Websites verlinkt wird (wichtiger Aspekt um in Suchmaschinen gefunden zu werden).

Erhalte mehr Traffic

Starte damit, die Blogs anderer Leute zu kommentieren, die ähnliche Themen behandeln. So kommst du auch mit anderen Bloggern in Kontakt. Teile deine Blog-Posts in den sozialen Medien und verweise in deiner E-Mail-Signatur auf deinen Blog. Vereinfache es, deine Inhalte zu teilen, indem du Teil-Buttons verwendest. Schließlich, wenn deine Inhalte gut und kontrovers genug sind, kannst du deine Posts an Seiten wie Reddit oder Hacker News schicken.

2.2.2 Einen sehr erfolgreichen Blog kreieren

2.2.3 Ich kann dir keinen Erfolg garantieren

Dein oberstes Ziel: Generiere einen (Mehr-)Wert für andere

- "If you help enough people get what they want, you will get what you want" - Zig Ziglar

Gib den Menschen, was sie möchten

Herauszufinden, was Menschen wirklich möchten, ist nicht einfach - vor allem, wenn sie dies selber nur recht vage wissen. Versuche, die Zeichen zu lesen. Gehe raus und versuche herauszufinden, für was die Menschen sich interessieren. Über welche Themen wird in Internet-Foren gesprochen, die sich auf deine Nische beziehen und die relevant für dich sind? Welche Trends siehst du allgemein in der Industrie? Welche Ängste haben die Menschen und wie kannst du diese Ängste adressieren?

Stelle 90% deiner Leistung kostenlos zur Verfügung

Kostenlose Inhalte werden häufiger geteilt als solche, für die man bezahlen muss. Kostenloses zur Verfügung Stellen ermöglicht es Kunden, den Wert deiner Inhalte zu erfahren, ohne dass sie dafür vorher Geld investieren müssen. Dadurch kannst du die Menschen viel einfacher davon überzeugen, für dein Produkt zu zahlen, das sie sich über die hohe Qualität deines Produktes schon vorher im Klaren.

Der schnelle Weg zum Erfolg

Wie kann dein Inhalt einen (Mehr-)Wert für andere generieren? Dein größter Erfolg ist es, wenn du die Probleme anderer Menschen lösen und ihnen auf diese Weise helfen kannst.

Biete mehr von dir an

Die produktivsten Menschen sind die hilfreichsten.

2.2.4 #UsingSocialNetworks

Vergrößere dein Netzwerk

Folge anderen Leuten oder frage ander deinem Netzwerk beizutreten. Viele Entwickler warten geradezu auf Leute, die ihnen folgen oder mit ihnen interagieren. Platziere deine Links zu deinen Social Media-Profilen in deine Online-Biographien, ans Ende deiner Blog-Beiträge oder auch in deine Email-Signaturen.

Nutze soziale Medien effektiv

Bewege Leute dazu von einfachen Followern zu deinen Fans zu werden, sodass sie sich mehr mit deinen Inhalten befassen und deine Reputation in der Branche aufbauen. Poste alles, was du nützlich oder interessant findest, denn wenn du es wertvoll findest, finden es andere es wahrscheinlich auch. Mögliche Inhalte, die du teilen und veröffentlichen kannst wären:

- finde beliebte Blog-Einträge oder teile deine eigenen
- teile interessante Artikel, die wenn möglich inhaltlich zu deiner Nische passen oder allgemein zum Thema Software-Entwicklung gehören
- bekannte Zitate, besonders die inspirierenden sind besonders beliebt
- spezielles Wissen, dass du hast und jemand anderes wertschätzt
- ein bisschen Humor ist in Ordnung, aber sei dir sicher, dass es nicht zu provokativ, sondern einfach lustig ist
- mache ein bisschen von allem

Bleibe stetig aktiv

Du solltest ein oder zwei Plattformen wählen, auf denen du besonders engagiert bist, da du nicht alle Plattformen bedienen kannst, ohne sehr viel deiner Zeit damit zu verbringen. Ein Tipp ist es deine Social Media-Beiträge zu terminieren, sodass sie zu verschiedenen Zeiten innerhalb einer Woche veröffentlicht werden. Am besten parallel auf allen deinen Plattformen. Ein Werkzeug dafür wäre z.B. Buffer.

Netzwerke und Profile

Du solltest eine Präsenz besonders in den Technologie-relevanten und Karriere-fokussierten sozialen Netzwerken haben. Es ist empfehlenswert ein Twitte-Konto zu haben, da viele Entwickler Twitter nutzen und es leichte Mechanismen hat, um auf Tweets von anderen zu antworten und ebensolche zu referenzieren. Zudem solltest du ein LinkedIn-Konto haben, weil es ein soziales Netzwerk speziell für Fachleute ist und dadurch besonders gut für's

Netzwerken ist. Eine sehr empfehlenswerte Funktion von LinkedIn ist dabei die Möglichkeit seine eigenen Kunden nach sog. "Endorsements" zu fragen, was es möglich macht Feedback für deine Aufträge zu erhalten, die du in deinem Profil aufführst.

2.2.5 Vorträge, Präsentationen und Schulungen: Sprachkünstler

Vorträge halten oder Schulungen geben sind die effektivsten Wege mit anderen Leuten in Kontakt zu treten, auch wenn der Einflussradius dabei nicht sehr groß ist im Vergleich zu anderen Methoden. Aber vor einem Publikum zu stehen und direkt zu ihm zu sprechen ist eines der wirkungsvollsten Dinge, die du tun kannst.

Warum live vortragen so wirkungsvoll ist

Es entsteht eine persönliche Verbindung, wenn du ein Live Event besuchst, die du nicht bekommst, wenn du einfach eine Aufzeichnung hörst oder schaust. Auf diese Art ist wahrscheinlicher, dass Leute sich an dich erinnern und eine persönliche Verbindung zu dir aufzubauen. Zudem kann Vortragen ein interaktives Medium sein, bei dem du direkt auf Fragen des Publikums eingehen kannst und dieses sich an deiner Präsentation beteiligt.

Wie man mit dem Sprechen beginnen kann

Beginne damit Präsentation an deinem eigenen Arbeitsplatz zu halten über dort relevante Themen. Die meisten Unternehmen sind glücklich darüber, wenn ihre eigenen Mitarbeiter Themen intern vorstellen. Besuche sog. User Groups bei denen du nach einer Weile den Organisator fragen kannst, ob du über ein bestimmtes Thema vortragen kannst. Eine User Group ist dabei eine kleinere Gruppe und ein Publikum, das vergibt. Zusätzlich gibt es jährlich Code Camps überall auf der Welt, bei denen jeder mit beliebigen Erfahrungs niveau vortragen darf. Diese Veranstaltungen sind Situation mit wenig Druck, da niemand für die Vorträge zahlt. Sobald du einige Erfahrung gesammelt hast, kannst du an Entwicklerkonferenzen teilnehmen.

Was ist mit Schulungen?

Online Video-Schulungen sind deutlich leichter und skalierbarere Lösungen für Entwickler, um sich Reputation aufzubauen. Man kann mit einem einfachen Screencast anfangen und diesen auf einer freien Video-Seite teilen. Danach kannst du damit anfangen dich für deine Inhalte, die du produzierst, bezahlen zu lassen. Der einfachste Weg sind Portale die dich für deine Inhalte bezahlen und dir einen Teil vom darauf generierten Profit geben in Form von Autorenhonoraren. So hast du dich nicht über Vermarktung und Vertrieb zu kümmern.

2.2.6 Schreibe Bücher und Artikel die eine Anhängerschaft generieren

Warum Bücher und Artikel wichtig sind

Wenn du als jemand glaubwürdiges in deiner Branche angesehen werden willst, solltest du ein Buch schreiben oder Artikel in Software-Entwicklungs-Zeitschriften veröffentlichen. Ein Buch ist eine Möglichkeit deine Nachricht sehr gezielt und fokussiert zu übermitteln. Denn wenn sich jemand hinsetzt, um ein Buch zu lesen, bekommst du als Autor die fokussierte Aufmerksamkeit des Lesers für eine lange Zeitspanne.

Bücher und Zeitschriften bezahlen nicht

Du schreibst ein Buch nicht, um Geld zu machen, sondern du schreibst ein Buch, um deine Reputation zu erhöhen. Die meisten Zeitschriften bezahlen nur sehr wenig für einen Artikel, während das Schreiben und Bearbeiten sehr lange dauern kann. Jedoch agiert die Verlagswirtschaft als eine Art Gate-Keeper für Qualität. Zudem wirst du erkennen, dass es viele andere lukrative Möglichkeiten, die sich indirekt selber präsentieren und publiziert werden können. Autoren, die bereits publiziert haben, werden es einfacher haben zu Konferenzen eingeladen zu werden und können sich selber als Autorität in einem bestimmten Themenbereich etablieren. Da führt wiederum zu mehr Kunden und besseren Jobangeboten.

Publiziert werden

Publiziert werden ist nicht einfach, besonders. Getting published isn't easy - especially for your first book as not too many publishers want to take a risk on a completely unknown author. Best way to give yourself an opportunity to get published is to have a clearly defined topic that you know there's a market for and you can demonstrate your knowledge as an expert in that area. Start with blog posts, magazine articles and so get bigger and bigger. Publishers like to publish authors who already have a fairly large audience. You should have a solid proposal or magazine abstract

Sich selber publizieren

More authors are finding success by self-publishing - especially if they have an existing audience. It is easy to do and a good training before entering into a contract with a publisher that will have deadlines that you'll be required to meet. There are many services you can use to help you self-publish your book like Leanpub, Amazon Kindle Direct Publishing, Smashwords or BookBaby

2.2.6 Schreibe Bücher und Artikel die eine Anhängerschaft generieren

2.2.7 Habe keine Angst, dumm dazustehen

Alles ist anfangs unangenehm

When you first do something that makes you feel uncomfortable, you can't imagine how you could ever feel comfortable doing that thing. But you just have to learn to overcome this kind of thinking and realize that almost everyone goes through the same kind of uncomfortable feelings when they first do anything challenging - especially in front of a group of people

Es ist in Ordnung, wie ein Idiot zu erscheinen.

Things will get easier over time. And when things go wrong while presenting you should just don't care. And after it's over, chances are no one will even remember it. If you want to succeed, you have to learn how to swallow your pride and get out there and not be afraid to make a fool of yourself

Mache kleine Schritte (oder springe ins kalte Wasser).

Just diving in is the most effective way. But if you're nervous about doing speaking, writing, or something else, try to think of the smallest thing you can do that doesn't make you quite as nervous. Start with commenting and contribute to conversations but be prepared for criticism. Once you feel a bit braver, write your own blog posts like "how-tos". From there expand further by writing a guest post for someone else's blog or you can be interviewed on a podcast. And you might even join a club like Toastmasters to help you get used to speaking in public.

2.3 Lernen

Als Softwareentwickler arbeitet man in der Regel mit einer großen Anzahl an Technologien zusammen. Technologien, die heute aktuell sind, sind es morgen eventuell nicht mehr. Um auf dem neusten Stand zu bleiben, muss man sich daher ständig in neue Technologien einlesen.

Einer der wichtigsten Soft Skills ist daher, sich selbst etwas beizubringen. Doch wie lernen wir etwas am effektivsten? Es ist ein Mythos, dass wir alle auf verschiedenen Arten lernen. Wir lernen alle am besten, indem wir etwas ausprobieren oder es jemandem erklären und wir tendieren dazu einfacher zu lernen, wenn es uns interessiert. Man kann noch so viele Bücher übers Fahrradfahren lesen und noch so viele Videos schauen, wenn man es das erste Mal ausprobiert, wird es nicht sofort funktionieren. Trotzdem greifen viele Softwareentwickler zu einem Buch, um sich in eine neue Programmiersprache einzulesen. Anstatt dessen sollte man jedoch möglichst früh versuchen, etwas praktisch zu tun. Wir sind von Natur aus kreativ und neugierig. Nutzen wir diese Aspekte aus, können wir sowohl unsere Motivation, als auch unsere Lerngeschwindigkeit erhöhen.

Um eine neue Technologie zu erlernen, sind im Kern drei Punkte wichtig:

- Was benötigt man um anzufangen?
- Was kann ich grob damit tun?
- Was sind die Grundlagen? Dass heißt welche 20% muss ich lernen, um 80% meiner täglichen Aufgaben zu erledigen?

2.3.1 Die 10 Schritte

Dieses Kapitel soll anhand von zehn Schritten zeigen, wie wir diese Fragen beantworten können und es schaffen, möglichst schnell und effektiv etwas neues zu erlernen.

Schritt 1: Sich einen Überblick verschaffen.

Im ersten Schritt geht es darum, zu verstehen worum es grob geht. Dafür reicht in der Regel eine einfache Internetrecherche aus. Auch das Lesen von Einleitungen entsprechender Bücher kann hilfreich sein. Ziel ist es, die Größe des Themas zu bestimmen. Welche Unterthemen gehören beispielsweise zu dem Thema? In diesen Schritt sollte nicht zu viel Zeit investiert werden.

Schritt 2: Festlegen, was ich lernen will.

Man kann nicht alles lernen, daher sollte man den Fokus auf ein bestimmtes Thema richten. Möchte man beispielsweise etwas über digitale Fotografie lernen, so wäre ein Fokus alles über das Schießen von Porträt-Fotos zu lernen. Wichtig ist, dass man den Fokus auf ein einziges Thema richtet, denn wir können nicht mehrere Sachen auf einmal lernen.

Schritt 3: Das Lernziel festlegen.

Bezogen auf das Beispiel mit der digitalen Fotografie kann das bedeuten, alle Funktionen der eigenen Kamera beschrieben und nutzen zu können und erklären wann und warum man welche Funktion benutzt. Diese Lernziele sollten möglichst eindeutig sein. Denn so können wir später feststellen, ob wir dem Ziel näher kommen. Ein weiterer Vorteil ist, dass wir ein konkretes Ziel vor Augen haben, welches wir erreichen wollen.

Schritt 4: Quellen suchen.

Es ist wichtig, nicht nur mit einer Quelle zu lernen. Es gibt viele verschiedene Arten von Quellen wie Bücher, Videos, Blogs, andere Experten, Programmcode, Beispielprojekte oder Dokumentationen. In diesem Schritt sollten erstmal, ähnlich wie bei einem Brainstorming, alle Quellen, die in Frage kommen, zusammengetragen werden. Die Qualität der Quelle ist an dieser Stelle weniger relevant.

Schritt 5: Einen Lernplan erstellen.

Bei den meisten Themen bietet es sich an, bestimmte Unterthemen in einer bestimmten Reihenfolge zu lernen. Zum Erstellen dieses Planes kann man sich oft an den Inhaltsverzeichnissen der Bücher (aus Schritt 4) orientieren. Oder man schaut mit welcher Struktur andere das Thema erklären.

Schritt 6: Die Quellen filtern.

Viele Quellen werden sich thematisch überschneiden und meistens reicht auch die Zeit nicht aus, alle Quellen durchzuarbeiten. Deshalb ist es wichtig, die Quellen entsprechend zu filtern. Die ausgewählten Quellen sollten natürlich die im Lernplan ausgewählten Bereiche abdecken. In diesem Schritt sollte auch auf die Qualität der Quellen geachtet werden. Bei der Auswahl von Büchern kann es z.B. hilfreich sein, Amazon Bewertungen durchzulesen.

Die folgenden Schritte 7-10 sollten für jedes Modul aus dem Lernplan durchlaufen werden. Die Schritte folgen dem Prinzip „LDLT: learn, do, learn, teach“.

Schritt 7: Genug lernen, um anzufangen.

Wichtig ist, möglichst früh praktische Erfahrungen zu sammeln, denn wir lernen am besten, indem wir etwas tun. In diesem Schritt geht es darum, nur die grundlegenden Sachen zu lernen. Das kann beispielsweise das Durchlaufen eines „Hello-World-Beispiels“ sein oder das Einrichten der Entwicklungsumgebung. Eventuell reicht es auch schon, eine Kapitelzusammenfassung eines Buches zu lesen.

Schritt 8: Freies experimentieren.

Nutze deine Neugier und Kreativität, probiere etwas aus, bis du an einen Punkt kommst, an dem du nicht mehr weiterkommst. In diesem Schritt werden sich viele Fragen ergeben. Es kann hilfreich sein, diese aufzuschreiben.

Schritt 9: Genug lernen, um etwas sinnvolles zu tun.

Die Fragen, die sich im achten Schritt ergeben haben, sollen in diesem Schritt beantwortet werden. An dieser Stelle sollen die Quellen aus Schritt 4 intensiv genutzt werden. Der Focus sollte aber immer darauf liegen, die Fragen zu beantworten. Dass bedeutet eventuell nur einzelne Kapitel eines Buches anlesen, in denen man die Antwort auf eine Frage vermutet. Wichtig ist auch, zu prüfen ob man dem, in Schritt 3 definierten Ziel, näher kommt.

Schritt 10: Selbst erklären.

"Tell me and I forget. Teach me and I remember. Involve me and I learn." - Benjamin Franklin

Sobald wir versuchen das Gelernte jemand anderem zu erklären, werden wir merken, welche Themen von denen wir dachten, wir hätten sie verstanden, wir doch noch nicht verstanden haben. Es ist die beste Möglichkeit, das Gelernte zu überprüfen und Lücken zu füllen. Wichtig ist, das Wissen in eigene Worte zu fassen und das Ganze selbst zu Strukturieren. Möglich ist das beispielsweise, indem man ein YouTube Video erstellt, sich mit einem Freund oder Mitarbeiter unterhält, eine Präsentation erstellt oder Fragen in einem Forum beantwortet. Sobald wir selbst versuchen, etwas mit unseren eigenen Worten zu erklären, ordnen wir die unterschiedlichen Informationen in unserem Gehirn, so dass sie für uns Sinn ergeben. Erst dann können wir effektiv auf das Gelernte zurückgreifen.

Diese Schritte stellen sicherlich keine „magische Formel“ dar. Wenn man merkt, dass es so formal nicht funktioniert, sollte man die Schritte anpassen oder weglassen. Die Schritte an sich sind auch nicht wichtig, wichtig ist es, das Konzept dahinter zu verstehen. Nur dann kann man ein eigenes System entwickeln, um sich selbst effizient etwas beizubringen.

2.3.2 Mentor

Ein Mentor oder auch Trainer kann hilfreich sein, um neue Themen zu erlernen. Doch wie erkennt man einen geeigneten Mentor? Gute Mentoren sind meistens diejenigen, die die meisten Fehler durchlaufen haben. Allerdings muss der Mentor selbst das Thema nicht unbedingt beherrschen. Tiger Woods wird von jemandem trainiert, der selbst nicht so gut spielt wie er, aber ihm fallen Aspekte auf, die Tiger Woods nicht auffallen. Man sollte sich jemanden suchen, der bereits anderen geholfen hat, das zu erreichen, was man auch selbst erreichen möchte. Einen guten Mentor erkennt man auch oft daran, wie viele Personen er beeinflusst. Letztendlich muss man natürlich auch persönlich mit der Person zurechtkommen. Doch wo findet man so eine Person? Es gibt Portale, dort kann man für verschiedene Themen Mentoren bzw. Trainer mieten, doch man sollte sich eher im eigenen Umfeld umschauen. Eventuell kann ein Freund, ein Familienmitglied, ein Freund eines Freundes, ein Arbeitskollege oder eventuell auf der eigene Chef als Mentor fungieren. Doch selbst wenn man einen Mentor findet, heißt das noch lange nicht, dass er einem auch hilft. Erfolgreiche Personen sind oft beschäftigt und haben daher wenig Zeit. Eine Möglichkeit ist daher, immer etwas im Austausch anzubieten. Das kann beispielsweise schon ein Mittagessen sein, welches man für den Arbeitskollegen übernimmt. Außerdem ist es wichtig, nicht beim ersten „Nein“ aufzugeben. Man darf an dieser Stelle nicht zu nett sein und sollte wiederholt nachhaken.

Andersrum betrachtet kann und sollte man auch selbst die Rolle eines Mentors einnehmen. Im Prinzip kann das jeder tun. Jeder weiß bereits etwas, was andere versuchen zu lernen. Als Mentor muss man, wie bereits erwähnt, nicht perfekt sein. Oft hilft es dem Lernenden schon, einen anderen Blickwinkel einzunehmen oder eine zweite Meinung zu geben. Vorteil der Mentor-Rolle ist, dass man dabei i.d.R. am meisten lernt. Dazu kommt, dass die Leute, denen man hilft sich oft an einen erinnern und einem später dafür an anderer Stelle helfen. Ein großes Problem ist jedoch, dass man irgendwann nicht mehr allen helfen kann, da auch Zeit für die eigenen Aufgaben bleiben muss. Dann ist es wichtig denen zu Helfen, die wirklich Lust haben etwas zu lernen und die entsprechende Motivation mitbringen.

2.3.3 Lehren

Lehren ist der beste Weg, etwas zu lernen und wahrscheinlich der einzige Weg, etwas im Detail zu verstehen. Doch wir fühlen uns meist sehr unwohl, wenn wir daran denken zu lehren. Oft liegt das nicht daran, dass wir nicht lehren bzw. erklären können, sondern daran, dass wir nicht selbstbewusst genug sind. Wir möchten i.d.R. nur die Themen lehren, in denen wir selbst Experten sind. Doch um ein Experte in einem Thema zu werden, müssen wir zuerst lehren - ein Teufelskreis. Der Trick ist, viele von uns lehren, ohne es selbst zu bemerken. Lehren bedeutet nicht nur vor Gruppe von Leuten zu stehen und Themen zu erklären. Es geht vor allem darum, das Wissen zu Teilen. Wir haben alle schon einem Arbeitskollegen oder einem Kommilitonen etwas erklärt. Auch das ist Lehren. Weiter kann es hilfreich sein, einen eigenen Blog zu starten, Präsentationen im Unternehmen durchzuführen oder Videos bzw. Screencasts zu erstellen. Um Lehrer zu sein, braucht man keine Zertifikate und keinen Abschluss und man muss auch kein Experte sein. Wir alle sind bereits Lehrer.

2.3.4 Wissenslücken

Wir alle haben Wissenslücken und Schwächen, doch meistens fallen uns diese Lücken garnicht auf. Eine Möglichkeit Wissenslücken zu identifizieren, ist zu überlegen, wo man am meisten Zeit investiert. Meistens gibt es tägliche, wiederkehrende Aufgaben, die durch Wissenslücken verlangsamt werden. Ein Beispiel sind die Shortcuts der IDE. Wenn diese Shortcuts, die häufig benötigt werden, nicht bekannt sind, benötigt man für einfache Aufgaben wesentlich mehr Zeit. Eine weitere Möglichkeit um Wissenslücken aufzudecken, besteht darin, aktiv auf Verständnisprobleme zu achten und diese aufzulisten. Zusätzlich sollte man notieren, wie oft welches Verständnisproblem auftritt. Nicht jede Wissenslücke, die auftritt, muss unbedingt geschlossen werden. Aber anhand der Liste lassen sich Lücken finden, die besonders oft auftreten.

In einem zweiten Schritt geht es darum, die Wissenslücken zu schließen. Der schwerste Teil, nämlich das Identifizieren der Wissenslücken, ist bereits getan. Wichtig ist herauszufinden, was man konkret lernen muss. Es ist wenig hilfreich zu wissen, dass man beispielsweise schlecht in Physik ist. Doch wenn man weiß, dass man nich versteht, wie z.B. Federn funktionieren, lässt sich diese Lücke einfach schließen. Außerdem sollte man während eines Gespräches zeitnah nachfragen, wenn man etwas nicht versteht.

2.4 Produktivität (Productivity)

Wenn man Produktivität definieren will, kann man es mit einfachen Worten verdeutlichen: "Mach' deine Arbeit." bzw. "Tu' es."

Trotz der Trivialität dieses Imperativsatzes fällt es vielen Menschen dennoch schwer dies auch auszuführen. Dies ist zumeist Folge von Ablenkung wie z.B. Social Media Sites oder E-Mails und Mangel an Eigendisziplin.

Zudem ist produktives Arbeiten nicht unbedingt gleich effektives Arbeiten. Man kann sehr viel produzieren, was zu von einer Produktivität zeugt, und trotzdem nicht effizient sein, denn dafür muss man das richtige machen.

In den nachfolgenden Themen werden einige Faktoren und Techniken aufgezeigt, wie man die Produktivität nachhaltig steigern kann.

2.4.1 Fokus

Sich auf etwas fokussieren bedeutet, dass man sich einzig und allein auf eine einzige Sache konzentriert. Entsprechend bedeutet fokussiertes Arbeiten nichts anderes als dass man eine einzige Sache mit höchster Aufmerksamkeit bearbeitet. Demnach ist Fokus sozusagen das Gegenteil von Ablenkung, da diese uns Menschen die Aufmerksamkeit nimmt. Heutzutage ist die Welt voll mit Ablenkungen wie Social Media, Online Games oder TV-Sendungen, welche das fokussierte Arbeiten immens erschweren. Trotz dieser verlockenden Ablenkungen ist Fokus ein absolutes Muss für Produktivität, da dieser die Produktivität immens steigert.

Den Zustand von Fokus kann man jedoch nicht sofort erreichen. Es erfordert etwas Überwindungskraft.

Überwindung des "Initialschmerzes"

Der Initialschmerz ist eine entscheidende Phase, welche relativ unangenehm für jede Person ist, aber die ausgehalten werden muss, damit ein Fortarbeiten wahrscheinlicher ist. In dieser Phase ist man sehr anfällig gegen externe Ablenkungen. Daher ist es ratsam, Vorkehrungen zu treffen, damit solche Ablenkung erst gar nicht zustande kommt. Beispielsweise kann man vor Beginn seine Mails abchecken, ablenkende Internetseiten schließen, Zimmer abschließen (Sofern man ein eigenes Zimmer hat) oder Mitarbeitern signalisieren, dass man die nächste Zeit nicht gestört werden will). Im Regelfall dauert diese Phase 5-10 Minuten an. Danach ist man auch gegen Ablenkungen resistenter.

Aufrechterhaltung des Fokus & Handhabung mit Unterbrechungen

Es kann vorkommen, dass man plötzlich eine Nachricht per E-Mail oder Chatclient bekommt oder von jemand angerufen oder angesprochen wird (z.B. Kollegen oder Familienangehörigen), und dementsprechend seinen Fokus auf die eigentliche Aufgabe verlieren kann. Danach muss man sich erneut zu seinem Fokus auf die Arbeit ringen, was Zeit und dementsprechend Produktivität kostet. Deshalb sollte man jegliche Form von Unterbrechungen vermeiden. Beispielsweise kann man seinem Umfeld vermitteln nicht gestört werden zu wollen und jegliche Form der Kommunikation zu ignorieren.

2.4.2 Produktivitätsplanung

Es macht Sinn sein Leben systematisch zu planen - sowohl in Quartalen als auch in Monaten, Wochen und Tagen. Tools und Techniken wie Kanban (kanbanflow) oder Trello können dabei sehr unterstützend wirken.

Quartalsplanung

Hierunter fallen Aufgaben mit langer Zeitspanne oder Großprojekte. Dabei werden kleinere Aufgaben notiert, welche wöchentlich bzw. täglich im Rahmen dieses Projekts gemacht werden müssen. Es macht daher Sinn, an dieser Stelle der Planung eine grobe Granulierung der Aufgabe/des Projekts anzusetzen, sofern nicht erfolgt.

Monatsplanung

Hier wird der grobe Arbeitsaufwand der abgeschätzt. Aufgaben, welche im Zusammenhang mit den anstehenden großen Aufgaben/Projekten aus der Quartalsplanung stehen, werden dabei bei der Planung berücksichtigt. Zudem werden Dinge geplant, welche monatlich erledigt werden müssen (z.B. Deadlines von Projekten).

Wochenplanung

Planungen von Dingen die wöchentlich erledigt werden müssen (Projektmappe, Haushalt, Garten, Meetings).

Tagesplanung

Dinge, die persönlich als wichtig erachtet werden, machen (z.B. Trainingseinheiten), um Ablenkungen zu vermeiden und fokussiertes Arbeiten zu ermöglichen (s.o. "Fokus"). Danach werden die Aufgaben, welche für den Tag anstehen evaluiert und entsprechend ihrer Wichtigkeit zeitlich angeordnet, damit die wichtigsten Aufgaben garantiert erledigt werden.

Urlaub

Es ist sinnvoll, von Zeit zu Zeit Urlaub von der Produktivitätsplanung und jeweils angewandten Produktivitätssystem zu nehmen, um dessen Wichtigkeit bzgl. der Produktivität zu realisieren. Zudem ist es gesundheitlich nicht ratsam durchgehend wie eine Maschine zu arbeiten (siehe Burnout).

2.4.3 Promodoro Technik

Die Promodoro-Technik wurde in dem späten 80'ern von Francesco Ciritto entwickelt und soll System in die Produktivität bringen.

Prinzipiell ist die Technik relativ einfach:

1. Zuerst erstellt man einen Arbeitsplan für den ganzen Tag.
2. Danach widmet man sich der zuerst geplanten Aufgabe. Dabei arbeitet man 25 Minuten durch und fokussiert sich nur auf die eine Aufgabe ohne sich unterbrechen zu lassen.
Dies ist ein sogenanntes Promodoro.
3. Sofern man früher die Aufgabe beenden kann, wird ein sogenanntes "Overlearning" angewandt. Das kann beispielsweise zusätzliche Optimierungen bei einem Programm oder ein nochmaliges Überfliegen der schon gelesenen Materialien beim Lernen sein.
 - i. Nach dem Promodoro nimmt man sich 5 Minuten Zeit um Pause zu machen.
Danach widmet man sich der Aufgabe wieder, sofern sie noch nicht erledigt ist, oder man beginnt mit der nächstwichtigen Aufgabe.
 - ii. Alle 4 Promodori macht man eine längere Pause (ca. 15 Minuten).

Effektives Anwenden

Viele Leute versuchen diese Technik anzuwenden, ohne ihren richtigen Potential zu erkennen, weswegen sie meistens diese nach kurzer Zeit verwerfen. Eines der Potentiale liegt darin, dass man mit dieser Technik die zu machende Arbeit messen und abschätzen kann. Das Messen der getane Arbeit in Promodori ermöglicht einen besseren Überblick, wieviel man am Tag gearbeitet und erreicht hat. Zudem wird auch das persönliche Maximum an Arbeit besser ersichtlich.

Viele Leute, die diese Technik in dieser Art angewendet haben, waren danach viel motivierter. John Z. Sonmez, der Verfasser des zusammengefassten Buches, schafft zwischen 50-55 Promodori pro Woche. Diese Zahl ist dennoch kein Richtwert und kann von Person zu Person variieren. Zudem ist er der Meinung, dass eine 40-Stundenwoche nicht automatisch 80 Promodori bedeuten muss.

Quotensystem

John Z. Sonmez benutzt zusätzlich ein Quotensystem zu der Promodoro-Technik, um seinen Produktivitätsfortschritt messbar zu gestalten. Er verspricht sich davon viel konsistenter Fortschrittsergebnisse. Prinzipiell besteht das Quotensystem aus folgenden Schritten:

1. Such dir eine Aufgabe aus.
2. Bestimme den Zeitraum, in welcher diese Aufgabe erledigt sein und wiederholt werden

muss.

3. Bestimme, wie oft diese Aufgabe während dieses Zeitraums gemacht werden soll (Quote).
4. Umsetzen.
5. Quote anpassen, sofern die Quote zu tief bzw. hoch angelegt wurde.

John z. Sonmez produziert nun durch Anwendung dieses Quotensystems wesentlich mehr als zuvor und berichtet positiv von der Anwendung des Quotensystems.

Eigenverantwortung

Die Entwicklung von Eigenverantwortung ist wichtig für die Produktivität. Dabei ist Selbstbeherrschung und Selbstdisziplin der Schlüssel zur Selbstmotivation. So kann man damit anfangen, sein eigenes Leben Struktur zu verleihen, indem man sein Leben inkl. der dazugehörigen Aktivitäten plant. Dabei ist es nicht verwerflich andere Personen um Hilfe zu bitten (Ernährungs- und Sportplan von Ernährungsberatern und Fitnesstrainern als Beispiel). Auch Aktionen in der Öffentlichkeit können der Eigenverantwortung Nachdruck verleihen. Beispielsweise fallen dort fehlende Aktionen schneller auf, was zu einem Pflichtgefühl gegenüber der Öffentlichkeit mündet und somit einen Motivationsdruck zufolge haben kann.

**Multitasking

Es gibt Aufgaben, die man miteinander kombinieren kann. In den meisten Fällen ist es jedoch kein wirkliches Multitasking, sondern Task-Switching, was prinzipiell das hin- und herwechseln zwischen den Aufgaben bedeutet.

Task-Switching wiederum ist für die Produktivität von Nachteil, da man immer wieder den Fokus wechseln muss (siehe Fokus). Richtiges Multitasking ist, wenn man zwei oder mehr Sachen zur gleichen Zeit macht. Ein Mann der joggt während er Musik hört, betreibt beispielsweise wirkliches Multitasking.

Dabei profitiert er durch die Musik, indem er zum Takt der Musik läuft und somit länger laufen kann. Ähnlich verhält es sich bei anderen Multitaskings auch: Zwei oder mehr Sachen gleichzeitig machen kann unter anderem die Produktivität steigern. Jedoch kann man nicht beliebige Aufgaben zum Multitasking verknüpfen. Bewährt hat sich das Kombinieren von Aufgaben, die wenig Verstand benötigen, mit Aufgaben, die mentalen Fokus benötigen.

Stapeln von Aufgaben

Aufgaben, die zwar nicht gleichzeitig erledigt werden können, können jedoch zusammengefasst und stapelweise abgearbeitet werden. Das verhindert, dass man zwischendurch den Fokus verliert, und steigert entsprechend die Produktivität.

Beispielsweise kann man einmal am Tag alle Emails abchecken und antworten, anstatt dass man simultan bei jeder eintreffenden Mail seine eigentliche Aufgabe vernachlässigt.

Burnout und dessen Lösung

Burnout ist eines der größten Probleme bzgl. Produktivität. Mit der Zeit Gewöhnt man sich an seiner Arbeit, sodass es nichts Besonderes mehr ist oder gar verabscheut wird.

Entsprechend schwindet die Motivation und Interesse. Dann kommt irgendwann der Punkt, an dem man sowohl physisch als auch psychisch erschöpft ist. Man stößt umgangssprachlich an die Mauer. Das ist meist der Punkt an dem die Produktivität gering bis gar nicht mehr auftritt und die meisten aufgeben.

Die Lösung zu diesem Problem ist durchhalten, bis man die Mauer überwunden hat. Denn nachdem man diese überwunden hat, steigt sowohl die Motivation als auch das Interesse wieder. Die Lösung liegt quasi darin, das Burnout zu ignorieren. Unter anderem muss man auch realisieren, was auf der anderen Seite der Wand wartet.

Sehr hilfreich kann auch eine Struktur im Leben sein. Das Selbstaufstellen von Regeln kann für den nötigen Antrieb im Leben sorgen.

Zeitverschwender

Zeitverschwendungen ist ein Teil des Menschen. Man kann diese nicht komplett unterdrücken. Man kann aber einige der größten Zeitverschwender unterbinden. Dabei kann das Verfolgen jeglicher Zeitnutzung in Form von Protokoll hilfreich zur Identifikation dieser sein. Tools wie „RescueTime“ wirken dabei unterstützend.

Beispielsweise ist eines der größten Zeitverschwender TV schauen. Im Durchschnitt verbringt der Mensch ca. 40 Stunden mit Fernsehen. Wenn man bedenkt, dass man 8 Stunden zum Schlafen, 8 Stunden zum Arbeiten und 2 Stunden zum Essen braucht und die restliche Zeit für die ganze Woche betrachtet, bleiben dem Menschen nur 2 freie Stunden für eine ganze Woche nach Abzug dieser Zeit. Man sollte demnach in Erwägung ziehen, das Fernsehen aufzugeben oder auf ein Minimum zu reduzieren, damit mehr Zeit für sinnvollere Tätigkeiten bleibt (z.B. Trainieren).

Aber auch Zeitverschwender wie Social Media, Nachrichten, Kaffeepausen, Videospiele oder auch unnötige Meetings kann man optimieren, indem man diese reduziert oder zu einem Zeitpunkt zusammenfasst.

Wichtigkeit von Routinen

Personen, die vielleicht nur eine Aufgabe machen, weil es sein muss, werden meist nicht so schnell diese erledigen als Personen, die es sich zur Routine gesetzt haben, diese zu erledigen. Routinen können dementsprechend die Produktivität immens erhöhen.

Daher ist es sinnvoll, Routinen zu entwickeln, die bei der Aufgabenbewältigung unterstützend wirken. Um den Überblick nicht zu verlieren, ist es auch von Vorteil, diese in Tagesform zu planen.

Gewohnheiten

Genau wie Routinen können Gewohnheiten fördernd für das Ziel sein. Jedoch gibt es gute und schlechte Gewohnheiten. Entsprechend können schlechte Gewohnheiten hinderlich für das Ziel sein (permanenter E-Mailcheck).

Gewohnheiten werden durch 3 Dinge definiert:

1. Auslöser
2. Routine
3. Belohnung

Zum Beispiel kann man den E-Mailcheck nehmen: Der Auslöser wäre die E-Mailbenachrichtigung in Form eines Pop-Ups, die Routine wäre das Abchecken der E-Mail und die Belohnung wäre die Gewissheit über den Inhalt dieser E-Mail.

Abgewöhnen von schlechten Gewohnheiten

Man kann entweder versuchen, die schlechten Gewohnheiten zu vermeiden, oder diese sogar durch andere zu ersetzen. Bei letzteren muss man sich im Klaren sein, was diese Gewohnheit auslöst, und wie man diesen Auslöser für eine andere Gewohnheit verwenden kann.

Entwicklung von neuen Gewohnheiten

Die Entwicklung von neuen Gewohnheiten kann die Produktivität immens fördern, da diese eine unterstützende Rolle für Schlüsselroutinen für die zu bearbeitende Aufgabe haben kann. Denn das Planen von Routinen bedeutet nicht unbedingt, dass diese erfolgreich umgesetzt werden.

Aufgaben granulieren

Große Projekte können sehr einschüchternd wirken, da viele Leute nicht abschätzen können, wie groß der Arbeitsaufwand und wie viel Zeit das Projekt in Anspruch nehmen kann.

Das Herunterbrechen in mehrere kleine Aufgaben kann dabei helfen, Zeit- und Arbeitsaufwand besser zu berechnen.

Wert harter Arbeit: Wieso vermeiden wir sie?

Der Mensch hat die Tendenz, harte Arbeit, die ihm wichtig erscheint, zu vermeiden oder bei dessen Umsetzung zu zögern. Selbst bei der Umsetzung fühlt er sich meist nicht wohl dabei oder die Arbeit ist ihm zumeist zu langweilig.

Dennoch muss man lernen, sich hinzusetzen und die Arbeit zu machen, die man nicht gern macht. Die Entscheidung liegt dabei nur bei ihm, ob etwas gemacht wird oder nicht. Es hängt dementsprechend von der eigenen Willenskraft ab.

Jedes Tun ist besser als nix zu tun

Nichtstun ist der schlimmste Produktivitätskiller. Die meisten Gelegenheiten und Möglichkeiten werden dadurch verschwendet, indem man nichts tut. Angst vor einer falschen Entscheidung spielt dabei eine sehr große Rolle.

2.4.4 Weitere Hinweise

Eigenverantwortung

Die Entwicklung von Eigenverantwortung ist wichtig für die Produktivität. Dabei ist Selbstbeherrschung und Selbstdisziplin der Schlüssel zur Selbstmotivation. So kann man damit anfangen, sein eigenes Leben Struktur zu verleihen, indem man sein Leben inkl. der dazugehörigen Aktivitäten plant. Dabei ist es nicht verwerlich andere Personen um Hilfe zu bitten (Ernährungs- und Sportplan von Ernährungsberatern und Fitnesstrainern als Beispiel). Auch Aktionen in der Öffentlichkeit können der Eigenverantwortung Nachdruck verleihen. Beispielsweise fallen dort fehlende Aktionen schneller auf, was zu einem Pflichtgefühl gegenüber der Öffentlichkeit mündet und somit einen Motivationsdruck zufolge haben kann.

Multitasking

Es gibt Aufgaben, die man miteinander kombinieren kann. In den meisten Fällen ist es jedoch kein wirkliches Multitasking, sondern Task-Switching, was prinzipiell das hin- und herwechseln zwischen den Aufgaben bedeutet.

Task-Switching wiederum ist für die Produktivität von Nachteil, da man immer wieder den Fokus wechseln muss (siehe Fokus). Richtiges Multitasking ist, wenn man zwei oder mehr Sachen zur gleichen Zeit macht. Ein Mann der joggt während er Musik hört, betreibt beispielsweise wirkliches Multitasking.

Dabei profitiert er durch die Musik, indem er zum Takt der Musik läuft und somit länger laufen kann. Ähnlich verhält es sich bei anderen Multitaskings auch: Zwei oder mehr Sachen gleichzeitig machen kann unter anderem die Produktivität steigern. Jedoch kann man nicht beliebige Aufgaben zum Multitasking verknüpfen. Bewährt hat sich das Kombinieren von Aufgaben, die wenig Verstand benötigen, mit Aufgaben, die mentalen Fokus benötigen.

Stapeln von Aufgaben

Aufgaben, die zwar nicht gleichzeitig erledigt werden können, können jedoch zusammengefasst und stapelweise abgearbeitet werden. Das verhindert, dass man zwischendurch den Fokus verliert, und steigert entsprechend die Produktivität. Beispielsweise kann man einmal am Tag alle Emails abchecken und antworten, anstatt dass man simultan bei jeder eintreffenden Mail seine eigentliche Aufgabe vernachlässigt.

Burnout und dessen Lösung

Burnout ist eines der größten Probleme bzgl. Produktivität. Mit der Zeit Gewöhnt man sich an seiner Arbeit, sodass es nichts Besonderes mehr ist oder gar verabscheut wird. Entsprechend schwindet die Motivation und Interesse. Dann kommt irgendwann der Punkt, an dem man sowohl physisch als auch psychisch erschöpft ist. Man stößt umgangssprachlich an die Mauer. Das ist meist der Punkt an dem die Produktivität gering

bis gar nicht mehr auftritt und die meisten aufgeben.

Die Lösung zu diesem Problem ist durchhalten, bis man die Mauer überwunden hat. Denn nachdem man diese überwunden hat, steigt sowohl die Motivation als auch das Interesse wieder. Die Lösung liegt quasi darin, das Burnout zu ignorieren. Unter anderem muss man auch realisieren, was auf der anderen Seite der Wand wartet.

Sehr hilfreich kann auch eine Struktur im Leben sein. Das Selbstauferlegen von Regeln kann für den nötigen Antrieb im Leben sorgen.

Zeitverschwender

Zeitverschwendungen ist ein Teil des Menschen. Man diese nicht komplett unterdrücken. Man kann aber einige der größten Zeitverschwender unterbinden. Dabei kann das Verfolgen jeglicher Zeitnutzung in Form von Protokoll hilfreich zur Identifikation dieser sein. Tools wie „RescueTime“ wirken dabei unterstützend.

Beispielsweise ist eines der größten Zeitverschwender TV schauen. Im Durchschnitt verbringt der Mensch ca. 40 Stunden mit Fernsehen. Wenn man bedenkt, dass man 8 Stunden zum Schlafen, 8 Stunden zum Arbeiten und 2 Stunden zum Essen braucht und die restliche Zeit für die ganze Woche betrachtet, bleiben dem Menschen nur 2 freie Stunden für eine ganze Woche nach Abzug dieser Zeit. Man sollte demnach in Erwägung ziehen, das Fernsehen aufzugeben oder auf ein Minimum zu reduzieren, damit mehr Zeit für sinnvollere Tätigkeiten bleibt (z.B. Trainieren).

Aber auch Zeitverschwender wie Social Media, Nachrichten, Kaffeepausen, Videospiele oder auch unnötige Meetings kann man optimieren, indem man diese reduziert oder zu einem Zeitpunkt zusammenfasst.

Wichtigkeit von Routinen

Personen, die vielleicht nur eine Aufgabe machen, weil es sein muss, werden meist nicht so schnell diese erledigen als Personen, die es sich zur Routine gesetzt haben, diese zu erledigen. Routinen können dementsprechend die Produktivität immens erhöhen.

Daher ist es sinnvoll, Routinen zu entwickeln, die bei der Aufgabenbewältigung unterstützend wirken. Um den Überblick nicht zu verlieren, ist es auch von Vorteil, diese in Tagesform zu planen.

Gewohnheiten

Genau wie Routinen können Gewohnheiten fördernd für das Ziel sein. Jedoch gibt es gute und schlechte Gewohnheiten. Entsprechend können schlechte Gewohnheiten hinderlich für das Ziel sein (permanenter E-Mailcheck).

Gewohnheiten werden durch 3 Dinge definiert:

1. Auslöser
2. Routine
3. Belohnung

Zum Beispiel kann man den E-Mailcheck nehmen: Der Auslöser wäre die E-Mailbenachrichtigung in Form eines Pop-Ups, die Routine wäre das Abchecken der E-Mail und die Belohnung wäre die Gewissheit über den Inhalt dieser E-Mail.

Abgewöhnen von schlechten Gewohnheiten

Man kann entweder versuchen, die schlechten Gewohnheiten zu vermeiden, oder diese sogar durch andere zu ersetzen. Bei letzteren muss man sich im Klaren sein, was diese Gewohnheit auslöst, und wie man diesen Auslöser für eine andere Gewohnheit verwenden kann.

Entwicklung von neuen Gewohnheiten

Die Entwicklung von neuen Gewohnheiten kann die Produktivität immens fördern, da diese eine unterstützende Rolle für Schlüsselroutinen für die zu bearbeitende Aufgabe haben kann. Denn das Planen von Routinen bedeutet nicht unbedingt, dass diese erfolgreich umgesetzt werden.

Aufgaben granulieren

Große Projekte können sehr einschüchternd wirken, da viele Leute nicht abschätzen können, wie groß der Arbeitsaufwand und wie viel Zeit das Projekt in Anspruch nehmen kann.

Das Herunterbrechen in mehrere kleine Aufgaben kann dabei helfen, Zeit- und Arbeitsaufwand besser zu berechnen.

Wert harter Arbeit: Wieso vermeiden wir sie?

Der Mensch hat die Tendenz, harte Arbeit, die ihm wichtig erscheint, zu vermeiden oder bei dessen Umsetzung zu zögern. Selbst bei der Umsetzung fühlt er sich meist nicht wohl dabei oder die Arbeit ist ihm zumeist zu langweilig.

Dennoch muss man lernen, sich hinzusetzen und die Arbeit zu machen, die man nicht gern macht. Die Entscheidung liegt dabei nur bei ihm, ob etwas gemacht wird oder nicht. Es hängt dementsprechend von der eigenen Willenskraft ab.

Jedes Tun ist besser als nix zu tun

Nichtstun ist der schlimmste Produktivitätskiller. Die meisten Gelegenheiten und Möglichkeiten werden dadurch verschwendet, indem man nichts tut. Angst vor einer falschen Entscheidung spielt dabei eine sehr große Rolle.

2.7 Spirit

Im Kapitel Spirit geht es um den Geist des Menschen und wie er mit seinem Körper verbunden ist. Er teilt das Kapitel in verschiedene Unterkapitel ein. Die Unterkapitel befassen sich mit Fähigkeiten, die man im Leben braucht. So beschäftigt er sich mit dem Verstand, der mentalen Einstellung, positiven Denken, Liebe und Ausdauer.

2.7.1 How the mind influences the body

Der Autor erklärt, dass wir nicht einfache Maschinen sind, sondern Menschen. Der Geist gibt uns die Möglichkeit Erfolg zu haben oder uns scheitern zu lassen. (S. 396)

Er beschreibt, dass der Geist einen machtvollen Einfluss auf den Körper hat und das wir lernen müssen ihn zu beherrschen, wenn wir sogar den kleinsten Plan in die Tat umsetzen wollen.

Dies ist laut dem Autor nicht einfach. Man kann zum Beispiel glauben, dass Elefanten pink sind. Allerdings wird selbst der überzeugendste Beweis nicht verhindern, dass man an das glaubt, was man gewohnt ist. Das heißt natürlich nicht, dass man sich davon nicht überzeugen ließe.

Er behauptet, dass wir Opfer eines biologischen Prozesses unseres Kopfes wären.

Allerdings sind wir keine Tiere, sondern Menschen. Das heißt, wir haben die Möglichkeit den biologischen Prozess in unsere Richtung zu lenken. (S. 397-398)

Weiterhin stellt er die Frage, ob die physische Welt sich verändert, um die eigene Wahrnehmung zu erfüllen.

Die Antwort beantwortet die Frage damit, dass der Glaube die Macht hat die eigene Realität zu formen. Es ist ein indirektes Formen, welches dazu den eigenen Körper benötigt. So besitzen wir alle die Möglichkeit die physische Welt direkt zu manipulieren, sobald wir das erste Neuron feuern.

Er widerspricht der Meinung einiger Leute, die behaupten, dass wir eine Zusammensetzung von chemischen Elementen sind, die unsere Umwelt beeinflussen, eine Art lebenslanger Autopilot, eine Kette von chemischen Reaktionen, die auf unsere Umgebung basiert. Er wirft dabei folgende Fragen auf: Wie ist es möglich, dass wir sein Buch lesen können? Wie ist es möglich, dass er es schreiben kann? Seine Antwort ist die Freiheit der Wahl, der sogenannte "freie Wille".(S. 399)

Der Autor definiert den Verstand als nicht physischen Teil des Körpers abgegrenzt zum unteren Teil des Körpers und dem Gehirn. Er meint, dass es möglich ist durch Drogen unseren Verstand zu verändern. Dadurch beeinflusst unser Verstand unseren Körper in Richtungen, die wir nicht kontrollieren können.

Nach seiner Meinung gibt es eine Menge Formen und Philosophien. Die Populärste ist, dass negative Gedanken zu negativen Ergebnissen führen und umgekehrt genauso. Er behauptet von sich, dass er ein praktischer Mensch ist, aber er das Vorhandensein einer mystischen Komponente nicht ausschließen würde. (S. 400)

Abschließend meint er, dass unser Bewusstsein und unser Glaube einen positiven oder negativen Einfluss auf unser Leben haben können. Er will uns in seinem Text praktische Beispiele zeigen, wie wir unser Verstand so schärfen können, dass wir die höchste Produktivität für uns heraus holen. (S. 401)

2.7.1 How the mind influences the body

2.7.2 Having the right mental attitude: Rebooting

Im nächsten Unterkapitel wird beschrieben, dass positives Denken ermöglicht, die eigene Lebenszeit zu verlängern, Freundschaften zu entwickeln, ein höheres Einkommen zu haben und körperlich gesund zu sein. Er widerspricht der Behauptung, dass positives Denken destruktiv ist und man realistisch bleiben sollte. Für ihn ist positives Denken die ultimative Form von Realismus, weil der eigene Glaube die Möglichkeit hat, seine Realität zu verändern, so dass man kein Opfer seiner Umstände wird. Er meint mit einer positiven Einstellung lebt man nicht in einer Fantasiewelt, abgetrennt von der Realität, sondern in einer optimalen Welt, wo jeder seine best mögliche Zukunft sieht, und diese real werden lassen will.

Er meint, dass eine Person mit einer positiven Einstellung dazu tendiert, mehr Situationen gut statt schlecht zu betrachten und zwar nicht weil die Situationen objektiv gut oder schlecht sind, sondern weil er erkennt, dass er selbst wählen kann, wie er sie betrachten möchte. (S. 401-403).

Der Autor weiß, dass seine Einstellung seine Performanz beim Arbeiten beeinflusst. Er erkennt dies an seiner eigenen Produktivität. Wenn er eine positive Einstellung hat, ist es für ihn leichter mit Hindernissen umzugehen und Herausforderungen zu akzeptieren, als wenn er mit einer negativen Einstellung umgeben ist.

Er sagt aber auch, dass es nicht reicht, positiv sein zu wollen. Es ist nicht leicht seine Sicht auf die Welt von einer negativen in eine positive zu ändern. (S.405)

Er ist der Meinung, wenn man seine Einstellung ändern will, muss man erstmal seine Gedanken ändern. Und wenn man seine Gedanken ändern will, muss man seinen Ansatz zu Denken ändern. Der Ansatz zu Denken erfolgt durch die eigenen Gewohnheiten und dies führt dazu, dass man manche Wege im Leben signifikant ändern bzw. eine andere Gewohnheit entwickeln sollte. Er sagt, dass es schwierig ist über jedes Ereignis positiv zu denken, aber man selbst die Macht hat positive Gedanken zu erschaffen. Der Schlüssel dafür ist dies aktiv und ernsthaft durch den Tag zu versuchen, sich selbst daran zu erinnern, dass man seine sofortige Reaktion auf eine Situation nicht kontrollieren kann, sondern man kontrollieren sollte, wie man über die entstandene Erfahrung zu denken wählt.

Je mehr man diese Art von Denken übt, desto mehr positive Bilder entwickelt man und lässt dies zur Gewohnheit werden. Nach einiger Zeit ist man eher bereit, einen Unfall oder ein Missgeschick in einer positiven Art und Weise zu beantworten. Man kann sein Gehirn trainieren, die Dinge aus einer positiven Perspektive statt aus einer negativen zu betrachten. Weiterhin erwähnt er, dass manche Studien zeigen, dass Leute, die meditieren, eher bereit sind positive Emotionen zu empfinden. Also sollte man zu meditieren versuchen, um sein positives Mojo wachsen zu lassen. Auch findet er, dass es einfacher ist positiv zu sein, wenn man ab und zu Spaß hat. (S. 406)

Zusammenfassend meint er, dass positives Denken nicht als Chance kommt und nicht

etwas ist, was man über Nacht erzwingen kann. Es braucht eine gewisse Anstrengung seinen Verstand in eine positive Richtung zu lenken. Aber es lohnt sich. Man genießt nicht nur das Leben stärker, sondern ermöglicht auch den umgebenden Anderen das Leben zu genießen. (S.407)

2.7.3 Building a positive self-image: Programming your brain

Als nächstes spricht er davon, dass man lernen muss sein Gehirn so zu programmieren, dass man seine Ziele erreichen kann. Die wahre Schlacht richtet sich Mittelmäßigkeit und beginnt im Gehirn. Was man über sich selbst denkt hat die Macht einen zu limitieren oder voranzubringen. Er will zeigen auf welche Weise man ein positives Selbstbild entwickelt, das uns erlaubt im Gehirn ein Autopilot zu entwickeln, um seine Ziele zu erreichen. (S.408) Seiner Meinung nach ist das Selbstbild die Sicht, die man über sich entwickelt, ohne die Dinge die Andere über einem sagen. Es ist möglich sich seines wirklichen Selbstbildes nicht bewusst zu sein, weil dies sehr weit im eigenen Unterbewusstsein verankert ist. Tief innen haben wir alle ein Bild von uns, welches die ultimative Reflektion der Sicht unseres Gehirns auf unsere Wahrnehmung ist. Dieses Selbstbild ist machtvoll, weil unser Gehirn uns nicht erlaubt, etwas zu tun, was gegen unsere eigene Beurteilung ist. Es ist auch schwierig sich dazu zu überwinden, einfach weil man sich nicht bewusst ist, dass diese Grenzen existieren. Der Autor unterscheidet zwischen Dingen, die in unserer DNA verankert sind, unsere physischen Charakterzüge, und den Dingen, die wir in uns selbst manifestiert haben, so könnte man zum Beispiel faul, nicht gut in Mathe, schlecht im Umgang mit Leuten, schüchtern, reserviert oder aufmerksamkeitssuchend sein.

Er meint auch, wenn einem in der Kindheit eine Charaktereigenschaft nachgesagt wurde, auch wenn sie bis dahin nicht zutreffend war - aber in dem Moment der Erwähnung verinnerlicht wurde. (S.408-409)

Er beschreibt, dass wir die Macht haben unser Selbstbild zu ändern. Das Konzept dahinter ist das ständige Vortäuschen einer gewünschten Eigenschaft bis man sie verinnerlicht. Er sagt, dass es ein einfaches Konzept ist und das wir nur glauben, dass es hart ist. Wir hätten ein Teil in uns selbst, das krank und sadistisch ist und unsere Schwächen und Grenzen als kritischen Teil ins uns hervorhebt. Er sagt auch, dass unser Unterbewusstsein an unserem Selbstbild festhält und dass wir selbst den Willen haben müssen, unser Selbstbild zu ändern.

Er erzählt auch, dass die Kleidung, die wir tragen, einen Teil unseres Charakters ausmacht. Es fällt ihm auf, dass wir gerne nach der Art unserer Kleidung handeln und dies auch mit in unser Selbstbild einfließen lassen.

Er beschreibt wie er selbst von einer schüchternen und unathletischen Person zu einer athletischen und sozialen Person wurde, indem er die Kontrolle über sein Selbstbild übernommen hat, um so dafür zu sorgen, dass es in seiner Vorstellung für ihn arbeitet statt gegen ihn. (S. 409-411)

Im weiteren Kontext schildert er wie man sein Gehirn darauf trainiert sich Ziele zu setzen und vorzutäuschen, welche Person man sein möchte, um diese schließlich zu werden. Man soll seine Aufmerksamkeit auf sich selbst richten und nicht auf das was andere Leute von

einem wollen. Man soll positive Bestätigungen in seinem Alltag suchen und mental an sich selbst glauben. Man soll aufpassen, was man sagt, weil das Unterbewusstsein immer noch ein kleines Kind ist, das auf die eigene Stimme hört. (S.412-413)

2.7.4 Love and relationships: Computers can't hold your hand

Im nächsten Unterkapitel schreibt der Autor über Liebe und Beziehungen. Er beschreibt ein typisches Stereotyp von Softwareentwickler, die nerdig, allein und einsam sind. Er behauptet, dass Liebe und Beziehungen komisch sind. Er sieht die Liebe als Katz und Maus Spiel. Es gibt eine Person, die jagt und eine andere, die gejagt wird. Dies ist kein Problem, solange es abwechselnd und nicht einseitig ist. Weiterhin erwähnt er, dass wenn jemand verbissen versucht eine Beziehung zu finden, er in Verzweiflung versinkt und es hart ist, aus dieser herauszukommen. Auch neigen Leute dazu ihre Gefühle der Verzweiflung und der Einsamkeit dem Rest der Welt über soziale Medien mitzuteilen. Dies macht sie aber laut dem Autor unattraktiv und andere fangen an sie zu meiden. (S.414-415)

Er kann den Gedanken verstehen, dass man ehrlich über sich selbst und seine Gefühle sein will, er aber fragt sich, ob dies auch funktioniert. Er meint, dass man realisieren sollte, dass man ein Spiel spielt und eine Spielstrategie zum Gewinnen finden sollte. Er begründet dies damit, dass viele Leute nur das haben wollen, was sie nicht haben können und nicht das was leicht verfügbar ist. Das heißt je einfacher man zu haben ist, desto weniger wird man gewollt. Er sagt, dass das Leben ein großer Spielplatz ist - je mehr man jemanden jagt, desto mehr rennt er weg.

Auch hält er es nicht für eine gute Strategie zuhause rumzusitzen und auf die große Liebe zu warten. Seiner Meinung nach ist das beste sich so zu verhalten, dass man auf eine Person selbstsicher zugeht, ihr zeigt, dass man sich gut fühlt, dass man niemanden braucht um glücklich zu sein, aber dennoch Interesse an dieser Person hat.

Nach seiner Meinung soll man erkennen lassen, dass es ein Vorteil für die Person ist, wenn sie mit einem zusammenkommt. Aber man sollte auch nicht sich selbst als Geschenk Gottes ausgeben. Man soll genug Respekt für sich selbst zeigen und dort auftauchen wo man gewollt ist und sich nur mit Leuten abgeben, die einen mögen. Das heißt nicht, dass Erfolg garantiert ist, aber man hat eine bessere Chance seine Liebe zu finden, wenn man die Psychologie des Wegrennen und Jagens versteht und anwendet. Das gleiche gilt für ihn auch bei einem Jobinterview. (S.415-417)

Er hält es für ein Nummernspiel. Es gibt laut ihm viele Arten von Leuten, mit vielen verschiedenen Vorlieben. Er sagt, dass es viele potentielle Partner für einen geben wird. Man soll auf eine Person zugehen, die mit einem zusammen sein möchte und nicht auf eine, welche es nicht will. Es gibt auch genug andere. (S.417-418)

2.7.5 Facing failure head-on

Im letzten Unterkapitel schreibt er über die seiner Meinung nach wichtigste Fähigkeit: Die Ausdauer. Er hält alle anderen Fähigkeiten für wertlos, wenn es an der Ausdauer scheitert. Software zu entwickeln ist schwierig und so ist es laut ihm wichtig Ausdauer zu haben. Er sagt, dass die Angst vor Fehlern ein innerer Instinkt vieler Leute ist. Diese Angst ist gut, wenn unser Leben bedroht ist, aber schlecht, wenn ein Fehler nur harmlos ist. Wir versuchen das zu tun, was wir können und vermeiden Dinge für die wir inkompotent sind oder nicht die entsprechende Fähigkeit zu haben. Er hält es für einen Schutz des eigenen fragilen Ego. Er glaubt, dass wir denken, dass dies eine Reflektion unseres persönlichen Versagens ist. Der Autor meint, dass das Verletzen des eigenen Egos ein Missverständnis der Natur eines Fehlers ist, weil wir nicht dazu trainiert sind Fehler als Weg, in vielen Fällen als einzigen Weg, zum Erfolg zu sehen. (S.424-425)

Er beschreibt, dass Fehler nicht dasselbe wie Niederlagen sind. Laut ihm sind Fehler temporär, Niederlagen sind permanent. Eine Niederlage ist etwas was man wählt, wenn man einen Fehler permanent akzeptiert. Er hält das Leben für schwierig, man wird niedergeschlagen, aber es ist an einem selbst zu entscheiden, ob man liegen bleibt oder wieder aufsteht. Es liegt an einem selbst die Freude und das Vergnügen zu realisieren, welche zum Großteil dann kommt wenn man es trotz aller Schwierigkeiten und Kämpfe geschafft hat. (S.426)

Statt Fehler zu fürchten soll man sie akzeptieren, sie erwarten und bereit sein ihnen zu begegnen. Es ist für ihn ein notwendiger Schritt um erfolgreich zu sein. Nur wenige Dinge im Leben werden ohne Fehler getan.

Er sieht es als Problem, dass wir lernen Fehler im negativen Licht zu sehen. Als Beispiel nennt er einen nicht bestandenen Test in der Schule den man nicht als Lernfortschritt sieht, um seine Ziele zu erreichen. Stattdessen sieht man das Ganze negativ.

Er will damit sagen, dass ein Fehlschlag im echten Leben für gewöhnlich ein notwendiger Meilenstein ist, welcher uns immer näher an den eventuellen Erfolg bringt. Mit einem Fehlschlag lernt man dazu und wächst an der Erfahrung. Laut ihm ist unser Gehirn darauf trainiert auf solche Weise zu arbeiten. Das Gehirn macht über die Zeit geringe Korrekturen bei sich wiederholenden Fehlern, die man erlebt, ohne sich dessen bewusst zu sein. Alles was man laut ihm versuchen muss, ist es weiter zu machen und keine Angst vor Fehlern zu haben. (S.426-427)

Weiterhin sagt er, dass es nicht reicht die Angst vor Fehlern zu verlieren, sondern dass man Fehler suchen soll. Man soll sich selbst in Situationen bringen, wo es naheliegt, dass man fehlschlägt. Er behauptet, dass wir oft stagnieren und aufhören Dinge zu tun, die für uns gefährlich oder herausfordernd sind. Wir suchen uns einen komfortablen Platz in unseren Leben.

Manchmal sollte man gewillt sein in eine unkormfortable Situation zu geraten, welche einem

zwingt daran zu wachsen. Manchmal soll man versuchen diese Situationen herbeizuführen, weil je mehr man fehlschlägt, die anschließenden Erfolge umso größer erscheinen. Man soll das Akzeptieren von Fehlern zum Teil seines Lebens machen und hinnehmen, dass manchmal Fehler unvermeidlich sind. Man kann seiner Meinung nach nicht alles beim ersten mal perfekt machen, Fehler sind für ihn vorprogrammiert. Wenn man dies akzeptiert hat, dann hört man auf sie zu fürchten. (S.427-428)

3 Erfolgsfaktoren für Unternehmen im Hinblick auf die Digitalisierung

3.1 Informationstechnologie

Informationstechnologische Systeme haben eine enorme gesellschaftliche und technische Veränderung ausgelöst. Diese bilden durch das Sammeln und Verarbeiten von Informationen eine Erleichterung vieler Prozesse, was es stark in der Gesellschaft und Wirtschaft expandieren ließ.

3.1.1 Der Ursprung

Erste Entwicklungen begannen mit dem Telefon, welches viele für unzureichend hielten und niemals erwartet hätten, dass es einmal weltweit für vernetzte Kommunikation sorgen wird. Ebenso erging es den ersten Computern. Die großen Rechnerräume, welche benötigt wurden um die ersten Computer in Betrieb zu nehmen, gaben vielen das Gefühl, dass es nur einen äußerst kleinen Markt für solche Geräte gäbe.

3.1.2 Die Gegenwart

In der Wirtschaft werden informationstechnologische Systeme beispielsweise zur Unterstützung von Geschäftsprozessen verwendet um eine reibungslose und optimale Verwaltung von Ressourcen über vernetzte Computer zu gewährleisten. Diese Systeme, auch Enterprise-Resource-Planning-Software genannt, geraten immer häufiger in einen Konflikt mit der stetig wachsenden Komplexität der Anforderungen. Hieraus resultieren Ausfälle der Systeme, schlechte Laufzeiten bei Suchanfragen, unbefriedigende Ergebnisse für den Kunden und erhöhte Supportanfragen von Kunden. Daher ist es erforderlich Korrekturmaßnahmen durchzuführen um die Komplexität dieser Systeme zu verringern, beispielsweise mithilfe von Standardisierungen oder ausgereiften Architekturmodellen.

3.1.3 Die Zukunft

In Zukunft werden diese vernetzten Systeme im Alltag viel präsenter sein. So können in einer Vielzahl an Alltagsgegenständen Sensoren für einen komfortableren Lebensstil, sowie für optimierte Geschäftsprozesse sorgen. Die Vielzahl unterschiedlicher Sensoren sorgt für riesige Mengen an Daten, was eine effizientere Verarbeitung von Daten voraussetzt um nicht an alten Daten hängen zu bleiben, wenn sich die Bedingungen bereits geändert haben und bestimmte Reaktionen erwartet werden. Durch eine Vernetzung von Kunden und Geschäftspartnern (wie beispielsweise Lieferanten) können mithilfe der Analyse von Datenmengen sowie der Vernetzung verschiedenste Geschäftsbereiche, effizienter und flexibler auf Anforderungen bzw. Änderungen von Auftragsgebern reagiert werden. Diese Vernetzungen, Nutzungen von Services und die daher gehende Flexibilität, wird durch das

Cloud Computing voran gebracht. Es vereinfacht und spart Kosten bei der Bereitstellung solcher Dienste. Dabei kann die Infrastruktur frei gewählt werden und entschieden werden, ob die Dienste in der eigenen Umgebung oder in der verteilten (privaten oder öffentlichen) Cloud angeboten werden soll.

3.1.4 Welchen Nutzen hat Informationstechnologie

Die Informationstechnologie ermöglicht einerseits die Entstehung neuer Marktsegmente, andererseits aber auch die Optimierung bestehender.

3.1.4.1 Wirtschaftlichkeit

Bei der Optimierung von Geschäftsprozessen wird erst überprüft, ob eine Optimierung überhaupt wirtschaftlich ist. Hierfür werden verschiedenste Analysen bezüglich der Kosten und des Nutzens durchgeführt. Es wird dabei zwischen zwei Fällen unterschieden, bei welchen eine Wirtschaftlichkeit vorliegt. Einerseits ein identisches Ergebnis zum vorherigen Verfahren bei verringertem Aufwand, andererseits ein verbessertes Ergebnis (beispielsweise die Qualität des Produktes) bei nahezu gleichem Aufwand. Bei der Analyse ist es noch wichtig zu betrachten, dass die zu tätige Investition in Zukunft abgedeckt wird und höhere Einnahmen erzielt werden.

3.1.4.2 Wettbewerbsfähigkeit

Ein weiterer Nutzen der Informationstechnologie ist die Wettbewerbsfähigkeit. Ein Verzicht auf informationstechnologische Systeme würde ein Unternehmen so starke Defizite in Punkto Planung, Zuverlässigkeit, Effizienz und einigen weiteren Punkten einbringen, sodass es nicht mehr konkurrenzfähig wäre. Daher werden solche Systeme unerlässlich für Unternehmen und werden immer weiter entwickelt um bessere Verfahren gegenüber der Konkurrenz zu entwickeln.

3.1.5 Einsatz von Informationstechnologie in schlanken Unternehmen

3.1.5.1 Woran können wir ein schlankes Unternehmen erkennen?

In schlanken Unternehmen wird die kontinuierliche Verbesserung von Geschäftsprozessen und Kundenzufriedenheit angestrebt um einen möglichst hohen Gewinn bei einer möglichst geringen Verschwendungen zu erzielen.

3.1.5.2 Wie setzen schlante Unternehmen Informationstechnologie ein?

Im Fokus eines Unternehmens sollten zu allererst die Prozesse liegen. Sind die Geschäftsprozesse noch nicht ausgereift und könnten noch optimiert werden, sollte noch nicht mit der Arbeit eines Softwaresystems begonnen werden. Erst wenn ein Geschäftsprozess ausgereift und stabil im Unternehmen durchgeführt wird, ist es sinnvoll ein Softwaresystem zur Verbesserung der Durchführung zu entwickeln. Ein weiterer wichtiger Punkt für den Einsatz, ist die Flexibilität. Hierbei sollte darauf geachtet werden, dass Informationen immer zum richtigen Zeitpunkt versendet werden. Werden zu früh Informationen versandt, kann es unnötige Prozesse hervorrufen, die aufgrund von Änderungen nicht nötig waren.

3.1.5.3 Wie arbeiten schlante IT-Organisationen?

Schlante Organisationen verwenden Echtzeit-Prinzipien für ihre Services. Dabei wird schnellstmöglich auf Fehler reagiert, unter ständiger Verbesserung Services von hoher Qualität angeboten und die Antwort- und Verarbeitungszeiten möglichst gering gehalten. Ebenfalls wird darauf geachtet Verschwendungen zu finden wie beispielsweise nicht benötigte Geschäftsprozesse und diese zu eliminieren.

Fließ-Prinzip (One piece flow)	Synchronisations-Prinzip (Customer's cycle)	Nachfrage-Prinzip (Pull principle)	Null-Fehler-Prinzip (Zero faults principle)
Ein direkter Material- und Informationsfluss ermöglicht kurze und stabile Durchlaufzeiten.	Am Kundentakt ausgerichtete und aufeinander abgestimmte Prozesse erzielen Effizienz.	Oberproduktion lässt sich vermeiden, wenn nur das produziert wird, was tatsächlich gebraucht wird.	Stabile und fehlerfreie Prozesse erhöhen die Planbarkeit und reduzieren die Verschwendungen.

Anhand der Abbildung können Verfahren zu kontinuierlichen Entwicklung, besseren Wertschöpfung und erhöhter Kundenzufriedenheit eingesehen werden.

3.1.5.4 Wie erfolgen Problemlösungen in schlanken Organisationen?

Zum problemlösen wurde ein Problemlösungsprozess entwickelt, welche bei dessen Durchführung helfen soll, Probleme zu diagnostizieren und zu umgehen. Dazu wird erst geplant, was überhaupt das Problem ist und in wie fern es der optimalen Lösung abweicht. Ebenfalls wird überprüft, was eine machbare und geeignete Gegenmaßnahme sein könnte. Im nächsten Schritt wird diese Gegenmaßnahme ausgeführt und die darauf folgenden

Ergebnisse überprüft. War die Gegenmaßnahme erfolgreich wird die Korrektur standardisiert. War es nicht erfolgreich muss der komplette Prozess neu durchlaufen werden.

3.1.5.5 Wie lassen sich schlank Managementmethoden in Entwicklungsprojekten nutzen?

Es gibt verschiedene Managementmethoden, welche unterschiedliche Ziele verfolgen. Ist es vorgesehen, dass Fehler schnell erkannt werden sollen und eine hohe Präsenz des Kunden in den Planungsschritten vorhanden sein soll, bietet sich Scrum an. Hierbei werden Anforderungen ausgearbeitet, welche in Sprints (2-4 Wochen) ausgearbeitet werden. Die entstandenen Prototypen werden mit dem Kunden analysiert, welcher Verbesserungsvorschläge und Anmerkungen äußern kann. Hierdurch werden außerdem frühzeitig Fehler erkannt. Als Alternative gibt es die klassische Projektmanagementmethode, welche eine bessere Abhilfe in Punkt Zeit- und Budgetmanagement bietet, jedoch eine geringere Kundennähe und spätere Fehlererkennung beinhaltet. Eine weitere Möglichkeit ist es die Komplexität in einfache kleine Arbeitspakete einzuteilen und regelmäßig Meetings mit dem Kunden zu vereinbaren um die Zufriedenheit mit dem Produkt zu gewährleisten. Die Arbeitspakete sollten hierbei eine Dauer von einem Tag haben und werden in täglichen Meetings besprochen, analysiert und Probleme behoben.

3.1.5.6 Welche Erfolgsfaktoren gibt es für schlanke IT-Organisationen?

Um Unternehmen langfristig ihren Erfolg zu gewährleisten, sollten diese nicht auf bestehenden Verfahren hängen bleiben, sondern eine stetige Weiterentwicklung und Standardisierung dieser Fortschritte durchführen.

3.1.5.7 Wie lässt sich der Reifegrad schlanker Organisationen ermitteln?

Der Reifegrad misst sich an verschiedenen Eigenschaften des Unternehmens. Die wichtigsten sind die fortlaufenden Entwicklungen. Das bedeutet, dass Unternehmen immer wieder Problemanalysen durchführen, Prozesse dabei optimieren und überflüssige Prozesse minimieren bzw. eliminieren sollten. Ebenfalls spielt hierbei die Kundennähe stark mit ein und die Präsenz der Führungskräfte.

3.1.6 Was benötigt eine erfolgreiche IT-Organisation?

3.1.6.1 Organisationsstruktur

Eine IT-Organisation besteht aus dem Governance-Bereich, in dem Verwaltungs- und verschiedenste Managementtätigkeiten (z.B. Standardisierungs- und Qualitätsmanagement) ausgeführt werden, der Anwendungsentwicklung und –betreuung, welche sich sowohl mit der Entwicklung als auch mit dem Support und Projektmanagement befasst, sowie dem IT-Betrieb, welcher verschiedene administrative Dienste leistet. Ein weiterer entscheidender Punkt in der Organisationsstruktur ist die Entscheidungsfreiheit und Verteilung von Verantwortlichkeiten. Werden den Arbeitern die Aufgaben vorgesetzt ohne dass diese Verbesserungen oder ähnliches äußern können, führt dies zu einer Demotivation. Die Kreativität wird somit unterdrückt und der Arbeiter sieht sich selbst nicht als Verantwortlichen für die Aufgabe. Das löst ebenfalls eine stärkere Fehleranfälligkeit aus, da der Arbeiter nicht mehr an Verbesserungsmöglichkeiten denkt. Ein freies Arbeitsumfeld für den Arbeiter weckt die Begierde Erfolg zu haben und verbessert die Leistungen. Jedoch sollten auch nicht zu früh zu viele Freiheiten gegeben werden. Gerade in dem anfänglichen Reifungsprozess sollte eine ausreichende Kontrolle vorhanden sein. Diese sollte sich jedoch mit der Zeit immer mehr verringern und den Teams die Entscheidungsfreiheit lassen.

3.1.6.2 IT-Mitarbeiter

Die Funktion eines IT-Mitarbeiters wird über die Stellenbeschreibung festgelegt. Hierbei wird nicht nur Wert auf Fachwissen gelegt. Viel entscheidender sind Soft Skills, wie der Umgang im Team und mit Kunden, sowie das Herstellen von fachübergreifenden Verknüpfungen. Ebenfalls werden IT-Mitarbeiter häufig in den Kontakt mit neuen Technologien kommen. Daher benötigen diese fundamentale Kenntnisse von Technologien und die Fähigkeit sich in neue Sachverhalte einzuarbeiten. Da Technologien stetig weiterentwickelt werden, ist es ebenfalls erforderlich, dass IT-Mitarbeiter permanent lernen und sich weiterentwickeln. Um diese Fähigkeiten in einem IT-Mitarbeiter zu erkennen gibt es Assessment Center. Hierbei werden beispielsweise Stresssituationen getestet. Assessments werden nicht nur bei der Bewerbung genutzt, um die Qualifikation von Bewerbern zu testen. Es kann auch Verwendet werden um Mitarbeitern intern die richtige Rolle zuzuordnen. Um eine individuelle Bewertung aufzustellen, wird der Mitarbeiter nach bestimmten Kriterien analysiert. Angefangen wird mit einem Anforderungskatalog, wo niedergeschrieben wird, welche Kompetenzen und Anforderungen überhaupt gesucht werden. Hierbei können beispielsweise bestimmte Führungsqualifikationen niedergeschrieben werden. Nach dem Assessment werden die Kompetenzen im Kompetenzmodell individuell bewertet. Danach kann das Ergebnis in einem Potential- und Leistungsdiagramm eingestuft und verglichen werden. Hieraus können nun neue Entschlüsse gezogen werden, wie beispielsweise Fortbildungsmaßnahmen.

3.1.6.3 IT-Buisness-Alignment

Die korporative Zusammenarbeit zwischen dem Geschäftsbereich und der IT-Organisation, welche sich gegenseitig gleichberechtigt behandeln prägt den Begriff IT-Buisness-Alignment. Arbeiten beide Parteien zusammen, wird die Kreativität beider Seiten genutzt, eine höhere Effizienz erzielt, Fehler vermieden und eine gesunde Arbeitsatmosphäre geschaffen.

3.1.6.4 IT-Strategie

Eine IT-Strategie ist ein mittel- bis langfristiger Plan für die Entwicklung der IT in einem Unternehmen. Die IT-Strategie sollte mindestens einen Zeitraum von 2 Jahren umfassen. Für die Entwicklung einer IT-Strategie müssen die Bereiche der geschäftlichen Anforderungen an die IT, sowie der aktuelle Zustand der Unternehmens IT analysiert werden.

Eine fertige IT-Strategie sollte folgende Punkte enthalten:

- Grundsätze
- IT-Architektur mit IT-Applikationen, IT-Infrastruktur, Cloud-Computing und Mobilität
- IT-Services
- IT-Sourcing
- IT-Organisation
- IT-Projekt-Portfolio

3.1.6.5 IT-Governance

Die IT-Governance umfasst die Grundsätze, Verantwortlichkeiten und Prozesse zur Steuerung des Umgangs mit der IT im Unternehmen, inkl. des dazugehörigen Risiko- und Compliance-Managements.

Die Aufgaben des IT-Governance werden vom CIO, seinem IT-Führungsteam, den IT-Boards und dem CIO-Office mit Partnermanager, Sicherheitsmanager, IT-Controller etc. wahrgenommen. Zu den Aufgaben gehören u.a. die Entwicklung der IT-Strategie sowie die Steuerung und Überwachung ihrer Umsetzung, aber auch das IT-Risiko-, IT-Compliance- und IT-Sicherheitsmanagement.

3.1.6.6 IT-Architektur

Die IT-Architektur bildet eine Grundstruktur für die IT-Infrastruktur. Sie ist auf "das Design, die Auswahl, die Entwicklung, die Implementierung, die Wartung und das Management der IT-Infrastruktur" anzuwenden. Mit ihr erfolgt eine "einheitliche strategische Ausrichtung" der IT. Die gewählte Architektur muss in der Lage sein, zukünftige Anforderungen zu erfüllen. Währenddessen muss aber eine hohe Verfügbarkeit und Leistungsfähigkeit der Systeme

gewährleistet werden. Ähnlich wie bei der Planung einer Stadt gibt es bei der Entwicklung einer IT-Architektur einen "Bebauungsplan". Dieser dokumentiert den Ist-Zustand, zeigt aber auch die notwendigen Schritte hin zur neuen Soll-Architektur auf. Bei der Entwicklung der Soll-Architektur ist es wichtig, dass sie zum allgemeinen Geschäftsprozess des Unternehmens passt. Auch müssen die Anforderungen an die neue Architektur definiert werden. Zudem dürfen aktuelle Trends nicht ignoriert werden. Die zukünftige IT Landschaft soll: zentral, einfach, integriert und flexible sein, ohne unnötige Risiken zu beinhalten. So sollte innerhalb einer Unternehmensgruppe eine einheitliche Software-Plattform eingesetzt werden. Aber auch die genutzte Hardware sollte einer Standardkonfiguration entsprechen. Einige Aktuelle Trends sind: Hybrid Infrastruktur: Unternehmenskritische Dienste lokal hosten, andere Dienste auslagern SaaS: Software kann kostengünstig aus der Cloud bezogen werden. Industrie 4.0: Hoch vernetzte Fertigungsprozesse Big Data: Einsatz neuer Analyse Methoden für große Datenmengen Arbeitsplatz der Zukunft: Hohe Vernetzung ermöglicht trotz großer Mobilität einfache Kommunikation Design Thinking: Multidisziplinär arbeitende Teams entwickeln Produkte die sich nah am Kunden befinden

3.1.6.7 IT Standards:

Informationstechnologien sind Produktionsmittel und somit ist ihre Qualität und Verfügbarkeit für ein Unternehmen wichtig. Hierzu müssen Standards definiert und eingehalten werden. Sowohl das BSI als auch die ITIL definieren für ihren Bereich „best practises“.

3.1.7 Verschiedene Standards:

Corporate Security Policy: Jeder Mitarbeiter muss alle Regeln verantwortungsvoll einhalten (z.B. Passwörter und Virenschutz) IT-Architektur: Standardisiert die IT Architektur und führt damit auch zu einer strategischen Ausrichtung der IT IT-Prozesse: Definiert für ein Unternehmen Soll Abläufe IT-Sicherheit: Risikoanalyse mit CIA Aspekte. Um den Ablauf zu vereinfachen wurde ein IT-Sicherheits-Basis-Check erstellt. Dieser gliedert sich in verschiedene Kategorien und je einer gestellten Frage. Ausfalltoleranz: Wie lange kann das Unternehmen ohne IT auskommen? Sicherheitsrisiken: Finanzielles Risiko bei IT Ausfall bekannt? Notfallablauf: Wer wird informiert? Vorkehrungen: Virenschutz, Firewall, Sicherheits-Patches aktuell?

3.1.7.1 IT Budget

Das IT Budget bildet eine Entscheidungsgrundlage für Unternehmen. Die Angaben des „IT Demand Managers“ fließen in die IT Budgetplanung ein. Sie wird vom IT Controller in Zusammenarbeit mit den IT Führungskräften erstellt. Auftretende Änderungen sind aufgrund

des Vorjahresberichts gut festzustellen. Das Budget ermöglicht zudem Aussagen zur Realisierbarkeit von IT Projekten zu treffen. So darf der geschätzte Projekt Aufwand nicht die verfügbaren Mittel überschreiten.

Um einen transparenten Soll-Ist-Vergleich des Budgets zu schaffen wird vom Controlling ein monatliches „Kosten reporting“ erstellt. Neben dieser Aufgabe hat das Controlling noch weitere Beratende und Unterstützende Aufgaben. Auch muss es verschiedene Be- und Verrechnungen erstellen.

3.1.7.2 IT-Partner-Management

Das IT-Partner-Management überwacht, steuert und optimiert Ausgelagerte (outgesourcete) Dienstleistungen. Um zu erkennen, ob eine Partnerschaft nicht mehr profitable, ist ein regelmäßiger Informationsaustausch zwischen den Partnern nötig.

Das Outsourcing ermöglicht es dem Unternehmen sich auf die Kernaufgaben zu konzentrieren. Gründe für das Outsourcing können kosten Reduktion, Verteilung von Kompetenzen oder die fehlende technische Innovation des eignen Unternehmens sein. Um einen Outsourcing Dienstleister zu wählen sind verschiedene Kriterien anzuwenden. Sie gliedern sich von Preis über Referenzen hin zur finanziellen Stabilität des Dienstleisters.

3.1.7.3 Business-Process-Management

„Business-Process-Management (BPM) verbessert die Unternehmensleistung durch ein konsequentes Geschäftsprozessmanagement und die fortlaufende Optimierung der Geschäftsprozesse. Dies umfasst die Analyse, das Design, die Modellierung, die Implementierung, die Überwachung und die Optimierung von Geschäftsprozessen. Optimale Prozesse orientieren sich am Kundenbedarf und haben einen hohen Wertschöpfungsanteil. Sie sind effektiv und effizient.“ Da alle Aktivitäten optimal aufeinander abgestimmt sein müssen, wächst die Bedeutung von BPM weiter. Die definierten Prozesse können aber nicht dauerhaft perfekt sein, deshalb müssen sie ständig angepasst werden. Dieses resultiert aus dem sich ständig wandelnden des Marktes, der Technologien und des Wettbewerbs. Die Aufgaben eines Prozessberaters im BPM Team umfassen: Die Analyse und den Entwurf von Dokumenten, die Erfassung der Anforderungen an die Prozesse sowie Beratung und Moderation von Prozess Design Workshops.

3.1.7.4 Service-Level-Management

Das Service-Level-Management stimmt mit den internen Kunden ab, welche Dienstleistungen in welcher Qualität (Service-Level-Agreement) zu erbringen sind. Die Service-Level-Agreements definieren die Qualität und Verfügbarkeit der von der IT-

Organisation angebotenen IT-Dienstleistungen. Außerdem überwacht es die Einhaltung der Vereinbarungen und erstellt Reports. Zu den häufig benötigten Services gehören IT-Service-Desk, Telekommunikation und Computerarbeitsplatz (File und Print Service, Mail, LAN).

3.1.7.5 IT-Demand-Management

„Das IT-Demand-Management ist die zentrale Stelle für die strukturierte Aufnahme, Bündelung, Bewertung und Umsetzungskoordinierung von IT-Anforderungen“. Es dient dazu den "IT-Unterstützungsbedarf der internen Kunden zu identifizieren". Zudem sorgt es für den zielgerichteten Einsatz von IT Ressourcen. Hierbei werden wichtige Aufgaben höher Priorisiert als andere. Insgesamt ist das Demand Management eine elementare Komponente erfolgreicher IT Organisationen.

3.1.7.6 IT-Leistungsverrechnung

IT Organisationen erbringen die vereinbarten Leistungen als Shared-Service-Center. Alle möglichen Leistungen werden in einem IT Leistungskatalog zusammengefasst. Er wird veröffentlicht. In diesem Katalog wird auf die Leistungsart, die SLA Zuordnung, die Verrechnungseinheit und die Preise eingegangen. Mit Hilfe dieses Katalogs erkennen die Kunden der Wert der erhaltenen Leistung einfacher und die Kostentransparenz steigt. Für die Anbieter hat diese Methode den Vorteil, dass die Kunden nicht mehr benötigte Leistungen früher melden, da sie direkt Kosten einsparen können. Der Anbieter kann so knappe Ressourcen besser verwalten. Die zugrundeliegenden Preise werden jährlich festgelegt und bleiben über das Jahr konstant.

3.1.7.7 IT-Risikomanagement

Das IT-Risikomanagement dient zur Sicherstellung der Kontinuität des Geschäftsbetriebs. Es beinhaltet die „Erfassung, Bewertung, Behandlung und Überwachung von Risiken“. Das BSI unterscheidet zwischen 6 Kategorien („Elementare Gefährdung, höhere Gewalt, organisatorische Mängel, menschliche Fehlhandlung, technisches Versagen, vorsätzliche Handlung“). Mögliche resultierende Risiken sind der Verlust von Daten, der Ausfall von Systemen oder Verstöße gegen rechtliche Vorschriften. Um die Risiken und dessen Folgen einzuschätzen zu können wird eine Risikoanalyse durchgeführt. Hier werden Risiken mit ihren erwarteten Eintrittswahrscheinlichkeiten und dessen finanziellen Folgen aufgeführt. Außerdem wird für jedes unternehmenskritische System eine Gefährdungsanalyse durchgeführt. Bei dieser wird erfasst, welche Vorkehrungen bereits getroffen wurden und welche noch getroffen werden müssen (Redundanz, Wartungsvertrag). Hieraus ergibt sich ein Realisierungsplan. In diesem sind Verantwortliche, sowie Kosten und Termine aufgeführt.

3.1.7.8 IT-Asset- und Lizenzmanagement

Um nicht gegen die vertraglich festgelegten Nutzungsbedingungen zu verstößen, werden immer aktuelle Informationen über die eingesetzten Produkte benötigt. Andernfalls drohen Strafen. Ein IT-Asset ist jedes IT-System und Software-Lizenz die dem Unternehmen gehört. Die Informationen über diese kommen aus verschiedenen Quellen (Buchhaltung System bis Active Directory) und können in einem IT-Asset-Managementsystem verwaltet werden. Der Einsatz eines solchen bringt verschiedene Vorteile mit sich: Lizenzverträge können optimiert werden, Lizenzrechtsverletzungen sind vermeidbar, Vermeidung von nicht genutzten IT Assets.

3.1.7.9 BI-Management (Business Intelligence)

BI-Management beschreibt die systematische Erfassung, Auswertung und Darstellung von geschäftlichen Daten. Diese Daten dienen den Führungskräften bessere Entscheidungen zu treffen. Um ein solches zentrales Berichtssystem aufzubauen wird ein Business-Intelligence-Manager benötigt. Er arbeitet mit dem Controlling und der IT Organisation zusammen und Standardisiert unter anderem Kennzahlen und Berichte.

3.1.7.10 Master Data Management

Um die Stammdaten eines Unternehmens (Produkte, Lieferanten, Kunden, Mitarbeiter) zu verwalten arbeitet der Stammdatenmanager übergreifend über eine Unternehmensgruppe.

3.1.7.11 KPIs

Kennzahlen oder auch „Key Performance Indicators“ (KPIs) machen es möglich komplexe Sachverhalte besser zu verfolgen zu können. Außerdem dienen sie dazu, den Erfüllungsgrad wichtiger Ziele zu messen. Daher sind sie unverzichtbar um wichtige Entscheidungen zu treffen. Mit Hilfe dieser Zahlen ist es einfacher Erfolg oder Misserfolg einer Entscheidung festzustellen. So können die gemessenen Zahlen vor der Entscheidung mit denen, die nach der Entscheidung erstellt worden verglichen werden.

3.1.8 Wie lässt sich eine IT-Organisation am besten optimieren?

3.1.8.1 Due Diligence

Als Due Diligence wird die Prüfung einer Organisation nach den Kategorien Stärken, Schwächen, Chancen und Risiken bezeichnet. Gründe für eine solche Prüfung könnten u.a. die Entwicklung einer IT-Strategie seien oder die Übernahme einer neuen Organisation.

3.1.8.2 Business Plan

Der Business Plan beschreibt den Ablauf, die Steuerung, die Vermarktung und die Finanzierung des Geschäfts.

Bestandteile eines Business Plans sind:

- Management Summary
- Unternehmen
- Portfolio
- Markt und Wettbewerb
- Marketing und Vertrieb
- Management und Organisation
- Drei-Jahres-Planung
- Chancen und Risiken Finanzbedarf
- Anlagen

3.1.8.3 Business Case

Ein Business Case dient dazu, die Wirtschaftlichkeit zu ermitteln und beinhaltet alle einmaligen sowie laufenden Kosten, den finanziellen Nutzen, sowie en daraus resultierenden Nettonutzen. Außerdem den ermittelten Return on Investment (ROI) und den Amortisationszeitraum.

3.1.8.4 Programm- und Projektmanagement

Zum **Programmmanagement** gehört die übergreifende Leitung und Steuerung inhaltlich zusammengehöriger Projekte, welche ein gemeinsames Ziel haben und dessen Termine und Inhalte meist voneinander abhängen.

Das **Projektmanagement** umfasst das Initiieren, Planen, Steuern, Kontrollieren und Abschließen von Projekten, mit dem Ziel eines effizienten Ressourceneinsatzes.

3.1.8.5 Change Management

Änderungen an unternehmenskritischen Komponenten der IT-Landschaft, welche den bestehenden Betrieb von Business-Services gefährden können, werden mit Hilfe eines Change Managements durchgeführt. In einem Change Management werden

Änderungsanforderungen (Change Requests) nach ihren Risiken und ihrer Notwendigkeit analysiert. Dies wird gemacht, um vorbeugende Maßnahmen für die Risiken sowie Fall-Back-Maßnahmen festzulegen und außerdem kann an dieser analysiere über die Umsetzung entschieden werden.

3.1.9 CIO

3.1.9.1 Wie wird man CIO?

Um CIO zu werden hat man zwei Möglichkeiten, zum einen kann man diesen Posten über die klassische Laufbahn in einer IT-Organisation erreichen oder als Quereinsteiger mit einer General-Management-Laufbahn. Doch unabhängig welchen Weg man wählt, es gelten dieselben Anforderungen. Diese sind Engagement, ein hohes Maß an Eigeninitiative, überdurchschnittliche Leistungsbereitschaft, Lernbereitschaft und Offenheit zeigen.

3.1.9.2 Aufgaben

Die Aufgaben eines CIOs sind vielfältig, zu seinen Hauptaufgaben gehören, einen stabilen Betrieb der IT-Landschaft zu gewährleisten und Services sowie Innovationen in der vereinbarten Zeit, zum vereinbarten Preis und in der vereinbarten Qualität zu liefern.

3.1.9.3 Führungsinstrumente



3.1.9.4 Erfolgsfaktoren

Um als CIO Erfolg zu haben sind einige Faktoren sehr wichtig. Zum einen sollte der CIO eine sehr gute Kommunikationsfähigkeit haben, komplexe Sachverhalte verständlich erläutern, erfolgreich verhandeln und gut präsentieren können.

Doch der CIO alleine garantiert keinen Erfolg für die IT-Organisation, denn dazu gehört ebenso ein leistungsstarkes Team, welches selbstständig und eigenverantwortlich Arbeit und den wichtigen Situation vom CIO unterstützt und gefördert wird.

3.1.9.5 Networking mit anderen CIOs

Das Netzwerken mit anderen Geschäftspartner, Kollegen oder Bekannten ist wichtig, denn durch diese Beziehungen ist es möglich, auf das Wissen und die Erfahrungen von anderen zurückzugreifen. Bekannte Networking Plattformen sind z.B. LinkedIn.com oder Xing.de welches im deutschsprachigen Raum das bekannteste ist.

3.1.9.6 Work-Life-Balance

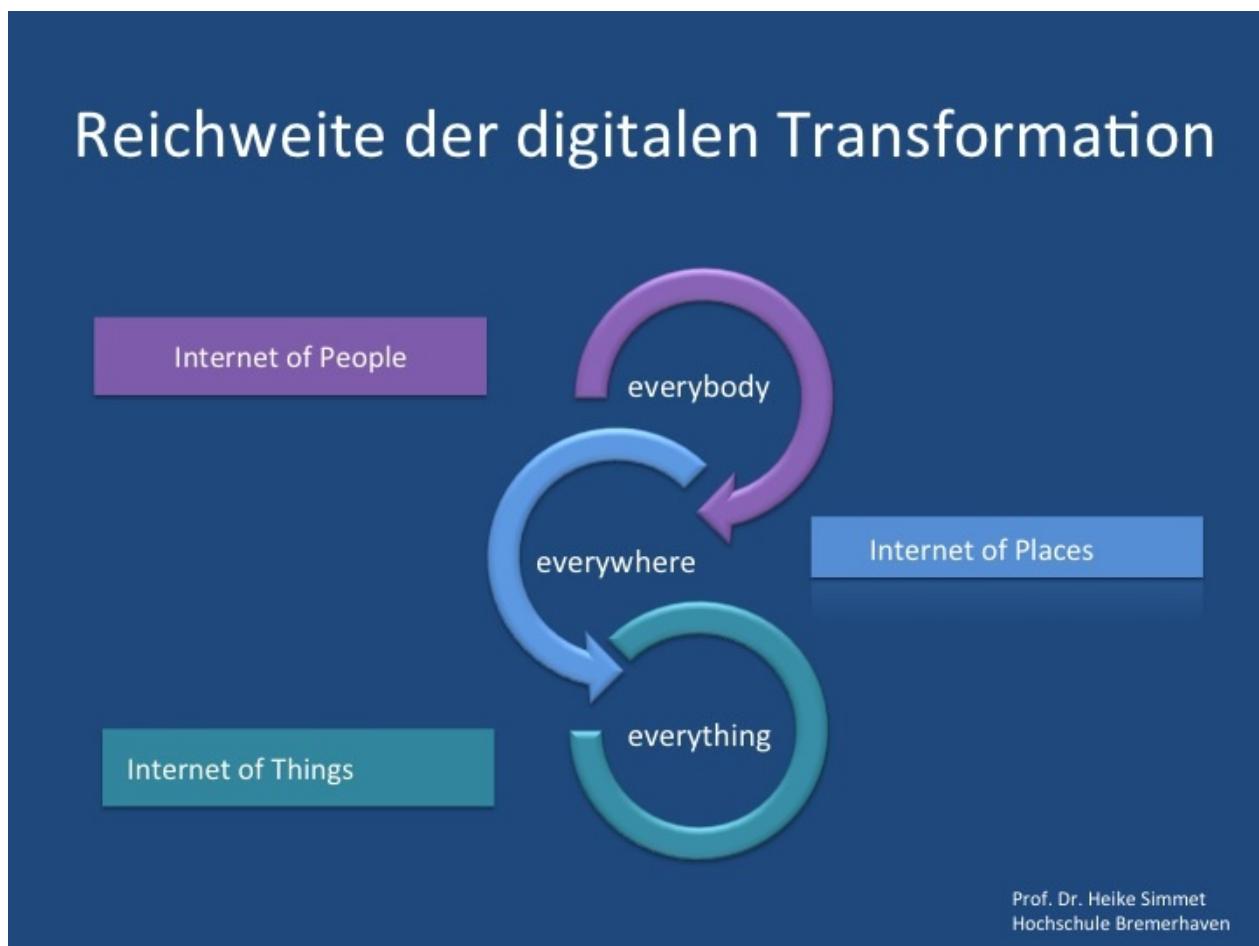
In Zeiten des mobilen Arbeitens und der dauerhaften Erreichbarkeit, ist die Balance zwischen Berufs- und Privatleben besonders wichtig um sowohl im Job sowie auch Privat Erfolg zu haben.

Um eine gute Balance zu erreichen sollte man in seiner Freizeit mal das Diensthandy ausschalten und sich die Zeit mit Sport vertreiben oder zusammen mit Freunden oder der Familie verbringen.

3.2 Connectedness

Die wichtigste Ursache für diesen Wandel ist dabei das Internet.

Es durchdringt unser gesamtes Leben und vernetzt Personen, Dinge und Orte. Ein Leben ohne Internet und diese Vernetzung ist heutzutage kaum mehr vorstellbar.



"Reichweite der digitalen Transformation." (Quelle:<http://hsimmet.com/2013/12/15/wearable-devices-neue-gamechanger-in-der-digitalen-tansformation/>)

Gerade die Sozialen Netzwerke wie z.B. Facebook und LinkedIn erfreuen sich einer großen Beliebtheit und bieten Unternehmen viele verschiedene neue Möglichkeiten, aber auch Gefahren.

Mithilfe von LinkedIn können beispielsweise neue Mitarbeiter fürs Unternehmen auf völlig neue Art gewonnen werden. Statt den ehemals klassischen Weg über Stellenausschreibungen zu gehen, können passende Kandidaten aktiv gesucht werden und somit Stellen schneller und besser besetzt werden.

Durch Facebook hingegen ändert sich die Marketingstrategie vieler Firmen. Statt in teure Marketingaktionen zu investieren, gewinnt die Mund-zu-Mund-Propaganda einen neuen Stellenwert. Durch die Kommunikation in den sozialen Netzwerken kann direkt auf den Kunden eingegangen werden, beispielsweise durch bereitstellen von Informationen, dem

beantworten von Fragen oder dem eingehen auf Beschwerden. Die Kundenbindung wird dadurch gestärkt und Kunden fangen an sich mit dem Unternehmen stärker zu identifizieren, was dazu führt, dass sie selbst das Unternehmen gegenüber anderen Mitgliedern empfehlen.

Das wichtigste Ziel für ein Unternehmen im Bezug auf Soziale Netzwerke ist dabei die Reichweite zu steigern, welche durch Kennzahlen wie Klicks, Besucherzahlen oder Followern etc. angegeben ist. Um diese Reichweite zu steigern gibt es die Taktiken des "Growth Hackings", also speziellen Taktiken ("Hacks") die darauf abzielen diese Kennzahlen zu erhöhen. Mittlerweile gibt es sogar den Job des "Growth Hackers", dessen Aufgabe es ist mit kleinen Tricks die Reichweite seines Unternehmens zu steigern.

Ein weiterer Wandel existiert bei den Business-Modellen. Der Wertschöpfungsprozess von digitalen Gütern, Services und Netzwerken wird durch folgende drei Faktoren bestimmt:

1. Niedrige Grenzkosten

Virtuelle Güter sind meist ohne weitere Kosten reproduzierbar. Zwar ist die Entwicklung von beispielsweise Online-Plattformen teuer, die Kosten einen neuen Nutzer zu integrieren tendieren jedoch gegen Null. Zudem übernehmen Nutzer auch selbst Aufgaben, wie beispielsweise das Melden von illegalen Inhalten oder als Moderator. Somit fallen auch in diesen Bereichen keine Kosten an. Ein weiterer Schritt ist, dass Nutzer sogar selbst die Inhalte liefern, wie beispielsweise bei Youtube. Die Kosten für das Unternehmen belaufen sich somit nur noch auf Entwicklung und Betrieb der Plattform.

2. Netzwerkeffekte

Der Wert eines Netzwerks orientiert sich meist an der Anzahl seiner Nutzer. Je mehr Nutzer ein Netzwerk hat, desto attraktiver ist es für eventuelle neue Interessenten. Mit jedem neuen Mitglied steigt somit der Wert des Netzwerks. Neben der Anzahl ist jedoch auch die Qualität der Teilnehmer eines Netzwerks relevant. Je mehr ein spezieller Teilnehmer des Netzwerks interessant für andere Nutzer ist, desto höher steigt auch hier der Wert des Netzwerks. (Beispiel: Größere/Bekanntere Youtuber)

3. "Long-Tail"-Effekte

Als Long-Tail Effekt wird bezeichnet, dass Anbieter im Internet ihren Gewinn nicht durch ein bestimmtes Kernprodukt erwirtschaften, sondern durch eine Vielzahl von verschiedenen Nischenprodukten.

3.3 Collaboration

Durch Zusammenarbeit lassen sich Ziele erreichen, die von einem einzelnen nur äußerst schwer oder gar nicht erreicht werden können. Durch die Digitalisierung sind verschiedene neue Möglichkeiten der Zusammenarbeit entstanden, so kann beispielsweise gleichzeitig von mehreren weit entfernten Orten aus an einem Projekt gearbeitet werden. Es sind eine Vielzahl von Plattformen und Tools entstanden um diese Kollaboration umzusetzen oder zu nutzen, wie beispielsweise Online-Projektmanagement-Plattformen, von Nutzern gepflegte Wikis, Instant-Messaging-Systeme und Services wie Dropbox und Google-Docs. Durch diese Möglichkeiten der (Interdisziplinären-)Kollaborationen eröffnen sich neue Innovationschancen.

Damit eine soziale Zusammenarbeit im Kreativbereich auch zu einem Erfolg führt müssen folgende Aspekte enthalten sein:

1. Dialogisch

Die Fähigkeit zuzuhören, ohne dass das Gegenüber etwas sagt; zu erspüren, was die Intention des anderen ist, wenn Sprache nicht mehr effizient ist.

2. Konjunktivische Rede

Diese Form der Kommunikation lässt Raum für Mehrdeutlichkeiten und Interpretationen. Dadurch entsteht Raum für Geselligkeit und alle Meinungen/Standpunkte können miteinbezogen werden.

3. Informeller Rahmen

Improvisation statt Verfolgen eines von vornherein vorgegebenen Ziels, um den Prozess der Kreativität nicht einzuschränken.

4. Empathie statt Sympathie

Sympathie ist das Identifizieren mit uns ähnlichen Individuen, Empathie das Verstehen anderer Personen. Sympathie kann bei der Zusammenarbeit hinderlich wirken, Empathie, also der Versuch Kollegen die anders sind als man selbst, zu verstehen, ist jedoch Horizont erweiternd.

Des Weiteren existieren Regeln, die dazu dienen sollen geteilte Ressourcen zu managen:

1. Grenzen zwischen den Nutzern und Ressourcengrenzen

Es existieren klare, lokal akzeptierte Grenzen zwischen 'legitimen Nutzern' und 'Nichtnutzungsberechtigten', sowie klare Grenzen zwischen einem spezifischen Gemeinressourcensystem und einem größeren sozio-ökologischen System.

2. Übereinstimmung mit lokalen Gegebenheiten (Kohärenz)

Die Regeln für eine Aneignung und Reproduktion einer Ressource entsprechen den örtlichen Gegebenheiten, sie überfordert die Menschen nicht und sind aufeinander

abgestimmt, das heißt sie müssen aufeinander bezogen sein. Die Verteilung der Kosten erfolgt proportional zur Verteilung des Nutzens.

3. Gemeinschaftliche Entscheidungsfindung

Teilnehmer haben Mitspracherecht an den Entscheidungen zur Bestimmung und Änderung der Nutzungsregeln eines Systems.

4. Monitoring der Nutzer und Ressourcen

Die Überwachung der Ressourcen wird entweder von den Nutzern selbst übernommen oder muss mit den Nutzern geteilt werden.

5. Abgestufte Sanktionen

Die Bestrafung von Regelverletzungen beginnt auf niedrigem Niveau und verschärft sich, wenn der Nutzer die Regel mehrmals verletzt hat. Alle Sanktionen sind dabei glaubhaft und nachvollziehbar.

6. Konfliktlösungsmechanismen

Konfliktlösungsmechanismen müssen gleichzeitig schnell, direkt und günstig sein. Es gibt lokale Räume die für die Lösung von Konflikten genutzt werden und zwischen Nutzern sowie Nutzern und Institutionen.

7. Anerkennung

Es ist ein Mindestmaß an Anerkennung des Rechts der Nutzer erforderlich, ihre eigenen Regeln bestimmen zu können.

8. Eingebettete Institutionen

Wenn beispielsweise eine Gemeinressource eng mit einem Ressourcensystem verbunden ist, sind Governance-Strukturen auf mehreren Ebenen miteinander verknüpft.

Diese kreative Zusammenarbeit bietet einem Unternehmen diverse Vorteile, beispielsweise sind Produktionsprognosen, die kollaborativ im Unternehmen erstellt wurden meist genauer, als wenn sie nur von einer verantwortlichen Person erstellt wurden.

Auch eine Zusammenarbeit mit dem Kunden und seine Einbeziehung in die Unternehmensprozesse tritt immer häufiger auf. Durch die wie bereits beschriebene Möglichkeit von Kunden beispielsweise durch soziale Netzwerke viel stärker als früher einen Einfluss auf Unternehmen zu haben, ist es notwendig auf die Wünsche des Kunden einzugehen und ihn bei der Produktentwicklung zu beteiligen. Dies hat den entscheidenden Vorteil, dass das Vertrauensverhältnis zwischen Produzent und Konsument gefördert wird.

Um diese strukturelle Kollaboration zwischen Unternehmen und Kunde zu einem Erfolg zu führen, existiert ein fünfstufiges Konzept. Der Kunde wird dabei nicht nur in einen Teil des Entstehungsprozesses miteingebunden, sondern in alle Entscheidungsprozesse der Firma, vom Brainstorming über mögliche neue Produkte über Co-Creation der Werbung, bis hin zum Gestalten des Preises.

1. Die Unternehmenskultur

Es ist darauf zu achten das die geplante Kollaboration zu der bestehenden

Unternehmenskultur passt. Das Unternehmen soll also nicht Kollaborationen angepasst werden, sondern genau andersherum. Veränderungen von firmeninternen Strukturen erfolgen langsam. Wurde eine neue Idee beispielsweise bisher von einem Mitarbeiter entwickelt, sollte dieser nicht ersetzt werden, sondern die Kollaboration sollte darauf abzielen diesen Mitarbeiter zu unterstützen. Nach mehreren erfolgreichen Kollaborationen wird sich die Kultur des Unternehmens nach und nach ändern und tiefergreifende Kollaborationen mit dem Kunden können angegangen werden.

2. Die richtigen Menschen

Nicht jeder beliebige Kunde ist für einen Kollaborationsprozess geeignet. Es sollten Kunden gefunden werden, die eine hohe Eigenmotivation haben und bestenfalls ein bestimmtes Fachwissen vorweisen und Erfahrung mit den relevanten Produkten/Dienstleistungen haben, da diese letztendlich auch die Kunden sind, die diese Produkte/Dienstleistungen kaufen/nutzen.

3. Die aktive Einbeziehung der Führungsebene

Top-Manager sollten Kollaborationsprozesse nicht nur unterstützen, sondern auch aktiv miteinbezogen werden, da ansonsten die Gefahr besteht, dass Führungskräfte Entscheidungen treffen ohne auf die Ergebnisse der Kollaboration zu achten.

4. Grenze zwischen interner und externer Kommunikation

Diese Grenze gilt es zu durchbrechen. Wenn die Kollaboration nur innerhalb der Firma stattfindet, wird niemals die maximale Reichweite erzielt werden.

5. Das Ausmaß abschätzen

Es ist notwendig, das Ausmaß der erforderlichen Arbeitsaufwände im Vorfeld abzuschätzen, sowohl firmenintern, als auch extern, um das finanzielle Risiko zu verringern. Dafür ist es erforderlich die Ziele zu definieren, Beteiligten mit einzubinden und Erwartungen zu definieren. Im Zuge der Kollaboration mit der Community ist auch die Ernennung eines Community-Managers ratsam, der für die Kommunikation zwischen Unternehmen und Community zuständig ist.

4 Digitalisierung

4.1 Beginn und Industrie 4.0

Die Ursprünge der heutigen digitalen Revolution liegen in den 1990er Jahren. Das Internet machte es möglich, Prozesse, wie Logistik und Kundenpflege, durch die IT zu unterstützen. Dies mündete in Entwicklungen, wie beispielsweise dem E-Business. Heutzutage werden aber grundlegend neue Modelle, Prozesse und Dienstleistungen auf Basis der Digitalisierung entwickelt. Dies führt zur Gründung vieler neuer Unternehmen sowie der notwendigen Anpassung bestehender Unternehmen, welche konkurrenzfähig bleiben müssen. Diese Veränderungen betreffen nicht nur B2C-, sondern auch B2B-Märkte. Diese Transformation geht von fünf wesentlichen technologischen Entwicklungen aus: Den mobilen Geräten, wie Smartphones und Tablets, die eine Interaktion an fast jedem Ort zu jeder Zeit ermöglichen. Den sozialen Medien, wie z.B. Twitter, wo Unternehmen noch direkter mit ihren Kunden zusammentreffen. Analytics und Big Data, welche hocheffizient und in Echtzeit Daten verarbeiten, um Entscheidungen zu treffen. Und schließlich dem Cloud-Computing sowie dem "Internet of Things", in dem alles was vernetzt werden kann, auch vernetzt wird.

Diese technologischen Entwicklungen haben auch wirtschaftliche Effekte. Der Kunde kann nun Informationen zu Unternehmen und Produkten gewinnen und sich besser zwischen verschiedenen Anbietern entscheiden. Die Digitalisierung bietet dafür vielerlei Wege. Serviceangebote werden ebenfalls immer wichtiger, so z.B. die Möglichkeit, Produkte zu vernetzen oder "remote" zu verwenden. Produkte müssen zudem nicht immer zwingend erworben werden, sondern sind über Sharing-Economies nutzbar. Aufgrund der Nachfrage des Kunden werden viele Produkte individualisiert, was zu einem starken Preisdruck führt. Hersteller müssen dem gerecht werden und setzen dabei auf Automatisierung ("Mass Customization").

Die Digitalisierung wird heutzutage vermehrt mit dem Begriff "Industrie 4.0" in Verbindung gebracht, welcher auf die vierte industrielle Revolution Bezug nimmt. Bei der ersten industriellen Revolution, die Ende des 18. Jahrhunderts begann, wurden erstmals mechanische Produktionsanlagen eingesetzt, die Wasser- und Dampfkraft verwendeten. Dieser Abschnitt wird "Industrie 1.0" genannt. Die darauffolgende industrielle Revolution ("Industrie 2.0") begann Anfang des 20. Jahrhunderts und führte die arbeitsteilige Massenproduktion ein. Maschinen wurden hierbei erstmals mit elektrischer Energie betrieben. Nachdem in den 1970er Jahren Elektronik und IT soweit waren, um in der Industrie eingesetzt zu werden, konnte die Produktion von Gütern (teil-)automatisiert werden. Dies wird als "Industrie 3.0" bezeichnet.

Heute soll die Industrie 3.0 zur Industrie 4.0 umgebaut werden, um die gesamte Wertschöpfungskette zu automatisieren und damit die Effizienz der Produktionsanlagen weiter zu steigern. Weiterhin soll die Qualität der Produkte verbessert und die Wettbewerbsfähigkeit der deutschen Industrie gesteigert werden. Um eine selbstorganisierende Produktion zu etablieren, ist es notwendig, dass Produktionsprozesse miteinander kommunizieren. So ist ein Unternehmen zum Beispiel in der Lage, flexibel auf Änderungen bei der Ressourcenverfügbarkeit zu reagieren. Durch die automatisierte Organisation der Fertigung können effizient auf Kundenwünsche eingegangen und individuelle Produkte kostengünstig hergestellt werden.

Bei der Transformation der Industrie zur Industrie 4.0 werden nicht nur unternehmensinterne Prozesse umgestellt, sondern auch nach außen hin Informationen und Dienstleistungen verknüpft. Somit kann mit den Kunden digital interagiert werden, was genutzt werden kann, um neue Produkte und Dienstleistungen zu entwickeln. Um Kosten zu sparen, werden (Teil-)Prozesse zum Kunden ausgelagert: Sie erledigen zum Teil Arbeiten, die früher das Unternehmen durchgeführt hat. Beispiele sind Selbstbedienungskassen im Supermarkt, bei denen der Kunde seine Artikel selbst einscannt, das Buchen und Einchecken von Flügen über Online-Portale sowie das Tätigen von Überweisungen bei der Bank, welches von Kunden selbst vorgenommen werden kann. Für die Umsetzung ist es notwendig, die Prozesse soweit zu vereinfachen, dass diese von jedem durchgeführt werden können.

Die aktuelle Situation in Sachen "Industrie 4.0" sieht in Deutschland nicht gut aus. Seit einiger Zeit gehören Industrie-4.0-Produkte zwar zum Standardsortiment der Fabrikausrüster, doch die Unternehmen sind bei der Umsetzung und Implementierung von Industrie-4.0-Prozessen sehr zögerlich. Einige Unternehmen setzen schon Industrie-4.0-Produkte ein, hauptsächlich aber nur mit Fokus auf Effizienzsteigerung. Die Forschung, die Arbeitsgruppen und die verschiedenen Gremien treiben dagegen die Standardisierung von Prozessen und Schnittstellen voran.

4.2 Transformation

Im Bereich der Digitalisierung stehen Führungskräfte vor neuen Herausforderungen: Es müssen neue Ausrichtungen von Unternehmensstrategien vorgenommen und neue Orientierungen vorgegeben werden. Es sind Entscheidungen darüber zu treffen, was zu tun ist. Dies ist ein komplexer Prozess, welcher von vielen Faktoren abhängt. Neue Geschäftsmodelle sind notwendig, um auch im internationalen Bereich marktfähig zu sein. Technologien stehen hierbei nicht im Vordergrund. Vielmehr rücken Konzepte in den Fokus, um Dienstleistungen auf eine bessere Art und Weise erbringen. Dadurch werden Produkte und Dienstleistungen adaptiv und decken einen großen Kundennutzen ab. Es entstehen sogenannte hybride Leistungsbündel. Ein anschauliches Beispiel ist das Internet der Dinge: Produkte werden zu Services bzw. zu hybriden Leistungsbündeln.

Digitale Geschäftsmodelle

Eine zentrale Fragestellung ist, wie sich traditionelle Produkte und Dienstleistungen digitalisieren lassen. Im Nachfolgenden werden zwei Muster für Geschäftsmodelle im Zeitalter der Digitalisierung vorgestellt:

- **Digitally Charged Product:** Im Internet der Dinge verbinden sich digitale Geschäftsmodellmuster mit solchen aus der nicht digitalen Welt zu einem hybriden Konstrukt. Zu den Kernbestandteilen von Digitally Charged Products gehören:
 - **Physical Freemium:** Zunächst kostenfreier digitaler Service zu einem gekauften Produkt, um Interesse für kostenpflichtige Services zu wecken.
 - **Digital Add-on:** Kostengünstiges physisches Gut ist vereint mit zahlreichen kostenpflichtigen digitalen Services.
 - **Digital Lock-in:** Nur Original-Komponenten sind kompatibel und keine Fälschungen.
 - **Product as Point of Sales:** Physisches Produkt kann direkt den Verkauf eines weiteren Produkts einleiten.
 - **Object Self Service:** Ausführung vom autonomen Bestellungen wird ermöglicht.
 - **Remote Usage and Condition Monitoring:** Daten über Zustände und Umgebung werden übertragen.
- **Sensor as a Service:** Sensordaten werden gesammelt, aufbereitet und zur Verfügung gestellt. Im Fokus stehen demnach die Daten selbst. Ein Beispiel ist die Firma Streetline: Sensoren werden auf Parkplätzen installiert, um die Belegungen zu überwachen. Die ermittelten Daten werden anschließend verkauft.

Aspekte

Es existiert eine Vielzahl an Aspekten, unter denen die digitale Transformation betrachtet werden muss. Nachfolgend werden wichtige Punkte beleuchtet, die in der heutigen Zeit entscheidende Rollen einnehmen:

- **Digitale Verträge:** Digitale Verträge sind ein wichtiger Bestandteil der Digitalisierung. Digitale Kaufverträge führen beispielsweise selbst Aktionen aus, von der Abbuchung der Raten bis hin zur digitalen Sperrung des Produkts. Vorteile von digitalen Verträgen sind unter anderem: geringere (bis gar keine) Transaktionskosten, beschleunigte Wirtschaft, Intermediäre (Banken, Börsen, Notare, ...) werden überflüssig.
- **Edison-Prinzip:** Ein bewährtes Prinzip für die Digitalisierung ist die Verwendung von vorhandenen Komponenten: Digitale Geschäftsmodelle werden durch das Zusammenfügen von bereits bekannten Technologien und Produkten entwickelt. Es sind somit nicht zwangsläufig neue Technologien o. ä. erforderlich. Oftmals ist die Verknüpfung und Vernetzung von bekannten Technologien erfolgreich.
- **Soziale Medien:** Der Einsatz von sozialen Medien, wie Facebook, Instagram, Snapchat, WhatsApp, YouTube, Twitter, LinkedIn oder Xing bringt Unternehmen heutzutage viele Vorteile in der externen sowie internen Nutzung: Verbesserung der Zusammenarbeit, bessere Nutzung des Mitarbeiterpotenzials, höhere Motivation und Zufriedenheit, Reduktion des Koordinierungsbedarfs, besseres Wissensmanagement, Verbesserung der Kundenkommunikation und des Unternehmensimages.
- **Customerization:** Aufgrund der hohen Transparenz durch die Digitalisierung ist der Kunde zum Mittelpunkt geworden. Die Konsequenzen für Unternehmen sind vielfältig. Customerization hat an Bedeutung gewonnen: Ob Mitarbeiter oder Kunde, alle nehmen größeren Einfluss auf die Gestaltung von Informationssystemen, Produkten und Dienstleistungen.
- **Social Listening:** Laut einer Studie von Adobe hat sich der Beruf eines Marketingprofis stark gewandelt: Aufdringliches Marketing bzw. Push-Marketing existiert nicht mehr. Heutzutage geht es vor allem um aktives Zuhören bzw. Social Listening. Es geht darum zu wissen, worüber die Kunden reden, wo es Probleme gibt und wie das Unternehmen diese lösen kann. Unternehmen müssen folgendes tun: Großartige Kundenerlebnisse schaffen, auf vielen Kanälen Kontakte zu Kunden herstellen, Kunden das Gefühl geben, verstanden zu werden, sowie Kundenverhalten und -bedürfnisse vorhersehen.

Erfolgsfaktor Flexibilität

Die Akzeptanz digitaler Produkte ist hoch: Aus der Historie von Industrie 4.0 ist ein deutlicher Trend hin zu Produkten zu erkennen, welche auf Kundenwunsch hin gebaut wurden. Laut dem Fraunhofer IAO wollen Kunden von produzierenden Firmen vermehrt Losgrößen mit einer großen Variantenvielfalt und kurzer Lieferzeit. Eine effiziente

Produktionsplanung, produktionsnahe IT- sowie geeignete ERP-Systeme, ein klares Lean Management und optimale Montageprozesse sind erforderlich. Ein Beispiel hierfür ist das Konzept Storefactory von Adidas: Der Konzern will Produkte nach Kundenwunsch direkt in kleinen Fabriken in unmittelbarer Nähe von Geschäften fertigen.

Entsprechend werden Geschäftsprozesse auf Softwarestrukturen so angepasst, dass Services interoperabel automatisch miteinander vernetzt werden. Eine Verbindung von Verkaufsorten zu Produktionsstätten ermöglicht die Fertigung von individuellen Gütern. Unternehmenssoftware wird soweit adaptiert, dass diese in Produktionsketten integrierbar ist.

Unternehmen, wie z. B. Uber, Spotify und car2go, haben zudem neue IT-Strategien entwickelt, um gleichzeitig das Nutzungsverhalten von Kunden zu studieren, damit mit diesem Wissen neue Produkte entwickelt werden können, während bestimmte Services angeboten werden.

Im Kern digitaler Geschäftsmodelle geht es um Daten, um den Erkenntnisgewinn mit dem Ziel der Transparenz über die eigenen Prozesse und Produkte, über das Nutzungs- und Kundenverhalten sowie über Kundenwünsche und -reklamationen.

Flexibilität in der technologischen Ausstattung ist entscheidend. Es ist wichtig, dass diese homogen und skalierbar ist. Die Kosten sollten im Hinblick auf den Betrieb proportional zur Last sein. Cloud-Lösungen spielen hierbei eine große Rolle. Auch die Anpassungsfähigkeit ist von Relevanz: Bestehende Geschäftsmodelle können sich innerhalb von kürzester Zeit ändern und neue Anforderungen an die Infrastruktur stellen.

4.3 Anforderungen

Die Digitalisierung bietet diverse Anreize für Unternehmen. Die Aussicht auf effizientere Prozesse sowie einen höheren Grad an Vernetzung sind hier nur zwei Beispiele. Um den Vorgang der digitalen Transformation jedoch optimal voranzutreiben bzw. eine nachhaltige Verankerung im Unternehmen zu gewährleisten, sind verschiedenartige Anforderungen zu erfüllen. Nur wenn die Weichen innerhalb der verschiedenen Bereiche richtig gestellt sind, kann mit einem positiven Ergebnis gerechnet werden. Die Erfolgsfaktoren werden im Weiteren skizziert.

Ziele

Anstatt den Innovationen nur blind zu folgen, ist es wichtiger, zu analysieren, was das Unternehmen wirklich will und benötigt. Es muss klar sein, was eine Digitalisierung bezeichnen würde. Erst wenn man sich über eine Digitalisierung einig ist, macht es Sinn, eine konkrete Planung voranzutreiben.

Planung

Die digitale Transformation ist detailliert zu planen. Die Kernprozesse eines Unternehmens müssen zuerst identifiziert werden. Anschliessend muss abgewogen werden, wie weit es Sinn macht, den Prozess bzw. einen Teil davon zu transformieren. Dabei hilft eine Entscheidungsmatrix, welche die benötigte Investitionssumme enthält und den Qualitätsabstand zur Konkurrenz. Ein anderer Ansatz stellt die benötigten Investitionsmittel dem Innovationspotenzial gegenüber.

Weitblick

Der Begriff "Digitalisierung" wird oft zu weit eingegrenzt. Unternehmen neigen dazu, nur an die Automatisierung von Fertigungsprozessen innerhalb einer Fabrik zu denken und lassen dabei viele andere Konsequenzen und Prozesse außer Acht. Synergetische Effekte müssen berücksichtigt werden und dies geht nur, wenn man einen Blick für das "Ganze" bekommt. Man muss die gesamte Wertschöpfungskette betrachten und dabei über gewohnte Fach- und Bereichsgrenzen hinaus gehen. Viele Möglichkeiten der Digitalisierung für Unternehmen liegen in produktionsfernen Bereichen, wie dem Vertrieb, der Preissetzung, der Planung, dem Controlling und auch dem Einkauf. Aus diesem Grund muss die Digitalisierung des Ganzen Unternehmens vorangetrieben werden.

Dabei kann die Digitalisierung von produktionsfernen Bereichen in vier Stufen unterteilt werden:

- **Stufe A (Information):** Dazu gehören das Bereitstellen allgemeiner

Unternehmensinformationen, Produkt- und Dienstleistungskataloge, Kontaktinformationen oder das Unterhalten von Stellenbörsen.

- **Stufe B (Kommunikation):** Hierzu zählen Dienste, wie Suchfunktionen, Formulare, FAQ, E-Mails, Newsletter, Chats, Diskussionsforen, Corporate Blogs und soziale Netzwerke, die die Interaktivität mit den Kunden über das Web fördern.
- **Stufe C (Transaktion):** Bei dieser Stufe geht es um die elektronische Geschäftsanbahnung und -abwicklung mit Online-Offerte-Erstellung, Bestellwesen, Bezahlung und Distribution.
- **Stufe D (Integration):** Die höchste Stufe betrifft die Integration und Kundenbindung, z. B. durch personalisierte Websites, One-to-One-Marketing, Online-Order-Tracking sowie den Einsatz digitaler Agenten für Beratung und Verkauf individueller Produkte und Dienste.

Dabei gilt, dass umso höher die Stufe (**Stufe D**), desto mehr Wert wird für das Unternehmen generiert. Der Aufwand für das Erreichen der Stufe wächst allerdings proportional mit. Dies hängt auch mit der Kultur im Unternehmen zusammen und ob Mitarbeiter diese Form der Digitalisierung annehmen und fördern, anstatt zu denken, dass das die Arbeit der IT-Abteilung sei.

Mitarbeiter

Die Mitarbeiter eines Unternehmens sind ein entscheidender Faktor im Hinblick auf die digitale Transformation. Ihre positive Einstellung gegenüber der Digitalisierung des Unternehmens ist offensichtlich eine Notwendigkeit. Zum einen führt eine fehlende Vision bezüglich des Wandels zu Verwirrungen innerhalb des Personals. Wenn die für den Wandel notwendigen Fähigkeiten zudem im Team fehlen, ist Besorgnis die Folge. Ohne vernünftige Anreize besteht im Weiteren die Gefahr, dass das Personal Widerstand leistet und das Projekt nicht unterstützt. Auch fehlende Ressourcen seitens des Unternehmens wirken sich negativ auf die Mitarbeitermotivation aus.

Es existieren vier verschiedene Reifetypen, die sowohl auf ein Unternehmen als Ganzes, als auch auf einzelne Mitarbeiter bezogen werden können:

- **Digitale Konservative** zögern, sich umzustellen, und lassen somit Chancen verstreichen.
- **Digitale Anfänger** besitzen eine niedrige digitale Intensität, sowohl bei der Nutzung neuer Technologien als auch beim Führungsstil.
- **Digitale Fans** befürworten einige digitale Initiativen, aber sehen keine Maximierung des Geschäftsnutzens daraus resultieren.
- **Digitale Experten** haben eine digitale Kultur geschaffen sowie Investitionen getätigt und profitieren dadurch von Wettbewerbsvorteilen – die digitale Elite sozusagen.

Mut

Viele Unternehmen sehen die Digitalisierung als Gefahr für ihr bestehendes Business-Modell. Vorsicht kann angebracht sein, kann aber auch schaden, wenn man Technologien verpasst und abgehängt wird. Oft entstehen solche Ängste auch aus Unwissenheit bezüglich der Gesetzeslage, wie z.B. im Bereich der Datensicherheit. In jedem Fall sollte man den Zug der Digitalisierung nicht komplett an sich vorbeifahren lassen, sondern solange es noch möglich ist, aufspringen.

Marketing

Ein offensives Marketing ist wichtig, um mit der Digitalisierung auch zu werben. Dies bescheinigt dem Unternehmen nach außen Innovationskraft und deutet an, dass man die Herausforderungen der Digitalisierung erfolgreich bewältigt.

Technik

Um ein Unternehmen der digitalen Wandlung zu unterziehen, sind diverse Anforderungen im Hinblick auf die Informationstechnologie zu erfüllen:

- Flexible, elastische Plattformen orchestrieren
- Übergreifende standardisierte Kommunikation
- Data Analytics zur Bewertung von Prozessen und Systemen
- Integration in die globale IT-Landschaft
- Orchestrierung von Systemen statt Programmierung
- Herstellerübergreifendes Engineering
- Verwaltung des gesamten Prozess- und Produktlebenszyklus
- Zusammenarbeit über Teams, Ressorts, Standorte, Organisationen hinweg ermöglichen

Die fortschreitende Digitalisierung der Unternehmensstrukturen schafft jedoch eine breite Angriffsfläche für Feinde aller Art. Um die Systeme vor Ausfall zu schützen bzw. die Integrität der Daten zu gewährleisten, sind Schutzmaßnahmen zu treffen. Hierunter fällt die Absicherung der Kommunikationswege, Schnittstellen und Systeme selbst. Weiterhin sollten auch interne Zugriffe nur unter Autorisierung erfolgen können.

Es ist hierbei wichtig, jede Ebene abzusichern: Defense-in-Depth Sicherheit. Zusätzlich zu dem Defense-in-Depth kann die Sicherheit durch ein Null-Trust-Konzept gesteigert werden. Dazu wird wiederum die Mikrosegmentierung auf der Software Ebene genutzt. Das heißt, die Prozesse können nicht alle miteinander kommunizieren, sondern nur mit den notwendigen Prozessen. Eine Punkt-zu-Punkt Kommunikation muss also ermöglicht werden. Auf dieser minimalen Kommunikationsebene greift dann das Null-Trust-Konzept und erlaubt ebenfalls nur minimalen Zugriff von außen (z.B. den Mitarbeitern), um so möglichst wenige Angriffsvektoren auf dem ganzen System zu haben.

Daten

Um die Kundenerfahrung zu optimieren, müssen zielorientiert Daten gesammelt und ausgewertet werden. Dies muss zeitnah geschehen, um auf Änderungen schnell reagieren zu können. Informationen müssen zentral gesammelt und für Anwendungen bereitgestellt werden.

Rechtssicherheit

Die rasanten Entwicklungen haben heute und zukünftig große Auswirkungen auf das Rechtssystem. Neue Technologien verarbeiten immer mehr und unterschiedliche Informationen. Elektronische Geräte werden zukünftig u. a. auch menschliche Emotionen und Gedanken erkennen. Rechtlich ungebremste Entwicklungen würden zu einer hohen Transparenz auf allen Ebenen führen: Im Privaten, im Unternehmen, in Staat und Gesellschaft. Es stellen sich die Fragen, wie der Umgang mit Daten geregelt wird und wessen Eigentum die Daten sind. Unternehmen sollten sich bei Unsicherheiten unter allen Umständen rechtlichen Beistand holen, um Risiken zu vermindern.

Agilität

Anpassungsfähigkeit ist vielleicht die wichtigste Anforderung der Digitalisierung. Veränderungen müssen von Unternehmen schnell erkannt und behandelt werden. Dabei werden neue Ideen schnell umgesetzt, was einen Bruch mit traditionellen Entwicklungen darstellt. Lösungen sollen schnell umgesetzt und durchgehend verbessert werden. Dafür lässt sich auch Cloud Computing effizient einsetzen. Der Ansatz SaaS (Software-as-a-Service) steht hier im Vordergrund. Weiterhin gibt es noch den PaaS Ansatz (Platform-as-a-Service) und zur Handhabung großer Datenmengen (Skalierung) IaaS (Infrastructure-as-a-Service).

4.4 Auswirkungen

Eine weitere Auswirkung des Wandels sind Änderungen beim Beschäftigungsverhältnis. Wie bereits erwähnt können Firmen dank der Digitalisierung selbständig, beispielsweise durch soziale Netzwerke, neue Arbeitskräfte finden, die in der heutigen Zeit flexibel und lernfähig sein müssen, dies aber auch vom Unternehmen fordern. Neue und flexiblere Regelungen bezüglich der Arbeitszeit sind dabei ein Aspekt und auch Lösungen wie das Home-Office Prinzip entstehen. Zudem gibt es immer mehr Personen die selbständig, beispielsweise auch als Freiberufler arbeiten und klassische Hierarchische Aufstiegschancen ablehnen. Arbeitnehmer suchen heute viel intensiver nach Arbeit die man tun möchte, Unternehmen müssen daher flexible Strukturen garantieren, sodass individuelle Potentiale voll ausgelebt werden können. Dies hat jedoch auch den positiven Effekt, dass Arbeitnehmer deutlich motivierter sind, auch ohne klassische Anreize wie gute Bezahlung.

Auch auf Business-Modelle hat die Digitalisierung Einflüsse, häufig geht es nicht mehr darum selbständig bestimmte Produkte zu vertreiben, sondern darum, Plattformen aufzubauen die von Personen genutzt werden und mit Revenue-Share Modellen Gewinn einbringen. Beispielsweise kann dies bei Verkaufsplattformen so umgesetzt werden, dass Nutzer selbständig Waren handeln können, wobei ein bestimmter Prozentsatz jeder Transaktion an den Betreiber der Plattform abgegeben wird. Hauptgründe für die Probleme klassischer Geschäftsmodelle sind dabei die freie Verfügbarkeit von Produktionsmitteln, freier Zugang zu Wissen und Dienstleistungen und der individuelle Wertewandel im Arbeits- und Sozialleben.

Entsprechende neue Business-Modelle haben daher meist ein Wertschöpfungsnetzwerk statt einer klassischen Wertschöpfungskette, sie haben offene Strukturen, sind kommunikativ und die Grenzen zwischen Produzenten, Konsumenten und Investoren verschwimmen.

Häufig spielen auch nicht mehr klassische Rohstoffe eine Rolle, sondern es geht viel häufiger um Daten, Wissen und (Vertrauens-)Beziehungen. Bezeichnet wird dies durch den Begriff Social Currency. Wie bereits erwähnt steigt beispielsweise der Wert eines sozialen Netzwerkes unter anderem durch seine Nutzer. Auch die angesprochene angestrebte Reichweite eines Unternehmens ist hierbei zu nennen. Je offener sich ein Unternehmen sich gegenüber dem Kunden verhält und je mehr mit ihm kommuniziert wird, desto besser wird seine Social Currency. Aber nicht nur für Unternehmen selbst spielt Social Currency eine Rolle, sondern beispielsweise auch für jeden Nutzer des Netzwerkes selbst. Je besser die eigene Social Currency ist, desto höher ist der Stellenwert der eigenen Person im Netzwerk. Konkret bedeutet dies zum Beispiel, dass man bei einem Verkaufsportal deutlich bessere Chancen hat Artikel zu verkaufen, wenn man bereits viele andere Verkäufe im Netzwerk

durchgeführt hat und für diese positive Rückmeldungen von anderen Nutzern bekommen hat. Hat man allerdings nur negative Rückmeldungen erhalten, oder hat man nur wenig verkauft, sinkt die Glaubwürdigkeit.

4.5 Die Cloud

Cloud als Motor der Digitalisierung

[entnommen aus "Was treibt die Digitalisierung?" Kapitel 1]

Das Rückgrat der Digitalisierung bildet die Cloud. Das "Internet of Things" wächst kontinuierlich, eine große Menge von Daten sammelt sich an, welche jederzeit von überall abrufbar sein müssen. Sie kann in Echtzeit Daten sammeln und auswerten für eine schier unbegrenzte Anzahl von Anwendern. Allerdings gibt es dafür einige Anforderungen:

- Leistungsstarke Breitbandverbindungen
- Effiziente und sichere Rechenzentren
- Flexibilität und Skalierbarkeit der Rechenzentren

Der Einsatz solcher Cloud-basierten Systeme ist in fast jedem Bereich denkbar: In der Medizin, der öffentlichen Sicherheit, Logistik, Einkauf oder der Verwaltung von Großunternehmen.

Um die Digitalisierung zu meistern müssen Unternehmen ihre Struktur, ihr Geschäftsmodell und ihre Prozesse transformieren. Durch die flacheren Hierarchien, Kommunikation und Verzahnung der unterschiedlichen Bereiche entstehen neue Wertschöpfungsketten mit dem Ziel neue, schnelle und flexible Produkte und Dienstleistungen anzubieten.

Der Markt ist internationaler als je zuvor und dadurch entsteht ein starker Wettbewerb. Unternehmen wollen Kunden gewinnen und behalten, und dafür wird die "Customer Experience" optimiert. Z.B. durch Big-Data Technologien sollen die individuellen Bedürfnisse jedes Kunden befriedigt werden und Dienstleistungen werden dabei auf einzelne Kunden zugeschnitten. Die Online-Komponente soll dabei zusammenarbeiten mit der Offline-Komponente. Am Beispiel des Autohauses soll es möglich sein, sich nicht nur die Autos anzuschauen, sondern auch per App-Anbindung und anderen Services mehr Informationen über Angebote und einzelne Objekte zu beziehen. Die Cloud soll also ebenso vom Kunden genutzt werden wie vom Unternehmen selbst.

Auch in der Logistik wo es um Warenströme geht, lassen sich Clouds einsetzen um Prozesse zu optimieren. In der Lebensmittel-Branche gibt es mittlerweile Branchen-fremde Unternehmen die den eigentlichen Unternehmen den Rang streitig machen. Logistiker die direkt vom Kunden die Bestellungen per Mobile-App annehmen und die Lebensmittel an einen vom Kunden gewünschten Ort abliefern. Mit Hilfe von Clouds ist es möglich schneller

auf Nachfrage und Angebot des Markts zu reagieren. Bei diesem Beispiel wird auch deutlich, warum Unternehmen sich verändern müssen, um sich in ihrer Branche noch zu behaupten. Um bestehen zu können gibt es sieben Schwerpunkte zu beachten.

1. Digitalisierte Geschäftsmodelle

- Fast jeder Geschäftsprozess benötigt heutzutage IT. Unternehmen müssen umdenken um weiterhin auf dem Markt eine Rolle zu spielen.
- Agilität und Schnelligkeit sind hier wichtige Schlüsselwörter.

2. CIO und CEO

- Der CIO spielt eine wichtige Rolle. Der CEO gibt die grundlegende Richtung vor, doch es liegt am CIO die Vorschläge auf Umsetzbarkeit zu prüfen und neue Prozesse im Unternehmen zu integrieren.
- Zur Seite hat der CIO Chief Digital Officers (CDOs). Diese kommen oft aus der Kreativ-Branche und helfen neue Lösungswege zu finden.

3. Zweigleisig fahren

- Bimodale IT Strukturen sind sinnvoll. Die altbewährte IT kann ihre Arbeit fortsetzen während man isoliert von ihr neue Wege ausprobiert. Letztlich müssen beide Wege wieder zusammengeführt werden.
- Ein kompletter Umbau der gesamten IT ist oft nicht ohne weiteres möglich.

4. Seite an Seite auf dem Weg in die Cloud

- Eine Zusammenarbeit mit externen Dienstleistern, welche oft eine zeitaktuelle Sicht auf die Technologie haben ist eine sinnvolle Option.
- Bei sicherheitskritischen oder speziellen Anforderungen ist es empfehlenswert, wenn der Dienstleister übergreifende Fachkenntnisse und Transformationserfahrungen besitzt.
- Es muss auch klar sein: Was soll in die Cloud? Welche Prozesse, welche Infrastrukturen, welche Teile der Wertschöpfungskette?

5. Kooperation mit den Besten der Besten

- Externe Provider müssen ein umfangreiches Know-How besitzen!
- Kooperation mit Technologiepartnern zur Erstellung einer optimalen Cloud-Lösung.

6. Maximale Performanz durch maximale Sicherheit

- Sensible Daten von Unternehmen oder Kunden müssen auch entsprechend behandelt werden.
- Clouds benötigen entsprechende Abwehrmechanismen gegen Angriffe (Verschlüsselung).
- Länderspezifische Gesetze und Richtlinien beachten!

7. Höchste Qualität als Grundlage für IT-Transformation und digitales Wachstum

- Zuverlässigkeit, Stabilität und Agilität sind wichtige Faktoren.
- Es ist wichtig Mechanismen zu haben die auf Ausfälle und andere Probleme reagieren.
- Die Digitalisierung ist abhängig von der Cloud, und die Cloud ist abhängig von ihrer

Qualität.

Cloud in der Praxis

[entnommen aus "Was treibt die Digitalisierung?" Kapitel 6, 10, 11]

Die Cloud setzt Trends, ermöglicht neue Geschäftsmodelle und ist der Motor der Digitalisierung, doch wie sieht sie in der Praxis aus? Ein wichtiger Bestandteil sind branchenübergreifende und bracheninterne Beziehungen, die richtige Vertriebsstrategie auf neuen digitalen Wegen ein weiterer. Und zu Letzt muss die Basis stimmen, also die Technologie im Rechenzentrum in Form von virtuellen IT-Ressourcen. Faktoren wie Verfügbarkeit und Skalierbarkeit spielen eine wesentliche Rolle.

Über kurz oder lang unterliegen alle Unternehmen und Geschäftsmodelle der digitalen Transformation. Das ist „die“ Chance der IT-Industrie, denn die Cloud als technologischer Grundstein, bekommt dabei eine Schlüsselrolle. Die Cloud ebnet den Weg für digitalisierte Geschäftsprozesse, Produkte und durchdigitalisierte Unternehmensstrategien.

IT und Kunde im Fokus

Ohne IT kein Online Banking, autonomes Fahren, keine wirtschaftliche Produktion und keine Geschäftsfähigkeit. Und ohne IT keine Kunden! Schließlich agieren die meisten Kunden bereits heute und bald online. Unternehmen sind oftmals noch nicht bestens informiert und „always on“. Unternehmen müssen sich also die Vorteile der Digitalisierung genauso zu Nutze machen und z.B. die Cloud als Vertriebskanal nutzen. Die Digitalisierung erfordert also nicht nur eine Transformation der eigenen Geschäftsprozesse, sondern auch neue Vertriebswege, Kundenansprachen und Kooperationsmodelle.

Die Digitalisierung im Berufs- und Privatleben hat zu Veränderungen geführt, die sich intensiv auf die Unternehmensausrichtungen auswirkt. Dabei beeinflussen im wesentlichen drei Trends derzeit den Markt:

-Erhöhte Transparenz: Preisvergleiche, Leistungsbeschreibungen und Kundenmeinungen Internet macht es möglich

-Akzeptanz der Standards: Schnelllebigkeit des Marktes und neue Kommunikationskonzepte sorgen für eine deutlich erhöhte Akzeptanz für Standards

-Produktvielfalt: immer mehr Teilnehmer durch neue Marktplätze im globalen Internet

Mit dem Kunden gleichziehen / Partnering

Die Zeiten des Handlungsreisenden und des klassischen Verkäufers sind vorbei. Es gilt, auf Kundenwünsche direkt reagieren zu können, eben „always on“ zu sein, wie der Kunde. Das Tempo des „Feedbacks“ der Kunden hat sich rasant weiterentwickelt und ist geschäftsentscheidender denn je. Das Kundenfeedback darf nicht mehr nur noch direkte Auswirkungen auf Maßnahmen der Kundenbetreuung haben, sondern sollte gleichermaßen transparent für die Produktion, Zulieferer oder eigene Forschungs- und Entwicklungsabteilung gemacht werden, unter höchsten Datenschutzvoraussetzungen. Die Omnichannel-Präsenz ist das Gebot der Stunde, um Kunden bestmöglich einzufangen. Ein Online-Shop oder Verkaufsraum reicht nicht mehr aus. Es ist also ein Gesamtpaket perfekt aufeinander abgestimmter Maßnahmen über sämtliche Kanäle gefragt, es sollte möglichst innovativ und intelligent sein. Die Brücke zu den neuen Kanälen ist die Cloud. Der Kunde will quasi ein „Rundum-sorglos-Erlebnis“. Im Idealfall könnte es wie folgt aussehen: Der Kunde konfiguriert sein Wunschauto am PC, kann am Tablet weiter daran arbeiten und diskutiert dabei Fragendirekt mit dem Händler über einen Online Chat. Abends an der Bushaltestelle gestaltet er das Interieur und lässt Farben und Materialien auf sich wirken. Einige Tage später kann er im Showroom des Autohändlers sein Auto virtuell Probe fahren. Nach der Klärung der letzten Fragen kann der Verkäufer weitere Features anbieten, an die der Kunde selbst noch nicht gedacht hat, weil der Händler seinen Kunden einfach schon besser kennt.

Omnichannel-Maßnahmen sind nur ein Teilschritt um Kunden besser zu binden. Die globale Wirtschaft macht Produkte auf aller Welt per Klick verfügbar. Auch das Telefon wurde längst durch die Cloud abgelöst. Dadurch sind Partnerschaften das A und O und sollten in Zukunft ein fester Bestandteil jeder Unternehmensstrategie in der digitalen Welt sein, denn erst gebündelte Kräfte schaffen eine ganzheitliche Lösung, die dem Kunden einen Mehrwert bietet. So kooperiert nicht nur der Softwarehersteller mit dem Hardwareproduzenten, um Anwendern eine bessere IT Lösung zu bieten. Auch ein Getränkehersteller partnert mit dem Automatenhersteller und der wiederum mit einem IT-Dienstleister. Partnerschaften sind nur erfolgversprechend, wenn Unternehmen in Sachen Kooperation auf das richtige Pferd setzen, denn der Markt bleibt schnelllebig.

Worauf kommt es in der Praxis wirklich an?

Marktführer wie Apple zeigen immer wieder, dass die IT und vor allem die Cloud, die Basis für das digitale Wachstum sind. Sie schafft die Voraussetzung, dass Dinge vernetzt sind und das vernetzte Dinge untereinander verfügbar sind und kommunizieren können. Die Cloud ist also der Motor der Digitalisierung. Für Unternehmen stellen sich nun Fragen wie: Wie passt die Cloud Technologie in die IT-Strategie? Welche Cloud Infrastrukturen und Applikationen verschaffen Mehrwert? Welche Cloud Modelle: Public, Private, Hybride? Welche Daten in welcher Form? Welcher Cloud Mix erzielt für mein Unternehmen das Optimum? Public

Cloud: mehr Innovation, mehr Agilität, schlechter Schutz sensibler Daten? Private Cloud: elektronischer und physischer Schutz? Hybride Cloud: beide Bereitstellungen vereint (public für weniger sensible Daten, private für geschäftskritische Operationen?)

Cloud braucht eine zentrale Plattform

Standardisierte und dynamische Cloud-Plattformen, mit variablen Leistungseigenschaften für schnelle Zugriffe von überall, sollten die Basis sein. Unabhängig vom Modell sollte sie flexibel und grenzenlos skalierbar sein. Der Vorteil solcher Plattformen ist, dass sie jeden Cloud-Service von jedem Server zu jedem Zeitpunkt bereitstellen können (any service on any server at any time). Das bedeutet einen einheitlichen Zugriff auf alle Cloud Angebote wie Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS) und Infrastructure-as-a-Service (IaaS). Hier müssen also auch Provider entsprechende Partnerschaften eingehen, damit Kunden auch wirklich aus den Vorteilen der Cloud schöpfen können.

Twin-Core-Technologie

Anforderungen an die Cloud-Infrastruktur sind neben der Kosteneffizienz vor allem Ausfall- und Datensicherheit, Datenschutz, Skalierbarkeit und Verfügbarkeit von absoluter Priorität umso mehr, wenn sie über einen Provider bezogen wird. Die Basis dieser Cloud-Dienste müssen ausfallsichere, moderne und effiziente Rechenzentren sein. Man gewinnt nur einen Mehrwert, wenn die Technologie stabil läuft. Twin-Core-Rechenzentren arbeiten genau dem einfachen Prinzip: „Doppelt gesichert hält besser“. Dabei findet eine permanente Spiegelung zwischen zwei Zwillingsrechenzentren statt und ist somit auch gegen Unwetter oder Hochwasser geschützt. Zum Beispiel bildet das größte Rechenzentrum Deutschlands in Bielefeld ein Twin-Core mit dem Rechenzentrum in Magdeburg.

IT-Sicherheit „everywhere“ erfordert Strategie

Jeden Tag entstehen Hunderttausende neuer Viren, Würmer und Trojaner. Experten des Computer Emergency Response Team (CERT) haben festgestellt, dass die Angreifer immer professioneller und die Methoden immer ausgefeilter werden. Cyber Security Services sind daher ein elementarer Bestandteil für das Vertrauen in einer vernetzten Welt. Bisher gleicht die Sicherheitsarchitektur vieler Unternehmen noch einer mittelalterlichen Stadtmauer, harte Schale, weicher Kern. Um auch in der zunehmenden Digitalisierung genug Sicherheit zu haben, braucht es intelligenter, umfassendere Konzepte, die in die Gesamtstrategie des Unternehmens eingebettet sein müssen. Unternehmenssicherheit setzt eine strategische Überlegenheit voraus und braucht daher eine individuelle Sicherheitsanalyse (Advanced

Cyber Defense). Aus diesen Ergebnissen lassen sich taugliche Sicherheitsstrategien ableiten, die dann mit oder mit Hilfe von Fachleute und Spezialeinheiten umgesetzt werden können.

IT-Qualität: IT ohne Ausfälle gibt es nicht

Unternehmen müssen sich im Klaren darüber sein, dass eine IT ohne Ausfälle Wunschdenken ist. Es gibt viele Gründe für einen Ausfall, daher reicht es nicht, einem potentiellen Ausfall nur mit Technologien vorzubeugen. Darüber hinaus benötigt man Strategien, um Störfälle möglichst schnell und systematisch zu beheben. Man benötigt Programme zur Sicherung der IT-Qualität auf allen Unternehmensebenen. Beispielweise Change Management, Incident Management und festgelegte routinierte Prozesse, mit regelmäßiger Überprüfung und Messungen in KPIs. IT-Verfügbarkeit ist nicht nur eine Frage der Qualität auf der Technologieebene, sondern auch eine Frage der Qualität auf organisatorischer und personeller Ebene.

Mehr Effizienz ist gefragt

Die Einführung einer Cloud-Lösung ist aus der Sicht der Unternehmen auch maßgeblich durch die Kostenoptimierung beeinflusst. Es ist notwendig, dass alle Prozesse hocheffizient ablaufen. Gerade im Eigenbetrieb lässt sich nicht immer eine Standardisierung über eine gesamte Produktpalette hinweg umsetzen. Hierin liegt ein weiterer Vorteil in der Zusammenarbeit mit einem Dienstleister. Je höher der Grad der Standardisierung in einem Provider-Rechenzentrum, desto höher profitieren auch die Unternehmen in späteren Schritten. Sind Standards implementiert, ist eine Konsolidierung auf wenige Systeme möglich, was wiederum die Basis für eine bestmögliche Automatisierung liefert. Kaum ein Unternehmen ist in der Lage, die ideale Kombination aus standardisierter Technologie und dem erforderlichen Grad an Automation von Prozessen komplett eigenständig und wirtschaftlich zu betreiben. Daher spielen Skaleneffekte eine entscheidende Rolle. Erst mit einer genauen Umsetzung der Konzepte und einer hohen Anzahl an Kunden, werden die Vorteile spürbar. Optimierungen sind nur möglich, wenn Ergebnisse kontinuierlich anhand von KPIs gemessen werden. Effizienz in allen Lagen ist also die entscheidende Grundlage für eine kostenoptimierte Cloud.

Harmonisierung und Standardisierung durch die Cloud am Beispiel ThyssenKrupp

Jedes Unternehmen hat "IT-Inseln", die "aus der Flut der Daten (ragen)" (S. 129). Sie sind aufwendig zu administrieren und führen zu unnötig viel Aufwand für CEOs und IT-Mitarbeiter.

Nachfolgend soll am Beispiel von ThyssenKrupp aufgezeigt werden, wie dieser Zustand unter Zuhilfenahme von Cloud-Technologie verbessert werden kann.

Mit Hilfe der Cloud sollen vor allem lokale Beschränkungen aufgehoben werden und alle Daten und Dienste von überall auf der Welt verfügbar machen.

Außerdem soll eine zentralisierte Administration den notwendigen Aufwand verringern, sowie umfassender agieren können.

Die Bewältigung dieser "vierten industriellen Revolution", die auf Information und Kommunikation setzt,

ist gleichzeitig Herausforderung und Chance. Deshalb

"(sollte) man sich kompetente und integrative Partner holen. Das ist der Grund, weshalb sich ThyssenKrupp für T-Systems entschieden hat. [...]

[Sie] brauchten eine hochverfügbare Basis, wie T-Systems sie mit ihren Twin-Core-Rechenzentren bieten kann" (S. 130)

Die Umstellung soll in zwei Schritten erfolgen, zuerst aus den 700 verschiedenen Datencentern

und Serverräumen in eine Public Cloud und von dort aus in eine Private Cloud (vgl. S. 131).

"Das Potenzial der Cloud lässt sich gut am Beispiel Aufzug darstellen:

Er verbindet nicht nur Etagen, sondern auch Menschen. Und ebenso wie ein Aufzug müssen Daten und Anwendungen dort sein, wo sie gebraucht werden." (S. 131)

Außer den Datencentern sollten zusätzlich auch sämtliche Arbeitsplätze in die Cloud verlagert werden.

"Daten sind verfügbar, Anwendungen einsetzbar, Dokumente und Informationen stehen bereit, wenn der Mitarbeiter sie braucht. Und genau das schafft ThyssenKrupp mit dem Dynamic Workplace von T-Systems." (S. 132)

Dabei bleibt die Bedienung unverändert, einzig wird das Desktop-Betriebssystem und die Anwendungen

in der Cloud virtualisiert. Das schafft Unabhängigkeit vom Endgerät und erhöht die Mobilität der Mitarbeiter. Bei diesem Vorgehen können während der Umstellungsphase alte Infrastrukturen weiter genutzt und sukzessive übernommen werden.

Auch ist für ein internationales Unternehmen nicht zu vernachlässigen, wo seine Daten gespeichert sind.

"So kann ThyssenKrupp im Rahmen des Konsolidierungsprogramms uniTe auf fünf Twin-Core-Rechenzentren von T-Systems zurückgreifen. [...]

Die solchermaßen geerdete Datenwolke verdeutlicht, dass der Begriff Cloud mitunter zu unpräzise ist." (S. 133)

"Cloud Computing bedeutet ja im Grunde, dass man Rechenzentrumsleistungen, also Serverleistungen, in ein Netzwerk verlegt und in dem Netzwerk Softwareleistung erbringt. Cloud Computing ist kein Mirakel, sondern ein Ordnungskriterium." (S. 136)

Ein Weg ohne Cloud wäre für Unternehmen langfristig nicht sinnvoll, da die Dynamik am Markt zu hoch ist

und für tiefgreifende Analysen eine sichere globale Infrastruktur benötigt wird.

Die Unternehmens-IT erfüllt im Zusammenspiel mit einem Cloud-Provider die Rolle des Technologiepartners

der Fachabteilungen und steuert die Unternehmens-Cloud.

"Es ist die Rolle der IT, das Backbone hochzufahren und dann am Frontend zu den Partnern die Dynamik zu liefern." (S. 137)

Sicherheitsziele in der Cloud

Je mehr Daten in der Cloud liegen, desto wichtiger ist auch deren Sicherheit.

Das gilt sowohl für Ausfallsicherheit (Safety), als auch Sicherheit vor Angriffen (Security).

Die Erwartungen sind dabei an einen externen Dienstleister deutlich höher, als an eine interne IT-Abteilung. Es geht dann um Garantien von bis zu 99,999% Erreichbarkeit (vgl. S. 143, 144).

Sobald ein externer Cloud-Provider beauftragt wird, wird also vom Unternehmen die Erreichbarkeit

blind vorausgesetzt, frei nach dem Motto: "Warum sonst würde ich meine Daten auslagern, wenn es nicht billiger, besser und ausfallsicherer ist?" (S. 144).

Datensicherheit ist von besonderem Interesse für Unternehmen, deren Geschäftsmodelle ausschließlich

auf Informationen und deren Verarbeitung ausgelegt sind, z.B. Facebook, Uber, Alibaba oder AirBnB.

Sie bauen stark auf das Vertrauen in den Schutz der Daten und die ständige Erreichbarkeit (vgl. S. 145).

Fazit: "Die Cloud ist wie die Erfindung des Buchdruckes. Plötzlich haben alle Menschen Zugang

zum Wissen der Welt, aber um erfolgreich zu sein, muss man lesen (und verstehen) können. 'Zero Outage' von T-Systems ist – um in diesem Bild zu bleiben – nicht der Deutschlehrer, sondern der Buchhändler des Vertrauens." (S. 146)

Fazit

[entnommen aus "Was treibt die Digitalisierung?" Kapitel 6 & 12]

Es gibt nicht nur „die eine“ Cloud. Die Cloud ist abhängig vom Unternehmen und von den Anforderungen der Unternehmens-IT. Wichtig ist, dass ein Unternehmen ihren richtigen Weg definieren und Cloud-Services mit der notwendigen Flexibilität auswählen und diese immer wieder anzupassen. Die Effizienz der Cloud ist nicht nur von ein paar Server-Racks im Rechenzentrum abhängig. Es braucht Erfahrung, Technologien und Qualität. Idealerweise ist die Cloud in Deutschland beheimatet, damit auch das Thema Datenschutz genügend berücksichtigt wird. Dieses Gesamtpaket wird auch benötigt, damit die Cloud tatsächlich für jedes Unternehmen Zukunftsfähigkeit bringt.

Abschließend sollen drei zentrale Punkte aufgelistet werden, welche die zukünftige Entwicklung von Cloud-Technologie charakterisieren:

- **Cloud ist Normalität und der Markt wächst:**

Sowohl bei Endverbrauchern, als auch im Business-Umfeld hat sich die Cloud mittlerweile als Normalität etabliert.

Es gibt mittlerweile Geschäftsmodelle, die nur dank Cloud möglich sind und es werden sicherlich noch mehr dazu kommen und wachsen.

- **Die Cloud ist und bleibt Kooperationsthema:**

Für eine flexible und effiziente Cloud-Lösung benötigt man einen Partner, der entsprechende Erfahrungen und Manpower hat, um sicher und zuverlässig zu sein.

- **Die Cloud muss einfach, sicher und bezahlbar sein:**

Eine Cloud ist nur dann sinnvoll, wenn sie ein Unternehmen sinnvoll bereichert und effizienter macht,

oder das Geschäftsmodell verbessern kann. Dabei muss natürlich die Sicherheit der Daten

flexibel auf die eigenen Bedürfnisse anpassbar sein.

4.6 Industriestand Deutschland & die Welt

Wie Deutschland von der digitalen Transformation profitiert

[entnommen aus "Was treibt die Digitalisierung?" Kapitel 3]

In Produktion, Logistik und Wissenschaft ist Deutschland international ein Spitzenreiter. Zusammen mit Japan und den USA gehört Deutschland zu den größten Entwicklern von eingebetteten Systemen. Es gibt aber auch negative Seiten: Nur 10 Prozent des weltweiten Umsatzes an Informations- und Kommunikationstechnologie (IKT) wird von europäischen Unternehmen erwirtschaftet und das Wachstum beträgt nur 1,3 Prozent. China bringt es hier auf elf Prozent und die USA auf vier Prozent. Auch in der Branche der Mobilfunkgeräte gehört kein europäischer Hersteller mehr unter die Top Ten. Die Mehrheit der IT Produkte kommt von Herstellern aus dem asiatischen und nordamerikanischen Markt.

Digitale Autobahnen ausbauen

Als Voraussetzung einer leistungsfähigen Wirtschaft ist die Infrastruktur. In diesem Fall die digitale Infrastruktur. Sie muss flächendeckend und leistungsstark vorhanden sein. International schneidet Deutschland hier ebenfalls schlecht ab. Nur auf Platz 28 schafft es Deutschland im weltweiten Vergleich der durchschnittlichen Verbindungsgeschwindigkeit. Im Jahr 2025 wird laut Studien der durchschnittliche Geschwindigkeitsbedarf bei 350 Mbit/s liegen. Um dies zu erreichen ist eine Investition von bis zu 35 Milliarden Euro notwendig. Wenn der Ausbau mit Glasfaserverbindungen stattfinden soll, sind es sogar bis zu 95 Milliarden Euro.

Um Anreize für Telekommunikationsunternehmen (TK) zu schaffen, müssen die Regulierungsrahmen innovationsfreundlicher ausfallen. Nur so schafft man genug Anreize, damit die TK-Industrie das nötige Kapital in diese notwendigen Entwicklungen investiert. Sind die Voraussetzungen geschaffen, können auch garantierte Verbindungsqualitäten gegeben werden, was für die Industrie 4.0 unerlässlich ist.

Das Netz benötigt Vertrauen und Sicherheit

Die Enthüllungen der Nachrichtendienste und andere Data-Leaks haben das Vertrauen beim Kunden in die Sicherheit stark beschädigt. Gerade in der EU gibt es viele unterschiedliche Standards im Bereich Datenschutz, was wirtschaftlich schädliche Folgen mit sich bringt.

Eine EU-weite Datenschutz-Grundverordnung sollte endlich beschlossen werden um einheitliche Standards zu schaffen.

Datentransfers, mitunter von sensiblen personenbezogenen Daten, zwischen Unternehmen sind unvermeidbar. Hier braucht es mehr Schutz und Vertrauen durch geeignete Reglementierungen seitens der EU. Durch das Marktortprinzip werden auch Unternehmen mit Sitz im Ausland gezwungen sich an europäische Standards zu halten.

Durch Cyberkriminalität entsteht Deutschland jährlich ein Schaden von ca. 50 Milliarden Euro. Um dies zu vermeiden müssen Staat und Wirtschaft zusammen arbeiten. Das nationale IT-Sicherheitsgesetz sowie Transparenzverpflichtungen, wie es der Plan KRITIS (Schutz der kritischen Infrastrukturen) vorsieht, leisten hier einen wertvollen Beitrag.

Starker digitaler Binnenmarkt

Die Digitalisierung macht auch vor Ländergrenzen nicht halt. Ein wichtiger wirtschaftlicher Faktor ist die Kundenanzahl. Im Mai 2015 hat die EU einen Vorschlag zu einem gemeinsamen digitalen Binnenmarkt gemacht. Geschäftsmodelle müssten nicht mehr für 28 unterschiedliche Staaten und deren Märkte angepasst werden. Es würde den Unternehmen wesentlich einfacher machen auf einem einheitlichen Markt zu operieren, welcher zudem auch noch größer wäre als der US-amerikanische.

Arbeit 4.0

Die Industrie 4.0 benötigt ebenfalls neue Fachkräfte. Arbeitszeiten und -aufgaben werden dynamischer gestaltet, die Komplexität der Prozesse steigt und es wird ein systemübergreifendes Verständnis verlangt. Darauf muss sich auch das Bildungssystem in Zukunft einstellen. Die Zunahme von Informationsgewinnung und -verarbeitung bringt neue Berufsbilder mit sich, wie z.B. den Data Scientist. Eine attraktive Anwerbung von ausländischen Fachkräften kann hier ebenfalls ein wichtiger Faktor darstellen, wenn die bürokratischen Hürden dafür überarbeitet werden.

Leitplanken für das Cloud Computing

Auf der technischen Seite dieser Transformation steht wieder die Cloud. Doch oftmals haben Unternehmen nicht genug eigene Kapazitäten um eigenes Cloud Computing zu betreiben. Dieser Service wird daher oft in Anspruch genommen von Dienstleistern. Die Cloud Dienste müssen dabei ebenfalls sicher und vertraulich sein. Eine Mehrheit der Kunden spricht sich für einen Standort von Clouds in Deutschland aus, aufgrund der starken Datenschutzrichtlinien, dem weltweit größten Internetknoten DE-CIX und der stabilen

politischen Verhältnisse. Negativ zu bemerken ist hier der hohe Strompreis, welcher 40 Prozent der gesamten Kosten eines Rechenzentrums ausmacht. Gerade in Frankreich ist der Strom wesentlich günstiger was für Abwanderung sorgen kann.

Zentrale digitale Plattformen und Start-ups

Um die Vorteile der Digitalisierung nutzen zu können bedarf es neuer Standards und Normen. Die Plattform Industrie 4.0 und der Industrial Data Space bieten Initiativen in denen zusammen mit Gewerkschaften und Politik versucht wird, diese Problematiken anzugehen. Gerade bei Start-ups gibt es in Deutschland Nachholbedarf. In den USA fließt wesentlich mehr Kapital in junge Unternehmen als in Deutschland (Venture Capital). Um auch hier aufzuholen, bedarf es politischer Aktionen, damit junge Unternehmen auch in Deutschland bessere Finanzierungsmöglichkeiten haben.

Die Cloud und die Autoindustrie

[entnommen aus "Was treibt die Digitalisierung?" Kapitel 6]

Die Cloud Nutzung und das Verlangen nach mehr Mobilität wirken sich immer mehr auf andere Branchen, insbesondere der Autoindustrie, aus. Die Auto Branche, Deutschlands Schlüsselbranche, sieht sich durch die Digitalisierung, mobiles Internet und eine umfassendere Vernetzung aller technischen Geräte, in jedem Glied der Wertschöpfungskette vor einer Zerreißprobe. Ständig drängen neue Geschäftsmodelle aus der IT Branche. Neue sogenannte „mobile Devices“ verursachen einen Wandel in der Auto Branche, da sie neu produziert, konstruiert und gewartet werden müssen. Elon Musk trifft einen treffenden Vergleich zu den neuen Anforderungen an die Autos und zwar „iPhones auf Rädern“. Nur noch Autos zu verkaufen, funktioniert also in Zukunft nicht mehr. Dadurch lastet auf der Autoindustrie ein enormer Druck, durch die sogenannten „Big Boys“ des Internets. Sollten die Autohersteller sich nicht schneller mit dem IT Wettbewerb mit bewegen, werden sie überholt, so hat Google bereits sein eigenes autonom fahrendes Auto vorgestellt und trifft damit genau die Kundengruppe, durch die die Autoindustrie am meisten Einnahmen erzielt. Es stellt sich die Frage wer in Zukunft die IT Strategien für die Autoindustrie entwickeln wird, die Autoindustrie selbst oder die Internet „Big Boys“?

Industrie 4.0

Das Zusammenspiel zwischen Mensch und Roboter wird immer wichtiger. Die Digitalisierung und Vernetzung führen zu einem Innovationssprung in der Robotertechnik. Die Übernahme von Jobs durch Roboter hat nicht nur was mit der Gesundheit der Mitarbeiter zu tun. In den nächsten Jahren gehen die geburtsstarken Jahrgänge in den Ruhestand und die Autoindustrie wird große Probleme haben, diese Stellen durch qualifiziertes Personal zu ersetzen. Laut Horst Neumann, Personalvorstand von Volkswagen, werden zwischen den Jahren 2015 und 2030 außergewöhnlich viele Beschäftigte das Unternehmen verlassen. Deshalb versucht man viele Stellen durch Roboter zu ersetzen, ohne dass dadurch die Arbeitslosigkeit steigt. Ein weiterer Vorteil liegt darin, dass die Arbeitskosten durch die Automatisierung gesenkt werden können. Neue moderne Produktionsprozesse bergen auch Risiken. Kaum eine Branche ist so stark von Industriespionage betroffen. Frühzeitig zu erkennen, welche neuen Technologien geplant werden, ist viel Geld wert. Daher gehen die Entwickler sehr skeptisch mit ihren Daten, gegenüber Zulieferern und der Produktion, um. So beschäftigt Daimler bereits eine Arbeitsgruppe aus geübten Hackern, die schnell Sicherheitslücken finden sollen. Roboter sind besonders anfällig, daher befürchten Sicherheitsexperten, dass Hacker mit einem Produktionsstop drohen können und somit Millionenschäden verursachen.

Das Auto fährt in der Cloud

In Zukunft müssen auch Autos vor Hackern geschützt werden, da sie bald bzw. teilweise jetzt schon permanent im Internet sind. Bereits 80 Prozent der Neuwagen sind vernetzt. Die junge Zielgruppe verlangt immer mehr nach Möglichkeiten, die Lieblingsapps auch im Auto nutzen zu können. Wolfgang Ziebart, ehemaliger Technik Vorstand von Jaguar, beschreibt treffend das Dilemma der Autoindustrie: „Drei Jahre entwickeln wir ein neues Auto, fünf Jahren produzieren wir es und zehn Jahre fährt es dann auf der Straße“. Keine noch so intelligente Infotainment- Entwicklung kann über eine so lange Zeitspanne aktuell bleiben. Die Autoindustrie hat erkannt, dass die Vernetzung des Automobils nach neuen Geschäftsmodellen verlangt. Von zentraler Bedeutung ist also die Update Fähigkeit der IT- Systeme. Da die Autobauer ihre eigenen Geschäftsmodelle etablieren, um das Heft des Handelns nicht ganz zu verlieren, kommt es immer wieder zu Machtkämpfen zwischen den Unternehmen. So kam es z.B. zu einem monatelangen Machtkampf zwischen Nokia und einem Konsortium deutscher Hersteller aus Audi, BMW und Daimler. Entgegen der aktuellen Navigationssysteme im Auto, die nur einen kleinen Teil an verfügbaren Daten anzeigen müssen, braucht das vernetzte, hochautomatisierte und autonome Fahren, Cloud- Plattformen, die die Verarbeitung und Aggregation verschiedenster Sensordaten, Informationen, etc. hochverfügbar halten soll. Es ist also keine Überraschung, dass „Big Data“ die Branche in Atem hält. Ein System, das einem Fahrer den nächsten freien Parkplatz anzeigt, wirkt dem großem Verkehr entgegen und erspart dem Fahrer viel Stress und Aufwand. Diese Informationen könnte man gegen eine Gebühr nur seinen Kunden zur

Verfügung stellen. Solche Geschäftsmodelle elektrisieren derzeit die Branche. Die Technik wird bald so weit sein, dass vollständige pilotierte Fahren zu ermöglichen, doch einen konkreten Starttermin möchte in der Branche niemand geben. Solche pilotierte Autos brauchen eine zuverlässige und schnelle Internetverbindung, um für Sicherheit und eine geringe Fehlerquote sorgen zu können.

Skepsis gegenüber neuen Technologien?

Deutschland hat Angst vor den Risiken der neuen Technologien. Ungeklärter Datenschutz setzt vielen neuen Geschäftsmodellen Grenzen. Bereits heute werden von der Werkstatt Fehler- und Datenspeicher ausgelesen und online übermittelt. Wem gehören diese Daten? Selbst Juristen sind sich noch nicht einig. Ein sorgfältiger und sicherer Umgang der Daten ist ein zentraler Faktor. Zu diesen Daten zählen auch Informationen wie Gefahrenstellen, Staus und Unfälle, aber auch Daten wie „automatisch die Zimmertemperatur erhöhen“, sobald sich der Fahrer seinem heim nähert.

Autohäuser sind vom Wandel auch betroffen

Auch die Autohäuser müssen verstärkt digitalisieren, um mit dem vorhandenen digitalen Marketing und Vertriebsstrategien klarzukommen. Verlangt wird so etwas wie eine mobile Plattform, die rund um die Uhr erreichbare Cloud-Dienste anbietet, die den gesamten Lebenszyklus der Kundenbeziehungen abdecken. Einfach eine Vernetzung des Kunden mit dem Autohersteller, dem Autohaus, dem Fahrzeug und der Werkstatt. Nicht nur im digitalen Vertriebsprozess, sondern auch autonom fahrende Autos produzieren schnell riesige Datenmengen. Diese Mengen können schlecht im Auto verarbeitet und gespeichert werden. Damit die Bandbreiten und Kapazitäten für die vernetzte Mobilität ausreichen, sind immer schnellere und stabilere Mobilfunknetze nötig.

Fazit zum Stand der Autoindustrie

Beim Thema Industrie 4.0 sind die deutschen Autobauer und Zulieferer gut dabei. Bei der Vernetzung des Autos und der App-Versorgung hat die IT Industrie die Nase vorn. Es ist nicht sicher wer in Zukunft die Mobilitätsangebote stellt. Der Vorsprung der Technik wird nicht mehr in PS, sondern in Gigabyte und Megabit pro Sekunde, gemessen. Das Internet und die Cloud verändern wie auch das Auto unsere Mobilität.

China als Frontrunner bei der Digitalisierung

[entnommen aus "Was treibt die Digitalisierung?" Kapitel 9]

Als digitale Frontrunner werden Unternehmen oder Gegenden bezeichnet, die überdurchschnittlich innovativ sind. Ein Beispiel dafür ist das Silicon Valley.

Als Innovationstreiber bei der Digitalisierung werden meist westliche Industrienationen, sowie Japan und Korea gesehen. China und Indien werden von den meisten Menschen nicht damit in Verbindung gebracht.

Sie werden meist als reine (billige) Auftragsfertiger gesehen.

Dagegen stammen aus China mittlerweile auch innovative Unternehmen, wie Lenovo, Huawei oder ZTE (vgl. S. 113).

Ein Hauptgrund für die Entwicklung Chinas ist die protektionistische Wirtschaftspolitik, die z.B. Google und Facebook ausschließt und so Raum für eigene Entwicklungen schafft.

Im *2013 Global Manufacturing Competitiveness Index Report* ist China dennoch weit abgeschlagen

hinter den USA, Deutschland und Japan. Zu lange hat man sich als "Werkbank der Welt" (S. 117) verdingt

und auf Technologietransfer verlassen.

Durch eine stärker werdende Währung übersteigen Chinas Lohnkosten mittlerweile die anderer asiatischer Länder. Darüber hinaus gibt es in den USA den Trend, Produktion wieder zurück ins eigene Land zu holen. Den drohenden wirtschaftlichen Folgen soll nun geeignete Politik entgegen wirken.

Erste Ergebnisse dieser Politik sind gut ausgebauten Mobilfunknetze und ein Angebot an staatlichen

Apps, z.B. für Taxibestellungen und mobiles Zahlen.

Auch das Volumen des Online-Shopping war 2014 mit knapp \$400 Mrd. doppelt so hoch wie das der USA (vgl. S. 118). Die zuständigen Regierungsstellen sind das *Ministry for Industry and Information Technology* und das *Ministerium für Cybersecurity*.

Darüber hinaus werden soziale Medien von Regierungseinheiten zur Kommunikation mit dem Volk

und zu dessen Abhörung genutzt. Sämtliche veröffentlichten Inhalte bedürfen in Chinas censiertem Internet einer Genehmigung oder müssen sich an entsprechende Regelungen halten.

Das Regierungsprogramm *Internet+* widmet sich dem industriellen Teil der Digitalisierung und der Weiterentwicklung zum "Dreiklang aus Hardware, Software und Daten" (S. 118).

Das soll durch die "Integration von Cloud Computing, Big Data, Internet of Things und mobilem Internet" (S. 118)

ermöglicht werden, man spräche dann von der "Mobile Big Data Cloud" (S. 118).

Das Ziel des Programms ist eine führende Stellung bei der IT-Nutzung innerhalb von fünf Jahren,
eine Prognose von Gartner hat ähnliche Erwartungen.

Ein weiteres Programm ist *Made in China 2025*, das ein "Upgrade der chinesischen Industrie in das neue Zeitalter" (S. 119) anvisiert.

Der Fokus geht allerdings, wegen vieler hinterher hinkenden Unternehmen, über Digitalisierung hinaus.

Es geht auch um ökologische Industrieentwicklung, Strukturoptimierungen und Talententwicklung.

Zusätzlich soll auch der Anteil an der Wertschöpfung für chinesische Unternehmen erhöht werden.

Entgegen "Geschichten von geklauten Ideen, verletzten Patenten und detailgetreuen Kopien" (S. 120) hat China, vor allem auf Grund der Größe der Marktes gute Chancen, innovativer Digitalisierungsführer zu werden.

Die Gründe lauten im Detail folgendermaßen:

1. Um Zugang zum chinesischen Markt zu erhalten, werden westliche Firmen Teile ihres Wissens offenlegen, darauf aufbauend können eigene Entwicklungen entstehen.
2. Der Schutz geistigen Eigentums wird mittlerweile sehr ernst genommen und "durch das Wahrnehmen von Verburtsrechten, Unterlassungsklagen, oder durch Forderung von Schadensersatz- und Lizenzzahlungen gezielt gegen Wettbewerber" (S. 122) durchgesetzt.
3. China will im Jahr 2020 mindestens 3% des Bruttosozialproduktes in die Forschung investieren.
Damit wäre es dann nach OECD Vorhersage von 2014 das Land mit der höchsten Investitionssumme, noch vor den USA.
4. Basierend auf ausländischer Technologie werden Produkte entwickelt, die deutlich weniger komplex sind.
Dadurch sollen wie weniger wartungsanfällig sein und werden vor allem auch günstiger.
5. Durch eine Reihe sehr fokussierter Regierungsprogramme (über die bereits erwähnten hinaus) sollen chinesische Unternehmen das Internet besser kommerziell nutzen und die Industrie modernisieren. Dabei sollen auch internationale Partner eingebunden werden und chinesische Standards auch international Verbreitung finden.
6. Der Arbeitsmarkt in Hightech-Clustern wird zunehmend attraktiver und zieht viele chinesische Topleute zurück. Auch ausländische Experten werden durch deutlich zugenommene Lebensqualität

und hohe Löhne für Spitzenkräfte angezogen.

Ein Hindernis kann allerdings noch die chinesische Internet-Firewall werden, da dadurch der offene Wissensaustausch erheblich behindert wird, "ähnlich wie nach dem Bau der Großen Mauer" (S. 125).

4.7 Beispiele für aktuelle und zukünftige Technologien

Neue Technologien im Zuge der Industrie 4.0

[entnommen aus "Die Digitalisierung der Welt" Kapitel 1]

Smarte Eingabegeräte

Wearables, also kleine tragbare elektronische Geräte, sind derzeit sehr beliebt und können gut in Unternehmen eingesetzt werden. Mögliche Anwendungsszenario für Wearables in Unternehmen sind zum Beispiel der Einsatz von „smarten“ Handschuhen, die Monteure bei der Montage anweisen und Datenbrillen, die dem Träger wichtige Informationen einblendet.

Heutige Wearables enthalten immer mehr Sensoren, um möglichst viele Daten erfassen zu können. Auch das direkte Integrieren von Wearables in Kleidung ist derzeit sehr beliebt. Ein sehr wichtiger Faktor bei dem Erfolg der Wearables ist die Bedienbarkeit der Geräte.

Die Forschung zum Thema „Mensch-Maschinen-Kommunikation“ ist sehr aktiv und es werden immer neue Eingabe- und Steuerungsmethoden entwickelt und alte ständig verbessert. So sind heute zum Beispiel schon Gestensteuerungen in Autos und Unterhaltungselektronik zu finden. Ein bekanntes Gerät ist unter anderem die Microsoft Kinect, welche die Positionen und Bewegungen von Personen mithilfe einer 3D-Kamera erkennt.

Auch beliebt sind Sprachassistenten, die per Sprache gesteuert werden und allerhand nützliche Informationen bereitstellen sowie Smart-Home-Geräte steuern. Beispiele für Sprachassistenten sind „Alexa“ von Amazon und „Google Home“ von Google. „Google Home“ soll, um möglichst hilfreich zu sein, mit den verschiedenen Google-Diensten wie Google Mail und Google Maps verknüpft werden. Die Steuerung der Heiamtomaticierung „Google Nest“ mit Google Home ist (vorerst) nur in den USA verfügbar.

Die Smartphone-App „Google Assistant“ verwendet zudem künstliche Intelligenz um gesprochene Wörter zu erkennen.

Neben den Sprachassistenten gibt es zudem noch die „Eye Tracker“, die Augenbewegungen mithilfe von Kameras und Bildverarbeitungsalgorithmen erkennen und so eine Steuerung von Geräten über die Augen ermöglichen.

Einen anderen Steuerungsansatz untersucht das Projekt “Skin Track”: Die eigene Haut soll als Touchscreen dienen. Die Technik “Thermal Touch” erkennt über Wärmebildkameras Flächen, die von einem Benutzer durch Anfassen erwärmt wurden. So können beliebige Gegenstände, die sich erwärmen lassen, zu Eingabegeräten werden.

Visualisierung

Es gibt zwei Visualisierungsarten: “Virtual Reality” (VR) und “Augmented Reality” (AR). Der Unterschied der beiden Visualisierungsarten liegt dabei an der Einbindung der Umgebung: Während bei der Augmented Reality die Umgebung mit zusätzlichen Informationen ergänzt wird bzw. reale mit virtuellen Elementen kombiniert werden, ist der Nutzer bei der Virtual Reality von der realen Umgebung komplett abgeschottet und vollständig in einer virtuellen Umgebung.

Schon Anfang der 1990er Jahre war Virtual Reality ein großes Thema. Heute helfen aber verbesserte Sensoren und eine höhere Rechenleistung dabei, Virtual Reality besser umzusetzen als vor 20 Jahren. Verschiedene empfindliche Sensoren ermöglichen es aktuellen VR-Headsets, Kopfbewegungen genauestens zu erkennen und die Position des Benutzers zu bestimmen. Im Gegensatz zu Wearables haben VR-Applikationen andere Anforderungen an das Benutzerinterface: Die Benutzung sollte hauptsächlich mit den Händen erfolgen und so sehr intuitiv sein.

Bei der Augmented Reality wird die reale Umgebung durch virtuelle Elemente ergänzt. Dies wird zum Beispiel bei den so genannten “Smart Mirrors” gemacht, die Informationen wie Nachrichten und das aktuelle Wetter direkt im Spiegel einblenden.

Neben den Smart Mirrors gibt es noch die Datenbrillen. Eine gutes Beispiel ist hierfür die “Google Glass”, die in den Medien viel Aufmerksamkeit erregt hat. Die erste Version der Datenbrille wurde vom Markt genommen, aber Google arbeitet derzeit an einer “Enterprise Edition” der Brille, die zum Beispiel im Logistikbereich eingesetzt werden soll. Die “HoloLens” von Microsoft dagegen unterstützt 3D-Projektionen, welche durch projizierte Lichtpunkte erzeugt werden.

Eine dritte Art von AR-Anwendungen sind Augmented Reality Browser, die als App auf einem Smartphone oder Tablet laufen und bestimmte Muster oder Bilder erkennen und dann entsprechende Informationen anzeigen. AR-Browser gibt es auch als “Location Based Services”, welche anhand des aktuellen Standortes des Benutzers Informationen anzeigen.

Smarte Gebäude

In Smart Homes sind alle Geräte miteinander vernetzt und können automatisiert Umgebungsparameter wie etwa Temperatur und Licht steuern. So weiß das System beispielsweise, wann der Eigentümer nach Hause kommt und kann entsprechend die Temperatur in den Räumen anpassen, bevor der Eigentümer nach Hause kommt.

Neben den Smart Homes gibt es noch die sogenannten "Smart Rooms", welche von Einzelhändler eingesetzt werden, um die Bereiche im Laden zu erkennen, die von vielen Kunden aufgesucht werden. Dies wird unter anderem mithilfe von Wärmebildkameras realisiert.

Des Weiteren gibt es Bluetooth Beacons, welche sich über Bluetooth-Low-Energy (BLE) mit auf den Smartphones der Kunden installierten Laden-App verbinden und zum Beispiel Nachrichten über aktuelle Angebote senden.

Die Smart Factory ist eine Fabrik, in der alle Maschinen miteinander vernetzt sind und automatisiert alle notwendigen Arbeitsabläufe geplant und ausgeführt werden.

In intelligenten Städten ("Smart Cities") sollen Verkehr, Ver- und Entsorgung und Logistik automatisiert gesteuert werden. Ziel ist es, das Leben bequemer zu machen. Werden alle Vorgänge in einer Stadt automatisiert, gehören zum Beispiel Staus der Vergangenheit an.

Zukünftige Technologien

[entnommen aus "Die Digitalisierung der Welt" Kapitel 2]

Die Entwicklung von neuen Technologien ist rasant. Schon heute sind die Technologischen Tendenzen zu beobachten, welche sich erst in einigen Jahren etablieren. Ein gutes Beispiel ist die Entwicklung der Smartphones und was sich innerhalb von wenigen Jahren entwickelt hat. Es ist wichtig zu beobachten, in welche Richtungen sich Technologien entwickeln und was in den nächsten Jahren Marktreife erlangen könnte und welche Auswirkungen diese haben werden.

Technologien und Innovationen

Nachfolgend werden einige Beispiele von neuen und interessanten Technologien aufgelistet, welche in der Zukunft eine große Rolle spielen könnten:

- **Kristallkerne als Datenspeicher:** Mithilfe von speziellen Lasern werden Daten in Kristallkerne eingebrannt. Die kristallinen Strukturen werden dabei neu angeordnet. Die Daten werden in Form von Punkten in Schichten gespeichert. Somit können neben den drei Raumdimensionen auch die Größe und Orientierung der Punkte genutzt werden. Dies wird von Forschern der University of Southampton als 5D-Speicher bezeichnet.

Dies könnte zukünftig eine neue Form der Langzeitdatenspeicherung sein..

- **Aquaman-Kristall:** Die University of Southern Denmark hat im Rahmen des sogenannten Aquaman- Kristall-Projekts ein Material entwickelt, welches Sauerstoff aus der Luft oder aus dem Wasser in hoher Konzentration speichern kann. Der Sauerstoff kann bei Bedarf auch unter Wasser wieder abgegeben werden. Der Aquaman-Kristall speichert so viel Sauerstoff, dass man über einen langen Zeitraum unter Wasser bleiben kann. Eine Teelöffel große Menge kann Sauerstoff eines mittelgroßen Raums speichern.
- **Graphen:** Das im Jahr 2004 erstmals hergestellte Material Graphen besteht aus einer einzigen Lage von Kohlenstoffatomen. Es ist sehr dünn, widerstandsfähiger als Stahl, leitfähig, biegsam und durchsichtig. Beispielhafte Anwendungen könnten sein: Faltbare berührungssempfindliche Displays, Schaltungen auf unterschiedlichen Produkten (z. B. Tasse, Milchpackung usw.) zum interagieren mit der Umwelt und intelligente Kontaktlinsen mit Zoom. Die Zahlen sprechen für sich: Im Jahr 2014 wurden über 9000 Patente mit Graphenbezug angemeldet.
- **Smart Glass:** Ein Material, welches beim Anlegen einer Steuerspannung die Transparenz ändert. Viele Anwendungsfälle sind denkbar. Die Entwicklung geht in Richtung immer größerer Flächen und geringerer Kosten. Es ist vorstellbar das in der Zukunft ganze Gebäudeflächen mit diesem intelligenten Glas versehen werden um tagsüber Wärme zu speichern und nachts wieder abzugeben.

Ein weitere zu beobachtende Entwicklung ist die der **künstlichen Intelligenz**. Das folgende Zitat ist von Volkswagen Group CIO Martin Hofmann:

Selbstlernende Systeme, deren künstliche Intelligenz die gesammelte Leistungsfähigkeit der Gehirne aller Menschen um ein Vielfaches übersteigen wird, stellen die Welt bald auf den Kopf.

Es ist anzunehmen, dass Computer in den kommenden Jahren die Intelligenz von Menschen erreichen. Mithilfe von künstlicher Intelligenz werden Systeme den Benutzer verstehen können, was in vielen Bereichen, z. B. im Marketing für die Analyse von Kundenbedürfnissen, eine große Bedeutung finden wird.

Beobachtung neuer Technologien und Innovation ist wichtig

Zusammenfassend ist zu sagen, dass es von großer Bedeutung ist, den technologischen Fortschritt sowie neue Innovationen zu beobachten, da diese große Auswirkungen in vielen Bereichen haben könnten. Alte Technologien können schnell von neuen ersetzt werden, welche in kürzester Zeit Marktreife erlangen können.

4.8 Zusammenfassung

Die Veränderung durch die Digitalisierung findet bereits statt und Unternehmen müssen schnellstmöglich beginnen sich ihr anzupassen, oder sie laufen Gefahr ins Vergessen zu geraten. Geschäftsmodelle müssen angepasst, Organisationsprozesse verändert und kollaborative Strukturen eingerichtet werden. Firmen müssen flexibler, offener, agiler und vor allem Kreativer und Innovativer werden um auf schnelle Veränderungen angemessen agieren zu können. Dabei gilt: nicht der stärkste oder größte gewinnt, sondern der Anpassungsfähigste. Um diesen Wandel zu vollziehen lassen sich drei Handlungsebenen definieren:

1. Ebene 1 Nutzen von externen (Kreativ)-Ressourcen, Dienstleistungen und entsprechenden Plattformen für die unternehmenseigene Zwecke nutzen, um Innovation und Flexibilität zu erlernen und fördern.

Beispiel: Nutzen von Crowdsourcing-Kampagnen für die Ideenentwicklung, Auslagern von Tätigkeiten in die Crowd oder Aufbau von Kooperationen mit Start-Ups zwecks Wissenstransfer

2. Ebene 2

Anpassen des eigenen Geschäftsmodells an die digitale Co-Economy in Form von Pilotprojekten oder Prototypen, während das Kerngeschäft weiter läuft.

Beispiel: Ein Unternehmen transformiert sein Kerngeschäft, im Ansatz ist dies bereits bei Automobilherstellern zu beobachten, die in den Carsharing Markt eingestiegen sind.

3. Ebene 3

Ein kompletter strategischer Wandel des unternehmenseigenen Geschäftsmodells, welcher über mehrere Jahre hinweg erfolgt und einen integrierten Gesamtplan erfordert.

Wenn Unternehmen all diese notwendigen Änderungen beherzigen und dementsprechend handeln, werden sich ihnen eine Vielzahl von neuen Möglichkeiten und Chancen bieten, die ihnen in der Zukunft zu großem Erfolg verhelfen können.

5 Einleitung

Einleitung...

5.1 Rechtliche und organisatorische Grundlagen

Mit der Trusted Web 4.0 müssen nur einige der bestehenden Gesetze verändert werden, da der Bürger viele Aufgaben anonym und dezentral erledigt. Die Vorratsdatenspeicherung erfolgt in einem Stufenmodell. Durch dieses Stufenmodell ist der Bürger grundsätzlich anonym, da aus den dezentral gespeicherten Daten kaum Rückschlüsse auf einen bestimmten Bürger vorhanden sind. Wenn der Bürger aber eine Straftat begangen hat, dann kann über eine nachweisliche Notwendigkeit die Daten ermittelt werden. Dieses Verfahren ist vereinbart mit dem Datenschutz und der Strafverfolgung. Damit Daten von außen nicht manipuliert werden und man einzelne Bürger abhört, werden die Daten in dem Trusted Web 4.0 dezentral zweckbestimmt gespeichert. Alle Daten des Trusted Web 4.0 werden in 1000 verschiedenen Kategorien abgelegt. Diese Kategorien werden nach Allgemeinen Oberbegriffe angegeben. Die Begriffe ergeben sich aus dem Suchverhalten der Bürger. Jeder Kategorie kann eine Aufgabe und eine Straftat zugeordnet werden. Jede Kategorie hat für jeden einzelnen Bürger eine feste IP-Adresse. Eine Straftat findet in einer Kategorie, in einer Region, in einem Sprachraum und zu einer bestimmten Zeit statt. Ein Richter kann die passenden Daten für die Strafverfolgung über die Kategorie, die Zeit und die IP-Adresse anfordern, da die IP-Adresse zu einer bestimmten Region und zu einem bestimmten Sprachraum gehört.

Gerade bei Big Data gibt es viele gesammelte Daten bei großen Unternehmen. Wenn man Daten nur noch Kategorien zuordnet, dann wird der gläserne Kunde dadurch verhindert. Das erhöht die Datensicherheit deutlich. Sollte sich aus den gespeicherten Daten einer Kategorie eine Straftat ergeben, so können die personenbezogenen Daten rechtlich angefordert werden. Den digitalen Schlüssel für diese personenbezogenen Daten hat nur der Bürger selbst, sonst niemand anderes. Daraus folgt, dass die personenbezogenen Daten nicht gespeichert werden, sondern der Bürger über seinen digitalen Schlüssel identifiziert wird. Dieser Vorgang führt zu einer Anonymisierung des Bürgers und seiner Daten. Die Daten werden auch dezentral gespeichert. Die Speicherung der Daten findet an unterschiedlichen Orten statt. Diese Schritte führen zu einer besseren Sicherheit gegenüber Datenmissbrauch.

Eine Strafverfolgung findet nur innerhalb eines Rechtsraums statt und nicht außerhalb des Rechtsraums. Der Bürger bleibt nach außen unentdeckt. Um das zu erreichen muss ein weltweiter Kategorienstandard etabliert werden. Für jede Kategorie sollte ein anderer Speicherort, Sicherheitsstandard und Authentifizierung verwendet werden. Die Trusted Web 4.0 nutzt dafür eine eigene Domain. Durch die Benutzung einer eigenen Domain kann bei

einer richterlichen Anordnung der Benutzer mit Verlaufsprotokollen verfolgt werden und auch sein Rechner kann geortet werden. Alle anderen Benutzer sind anonym unterwegs. Optimal wäre, wenn man für jede Kategorie einen Name-Server benutzt.

Das Urheberrecht schützt ein Werk, welches von einem kreativen Menschen geschaffen wurde. Viele Werke werden heute in der digitalen Welt einfach kopiert und weiterverbreitet. Die Reichweite der Werke können Eigentümer kaum noch verfolgen. Die Werke können heute kaum noch geschützt werden davor. Es gibt Möglichkeiten das geistige Eigentum in der digitalen Welt zu schützen, wenn man Dezentralisierung und Anonymisierung benutzt. Der Eigentümer sollte für seine Daten einen eigenen individuellen Schlüssel haben und er sollte Tools haben, die ihn ermöglichen zu entscheiden, wer seine Werke ansehen darf. Von diesen Werken dürfen keine Kopien gemacht werden, dass muss technisch verboten sein.

5.2 Konzepte der Zukunft

Ein persönliches digitales System (PDS) ist ein System, welches den Benutzer im Alltag unterstützt. Es speichert die Daten des Benutzers und kann mit vielen anderen Diensten Kommunizieren. Denn die Daten sind für Unternehmen sehr wichtig. So sind die rechtlichen Hürden des Urheberrechts und des Datenschutzes leichter zunehmen und der Nutzer bewegt sich im System mit seinen Daten anonym. Aber die Unternehmen können mit diesen Daten wiederum neue Geschäftsmodelle entwickeln. Denn um Big Data zu benutzen, muss das Unternehmen den Kunden nicht kennen. Das Unternehmen muss nur wissen, wie oft wird ein Produkt gekauft und was die Kunden noch dazu kaufen. Mit einer PDS lässt sich E-Government auch schneller beschleunigen. Die digitale Transformation lässt sich mit diesen Konzepten umsetzen.

Derzeit scheitert E-Health an den Datenschutz und der Datensicherheit. Mit einer PDS hätte man diese rechtlichen Hürden nicht, da die Daten anonym vorhanden sind. Die Gesundheitsdaten würde der Benutzer selber speichern und würde den passenden Arzt, die gewünschten Daten zur Verfügung stellen. Wenn der Benutzer den Arzt wechselt, so kann er seine Daten zum nächsten Arzt mitnehmen. Man würde die Daten der Kategorie „Gesundheit“ zuordnen abhängig von der Region. Der Benutzer wäre Eigentümer seiner Daten.

Das Auto ist ein Hilfsmittel für die Mobilität des Menschen und daher im Alltag unverzichtbar. Das muss auch so bleiben, der Benutzer benutzt das Auto als zusätzliche Unterstützung im Alltag und darf damit nicht überwacht werden. Beim Auto gibt es einen Trend zur Digitalisierung, da die Menschen sicherer unterwegs sein wollen. Technik hilft diese Sicherheit zu schaffen. Dies ermöglicht aber auch Angreifern von außen das Auto zu manipulieren, um das Lenkrad oder die Bremse zu aktivieren und fernzusteuern. Die Daten sollten anonym, dezentral gespeichert und standardisiert werden. Man kann auch hier wieder Kategorien definieren und diesen Daten den Kategorien zuordnen. Dadurch ergeben sich auch neue Geschäftsmodelle. Das Auto kann mit der PDS verbunden werden. Die Daten vom Auto kann der Nutzer dann selber kontrollieren. Selbstfahrende Autos ermöglichen in der Zukunft neue Geschäftsmodelle und neue Dienstleistungen, wenn die rechtlichen und gesellschaftlichen Probleme beseitigt sind. Die Verkehrssysteme müssen sicher sein. Dazu müssen alle Teilnehmer im Straßenverkehr mit dem Auto vernetzt sein, der Mensch muss immer in das Geschehen des Autos eingreifen können und alle menschlichen Verkehrsteilnehmer müssen anonym untereinander kommunizieren. Wichtig ist auch, dass die Daten nicht zentral gespeichert werden, sondern nur die

Verkehrsteilnehmer die Daten bekommen. Dafür braucht es ein intelligentes autonomes System, welches mit der Umwelt vernetzt ist und auf die individuellen Bedürfnisse der Individuen eingeht.

Die Trusted Web 4.0 kann auch bei Smart Home angewendet werden. Statt die Daten zentral zu speichern werden die Daten dezentral gespeichert. Das Gebäude optimiert sich autonom selbst und muss den Versorger nicht ständig über die aktuelle Situation benachrichtigen. Angriffe von außen sind nicht machbar, da die Geräte keinen Zugriff von außen ermöglichen. Das Ziel ist eine dezentrale Energieversorgung. Die Energiekonzerne müssen in der Zukunft darauf umrüsten. Die Automobilindustrie geht diesen Schritt bereits jetzt schon. Ein Homebot verknüpft mit der PDS kann beim Energiekonzern, die benötigte Menge an Energie nachfragen und speichert die Daten beim Kunden dann vor Ort ab. So besitzt nur der Kunde seine Daten. Damit der Homebot eine Kommunikation zum Energiekonzern aufbauen kann, muss dieser erst den Kunden fragen, ob er das darf. Das führt zu einer besseren Sicherheit des Systems.

Das E-Government kann dezentralisiert und anonym mit der PDS durchgeführt werden. Die Daten werden derzeit zentral an einer Stelle gespeichert. Angreifer und Geheimdienste können sich diese Daten beschaffen, wenn sie diese zentralen Stellen angreifen. Wenn die Kommunen die Daten dezentralisiert speichern, dann ist es für Angreifer schwieriger an die Daten zukommen. Auch die Public-Key-Infrastruktur beruht auf einer zentralen Stelle, dass können sich Angreifer zunutze machen. Hier müsste man den Schlüssel auch dezentralisiert speichern. Die Schlüssel werden beim Personalausweis verwendet und die zentrale Stelle ist das BSI.

Die Industrie 4.0 kann die dezentralisierende Globalisierung einführen, also weg von der zentralisierenden Globalisierung. Die Industrie 4.0 ermöglicht individuelle Kundenanfertigungen. Durch Dezentralisierung können Produktionen wieder in Deutschland stattfinden. Man würde näher an die Geschäfte produzieren und auch näher an den Endkunden sein. Dafür müssen widerspruchsfreie und offene Standards her. Über diese Standards kann der Informationsaustausch stattfinden. Ein Standard wäre das Internet nach Kategorien und Regionen aufzuteilen, also das Internet zu dezentralisieren. Die Latenzzeiten wären dann dadurch bei der Kommunikation sehr gering. Die Industrie 4.0 muss aus intelligenten autonomen Systemen bestehen, die dezentralisiert sind. Die Kommunikation muss nach menschlichen Regeln funktionieren. Man kann eine anonymisierte Kommunikation über eine eindeutige Auftragsnummer machen. Diese Auftragsnummer ist dezentral und weltweit nur einmal verfügbar. Über diese Auftragsnummer können alle weiteren Aktionen eines Bestellvorgangs abgewickelt werden.

Der Logistiker 4.0 kann die Aufträge bündeln und dadurch den Einzelhändler ersetzen. Er kann eine Preispolitik anbieten abhängig von den Lieferzeiten. Wenn jemand seine Ware schnell will, dann muss er zusätzlich drauf zahlen. Wenn jemand bereit ist für seine Ware

länger zuwarten, dann zahlt er für seine Ware weniger. Der Logistiker kann für mehrere Branchen ausliefern, wenn die Logistikanforderungen für die Branchen gleich sind.

Bei der Dezentralisierung des Finanzwesens ist die Projektfinanzierung sehr wichtig. Bis jetzt investieren Investoren in junge und große Firmen. Die Firmen haben meist ein skalierbares Geschäftsmodell. Es gibt aber auch neu Ansätze wie das Crowdfunding. Beim Crowdfunding investieren viele Kleinanleger für ein Projektziel. Es gibt dezentralisierte Währungen wie Bitcoin, diese beruht auf Blockchaining. Wenn man das Blockchaining mit dem PDS kombiniert, dann können Transaktionen anonym durchgeführt werden.

In der Zukunft sind für die meisten Unternehmen Datensicherheit durch Dezentralisierung und die digitale Transformation wichtige Punkte. Das Ziel muss es sein eine dezentralisierte und anonymisierte IT zu etablieren. Dadurch sind Angriffe von außen für Angreifer unwichtig, da der Aufwand sehr groß ist. Der Zeit nutzt man für die Sicherheit Überwachungssysteme. Der Schaden für Cyberkriminalität ist sehr hoch und kann Unternehmen in die Insolvenz führen. Für die Unternehmen gibt es 3 verschiedene Wege. Der erste Weg ist für die Unternehmen die mit digitaler Transformation nichts zu tun haben wollen. Sie verschlafen die Entwicklung und merken durch Cyberangriffe von Seiten der Konkurrenz, dass sie den Trend verschlafen haben. Irgendwann durch massiven Stellenabbau und Verlusten geht das Unternehmen dann in die Insolvenz. Den anderen Weg gehen Unternehmen, die mit der digitalen Transformation wachsen. Sie kaufen zusätzliches Know-how am Markt ein. Auch hier kommen irgendwann Cyberangriffe und dann steht das Unternehmen genauso dar wie beim ersten Weg. Den letzten Weg gehen Unternehmen, den Weg der Dezentralisierung und der Anonymisierung. Das Unternehmen wächst hier langsamer als beim zweiten Weg, man ist aber gegen Cyberangriffe von außen sicher.

Für ein sicheres Internet müssen die Politik und die Wirtschaft handeln. Die IT-Sicherheitsbranche ist noch eng mit den Geheimdiensten vernetzt. Um ein sicheres Internet zu schaffen, müssen diese voneinander abgekapselt werden. Derzeit gibt es immer noch zentrale Portale, die massiv überwacht werden, damit sie vor Angriffen sicher sind. Derzeit setzt sich die GISAD für ein sicheres Internet ein. Für die Einführung einer PDS braucht es noch Zeit, technisch ist es aber möglich. Wichtig ist, dass die personenbezogenen Daten beim Nutzer bleiben. Der Nutzer muss anonymisiert im Netz unterwegs sein und muss sich mit seinem PDS als einzige Identifikations- und Authentifikationsmethode anmelden. Der Kopierschutz muss kopieren im Internet verhindern und der Zugriff auf Dateien muss erst gestattet sein. Smartphones Apps dürfen keinen Zugriff mehr auf personenbezogenen Daten bekommen. Die Hardwarefunktionen müssen vom Kern getrennt werden, damit Angreifer diese nicht mehr nutzen können. Es müssen dezentralisierte symmetrische Schlüssel erzeugt und gespeichert werden. Davon werden drei gleiche Schlüssel generiert

und an verschiedene Punkte gespeichert. Dezentralisierung kann auch in der Cloud benutzt werden, aber personenbezogenen Daten und Schlüssel haben in der Cloud nichts verloren. Diese Daten müssen bei dem Nutzer selber bleiben.

Die Gemeinwohl-Ökonomie

Zusammenfassung des Buches:

Titel: Die Gemeinwohl-Ökonomie - Eine demokratische Alternative wächst

Verfasser: Christian Felber

Verlag: Deuticke

Jahr: 2014

ISBN: 978-3-552-06291-7

Zusammenfassung von: Nils Kohlmeier, Tolga Aydemir, Yannick Kloss

Soft Skills

Zusammenfassung des Buches:

Titel: Soft Skills - The software developer's life manual

Verfasser: John Z. Sonmez

Verlag: Manning

Jahr: 2014

ISBN: 978-1617292392

Zusammenfassung von: Gamze Soylev Oektem, Justin Jagieniak, Marvin Schirrmacher,
Daniel Beneker, Tim Jastrzembski

Erfolgsfaktoren für eine digitale Zukunft

Zusammenfassung des Buches:

Titel: Erfolgsfaktoren für eine digitale Zukunft - IT-Management in Zeiten der Digitalsierung und Industrie 4.0

Verfasser: Egmont Foth

Verlag: Springer Vieweg

Jahr: 2017

ISBN: 978-3-662-53176-1, 978-3-662-53177-8 (eBook)

Zusammenfassung von: Malte Berg, Jonas Wiese, Niklas Harting

Die Digitalisierung der Welt

Zusammenfassung des Buches:

Titel: Die Digitalisierung der Welt - Wie das Industrielle Internet der Dinge aus Produkten Services macht

Verfasser: Peter Samulat

Verlag: Springer Gabler

Jahr: 2017

ISBN: 978-3-658-15510-0, 978-3-658-15511-7 (eBook)

Zusammenfassung von: Nils Kohlmeier, Tolga Aydemir, Yannick Kloss

Was treibt die Digitalisierung?

Zusammenfassung des Buches:

Titel: Was treibt die Digitalisierung? - Warum an der Cloud kein Weg vorbeiführt

Verfasser: Ferri Abolhassan, T-Systems International GmbH (Herausgeber) **Verlag:**

Springer Gabler **Jahr:** 2016 **ISBN:** 978-3-658-10640-9

Zusammenfassung von: Lukas Taake, Philip Viertel, Andre Kaleja

Trusted Web 4.0 - Konzepte einer digitalen Gesellschaft

Zusammenfassung des Buches:

Titel: Trusted Web 4.0 - Konzepte einer digitalen Gesellschaft - Konzepte der Dezentralisierung und Anonymisierung

Verfasser: Olaf Berberich

Verlag: Springer-Verlag

Jahr: 2016

ISBN: 978-3-662-49189-8

Zusammenfassung von: Benjamin Schmidt

Zusammenfassung: Cloud

Zusammenfassung der Bücher:

Titel: Cloud Computing als neue Herausforderung für Management und IT

Verfasser: Martin Reti, Michael Pauly, Gerald Münzl

Verlag: Springer

Jahr: 2015

ISBN: 9783662458310

Titel: Cloud Computing Basics

Verfasser: S. Srinivasan

Verlag: Springer

Jahr: 2014

ISBN: 978-1-4614-7698-6, 978-1-4614-7699-3(eBook)

Titel: Cloud Computing Web-basierte dynamische IT-Services

Verfasser: Christian Baun, Marcel Kunze, Jens Nimis, Stefan Tai

Verlag: Springer

Jahr: 2011

ISBN: 978-3-642-18435-2, 978-3-642-18436-9(eBook)

Titel: Requirements Engineering for Service and Cloud Computing

Verfasser: Muthu Ramachandran, Zaigham Mahmood

Verlag: Springer

Jahr: 2017

ISBN: 978-3-319-51309-6, 978-3-319-51310-2(eBook)

Zusammenfassung von: Alexander Schwietert, Timo Rolfsmeier

Einführung

Definition

Zum jetzigen Stand gibt es keine eine einheitliche Definition des Cloud-Begriffes, noch gibt es standardisierte Gütesiegel, welches einen Rahmen für die erforderlichen Eigenschaften eines Cloud Services schaffen könnte. Der grundlegende Gedanke einer Cloud ist es, dass Ressourcen nicht mehr besessen werden, sondern temporär in Form eines Services über ein Kommunikationsnetz genutzt werden.

Viele Anbieter schreiben Cloud jedoch lediglich im Zuge des Hypes, als Marketinginstrument, in ihr Portfolio. Das klassische Outsourcing ist per se jedoch nicht mit einem Clouddienstleister gleichzusetzen.

BITKOM bietet folgende Definition an.

Cloud Computing ist eine Form der Bereitstellung von gemeinsam nutzbaren und flexibel skalierbaren IT-Leistungen durch nicht fest zugeordnete IT-Ressourcen über Netze. Idealtypische Merkmale sind die Bereitstellung in Echtzeit als Self Service auf Basis von Internet-Technologien und die Abrechnung nach Nutzung.

Eigenschaften einer Cloud

Geschäftsbezogene Merkmale	Technikbezogene Merkmale
Bereitstellung und Nutzung von IT Ressourcen als Service	Flexible Bereitstellung skalierbarer IT Ressourcen
Schnelle und flexible Ressourcen-Verfügbarkeit / On-Demand	Mandantenfähige, gemeinsam nutzbare Infrastruktur (z.B. eine Installation einer Software die mehrere Kunden mit eigenen Accounts nutzen können)
Ressourcen-Zuordnung durch den Kunden Steuerbar (Self-Service)	Hohe Automatisierung / Standardisierung des Angebots
Kurze Vertragsbindung	Logisch zentralisierte, virtualisierte IT Infrastruktur
Keine oder nur minimale Vorabinvestition für den Kunden	Zugriff via Browser oder Apps
Variable Kosten nach Nutzung der Ressourcen	Vollständige, lastabhängige Skalierbarkeit
	Messbarkeit des IT-Verbrauchs

Chancen und Risiken

Die Chancen

Grade am Konsumentenmarkt gibt es bereits viele bekannte Beispiele, wo skalierbare Ressourcen aus der Cloud eine große Rolle spielen. Unternehmen wie Amazon, Facebook oder Google können dadurch dynamisch auf die wechselnde Auslastung ihrer Infrastruktur reagieren. Auch Startups, sowie kleinere und mittelständische Unternehmen können von der Cloud stark profitieren, da sie das Investitionsrisiko für neue Unternehmungen senkt und initiale Kosten einer IT Infrastruktur wegfallen. Folgende Punkte konkretisieren die potenziellen Vorteile der Cloudnutzung.

- Durch Auslagerung in IT Services können Unternehmen sich auf ihr Kerngeschäft konzentrieren und erreichen eine geringere Fertigungstiefe in der Produktion. So kann die Qualität der Kernprozesse sowie das Wachstum gesteigert werden.
- Statt fixer Kosten werden, transparent für den Nutzer von Cloud Services, so viele Ressourcen zur Verfügung gestellt wie benötigt und entsprechend abgerechnet. Dadurch sinkt das gebundene Kapital, sowie initiale Investitionskosten die für den Aufbau einer eigenen IT Infrastruktur nötig wären.
- Risiken bezüglich Personal sowie technischen und Sicherheitsaspekten werden auf den Dienstleister übertragen.
- Die Expertise des Dienstleisters wäre nur mit erheblich höheren Eigenaufwand zu

erreichen

- Standardisierte Cloudangebote beschleunigen bereits heute die Einführung neuer Geschäftsmodelle und Produkte, da neue Modelle schnell und flexibel eingeführt und bei einem Fehlschlag auch schnell wieder vom Markt genommen werden können (kürzere Time To Market)
- Im Bezug auf die Green IT resultiert eine, nur nach Bedarf genutzte, Infrastruktur in geringeren Stromverbrauch

Die Risiken

Unternehmen haben auf der anderen Seite auch hohe Ansprüche, die die Cloud erfüllen muss. Ein Dienst muss flexibel, sicher, skalierbar, zuverlässig, hochverfügbar, kostengünstig, effizient und transparent nutzbar sein.

Das Risiko beginnt bereits mit der korrekten Einführung von Cloud Services, für die ein Unternehmen eine einheitliche Strategie braucht, damit keine Insellösung für einzelne Fachbereiche oder IT-Abteilungen entstehen.

Risiken bei unkontrollierter ungesteuerte Nutzung von Cloud-Diensten sind Folgende.

- Verlust von Unternehmensgeheimnissen oder Missbrauch von vertraulichen Kundendaten
- Compliance-Verstöße, die strafrechtlichen Konsequenzen nach sich ziehen. Können finanziell schädigen und den Ruf des Unternehmens beeinträchtigen
- Etablierung von verschiedenen Standards in verschiedenen Unternehmensbereichen kann große Integrationsaufwände in der IT-Landschaft eines Unternehmens verursachen
- Unkontrollierter Einsatz von Cloud kann Effizienzprojekte zur Konsolidierung von Hardware und Applikationen untergraben

Ein Unternehmen muss sich daher genau überlegen welche Rolle Cloud Computing im Unternehmen spielen soll, welche Daten und Prozesse sich auslagern lassen und welche Anbieter für die eigenen Ansprüche in Frage kommen.

Auch für zukünftige Entwicklungen, wie beispielsweise die Frage, ob eigenen Applikationen zukünftig direkt cloud-ready sein sollen, müssen frühzeitig betrachtet werden und in die Strategie des Unternehmens einfließen.

Integration & Strategie im Unternehmen

Auch wenn die IT-Abteilung eine tragende Rolle bei der Integration von Cloud Services in einem Unternehmen einnimmt, so sollten sie nicht alleine über die Cloud Strategie entscheiden. Diese muss auf die Unternehmensstrategie abgestimmt sein und die

entsprechenden Geschäftsziele abbilden, wie beispielsweise Umsatz, Kundenzufriedenheit oder Produktentwicklungszeiten.

Nachdem die oberste Ebene, in Form der Geschäftsziele, abgesteckt wurde, kann auf die einzelnen Geschäftsprozesse geschaut werden. Welche Prozesse können von der Cloud profitieren, können automatisiert werden, müssen dynamischer und flexibler werden? Auch müssen potenziell notwendige, neue Prozesse definiert werden.

Die festgelegten Anforderungen für die Geschäftsziele- und Prozesse müssen anschließend zu einer IT-Strategie übersetzt werden. Je nach Geschäftsmodell, Zielen und Ausrichtung des Unternehmens wird dann der Rahmen für die IT gesetzt, in welchem Ausmaß die Cloud genutzt werden soll.

Der Prozess



Abbildung 1: Prozess der Integration in die Cloud

(Quelle: *Cloud Computing als neue Herausforderung für Management und IT* von Münzl et al. 2015, S. 24)

Im ersten Schritt des Integrationsprozesses muss der Ist-Zustand der Geschäftsprozesse erfasst werden, dies geschieht in Zusammenarbeit mit den jeweiligen Abteilungen. Die Änderungsanforderungen werden vom Vergleich des Ist-Zustands mit dem Zielbild abgeleitet. Dabei wird direkt überprüft, ob die geplanten Änderungen mit Compliance und Sicherheitsvorschriften vom Unternehmen einhergehen, sowie staatliche Vorgaben eingehalten werden.

Die genaue Ausgestaltung hängt dabei vom Marktumfeld, den Kundenerwartungen, dem Geschäftsmodell und der individuellen Prozesslandkarte eines Unternehmens ab und führen entsprechend zu einer individuellen Ableitung der Strategie.

Migration

Es empfiehlt sich für ein Unternehmen eine Roadmap mit kurzfristigen, mittelfristigen und langfristigen Entwicklungszielen - und Maßnahmen anzulegen. Anhand dieser Map können Cloud-Provider gesucht werden, die die qualitativen und quantitativen Ansprüchen des Unternehmens erfüllen. Der darauffolgende Umstieg kann entweder für alle Benutzer gleichzeitig (Big Bang) oder schrittweise erfolgen. Optional wird vorher ein Pilotbetrieb eingerichtet.

Als Einstieg werden in der Regel gerne IaaS-Lösungen verwendet, da diese keine großen Anpassungen der Geschäftsprozesse mit sich bringen.

IT-Abteilungen im Wandel

Durch die Auslagerung von Kompetenzen und Ressourcen wird der IT-Manager zukünftig eine businessorientiertere Rolle einnehmen. Er ist für das Monitoring der Leistungen für Geschäftsprozesse und deren genutzte Services zuständig. Er muss als Bindeglied die Zielvorgaben und Anforderungen des Unternehmens unter den Providern delegieren. Ein weiteres Stichwort ist hier IT-Governance, wo Aspekte wie die Informationssicherheit und Compliance-Vorgaben mit den Providern abgestimmt werden müssen. Der CIO ist somit für die Gesamtorchestrierung sowie die Funktionsfähigkeit der Gesamtlandschaft aller IT-Systeme zuständig.

Die Rolle des klassischen Mitarbeiters in der IT wandelt sich vom Administrator über den Servicemanager hin zum Service Broker. Klassische Aufgaben wie Installation und Wartung fallen weg, neue Aufgaben wie das Managen und Überwachen von Cloud Services kommt hinzu.

Geschichte & Evolution

Das erste Mal wurde der Cloudbegriff in einem Businessplan der Firma compaq aus dem Jahr 1996 erwähnt, in dessen Zeitraum die Verbreitung stark zunahm. Eric Schmidt, damaliger CEO von Google, gilt als die Person die den Begriff des Cloud Computing auf einer Konferenz im Jahr 2006 populär gemacht hat. Nach einer anfänglichen Skepsis, ob der erste Hype um die Cloud wirklich gerechtfertigt war und die Unternehmen nicht nur Buzzwords hinterherlaufen, zeigte sich besonders in den letzten Jahren eine enorme Investitionssteigerung. Alleine in Deutschland wurden aus 7 Mrd. im Jahr 2014 bereits 3 Jahre später bis zu 18,5 Mrd € investiertes Kapital.

Die Cloud ist keine revolutionäre Technologie, wie es einst das Telefon, der Fernseher oder das Internet war. Diese brauchten viele Jahre bis Jahrzehnte um eine weite Verbreitung zu erreichen, doch veränderten den Alltag der Menschen in einem großen Ausmaß.

Die Cloud hingegen optimiert bereits bestehende Technologien, nutzt die bereits vorhandene Infrastruktur in Form von weit verbreiteten, schnellen Internetverbindungen und die Leistungen der großen Rechenzentren, was in einer deutlich schnelleren Verbreitung der Cloud Technologie resultiert.

Die Cloud ist die logische Fortführung des Time-Sharing Modells aus den 1970er Jahren, welches der erste konzeptionelle Ansatz für ein Mehrbenutzersystem war, wo mehrere User sich die Rechenzeit eines Prozessors teilten (siehe Abbildung 2: Evolution der Cloudtechnologie).

Die Cloud ist die logische Fortführung des Time-Sharing Modells aus den 1970er Jahren, welches der erste konzeptionelle Ansatz für ein Mehrbenutzersystem war, wo mehrere User sich die Rechenzeit eines Prozessors teilten (siehe Abbildung 2: Evolution der Cloudtechnologie).

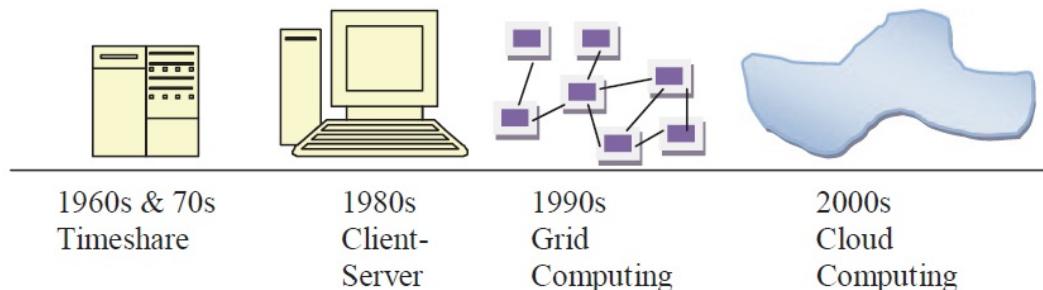


Abbildung 2: Evolution der Cloudtechnologie

(Quelle: *Cloud Computing Basics* von Sriinivasan 2014, S. 4)

Ein erster großer Schritt in der Entwicklung der Cloud wurde 1999 mit Salesforce.com getan. Die Firma schaffte ein Konzept wie man Enterprise-Anwendungen über eine einfach zu nutzende Webseite zur Verfügung zu stellen konnte.

Im Jahr 2002 kam mit den Amazon Web Services der nächste Schritt, indem der Konzern Rechenleistung und Speicher für seine Kunden anbot. Später startete Amazon zusätzlich die Elastic Compute Cloud, mit der Kunden Computer mieten konnte, um auf diesen ihre eigenen Applikationen laufen zu lassen.

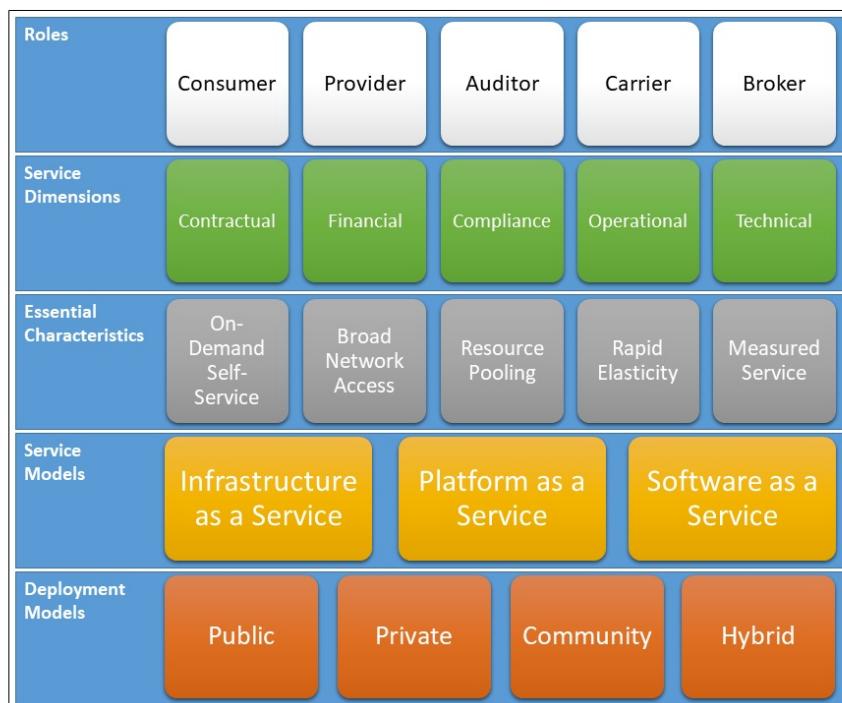
Mit der Verbreitung des Web 2.0 um das Jahr 2009 wurde ein weiterer Meilenstein erreicht. Google konnte mit seinen Apps auf neuem Wege eine Vielzahl von browserbasierten Enterprise-Applikationen anbieten und wirkte somit bei der Verbreitung des SaaS Konzeptes am Markt mit.

In den folgenden Jahren stiegen auch andere Größen wie Microsoft, IBM und Rackspace in das Cloudbusiness mit ein.

Cloud im Detail

Cloud Computing ist heutzutage ein viel benutzter Begriff, der sowohl in den Medien als auch im Unternehmensumfeld reges Interesse findet. Insbesondere die Publizität des Cloud-Ansatzes sorgt jedoch dafür, dass oft Unklarheit bezüglich seiner Bedeutung herrscht. Zwar erkennt die Mehrheit, dass Cloud Computing mit einer Auslagerung von Informationen oder Diensten gleichzusetzen ist. Eine weitere Spezifizierung ist aber nur den Wenigsten möglich.

Ziel dieses Kapitels ist es, einen einheitlichen Cloud-Begriff zu vermitteln. Hierzu wird im weiteren auf das Cloud-Computing Definition Framework eingegangen, welches vom National Institute of Standards and Technology (NIST) entwickelt und von Zalazar, Ballejos und Rodriguez um weitere Elemente erweitert wurde (siehe Abb. 1). Das Modell vollzieht eine Abstraktion der Cloud in verschiedene Schichten, welche sich teilweise bedingen. Zum einen wird auf die Teilnehmer eines Cloud-Szenarios eingegangen. Eine weitere Schicht beschäftigt sich mit den vertraglichen Bedingungen, welche zwischen den jeweiligen Teilnehmern herrschen können. Außerdem erfolgt die Aufzählung der typischen Charakteristiken einer Cloud. Abschließend wird darauf eingegangen, welche Art von Ressourcen eine Cloud bereitstellen kann bzw. wie offen eine Cloud für andere Teilnehmer ist. Eine detailliertere Beschreibung der einzelnen Schichten geschieht im Folgenden.



(Abbildung 1: Erweitertes Cloud-Computing Definition Framework)

(Quelle: in Anlehnung an *Cloud Dimensions for Requirements Specification* von Zalazar et al. 2017, S. 26)

Rollen

Das Cloud-Computing Definition Framework beinhaltet eine einheitliche Rollenbeschreibung. Hiermit ist es möglich, Zuständigkeiten und Verantwortungen in einem Cloud-Szenario exakt abzugrenzen und vertraglich festzuhalten. Generell wird zwischen fünf verschiedenen Rollen unterschieden, wobei nicht jede Rolle zwangsläufig eingenommen werden muss.

- **Consumer:** Der Consumer bzw. Konsument nimmt eine zentrale Rolle in einem Cloud-Szenario ein. Er hat das Bedürfnis, einen Cloud-Dienst wahrzunehmen. Somit kann er als Aktor angesehen werden, von dem die initiale Nutzungsanfrage ausgeht. Der Konsument hat zudem bestimmte Anforderungen im Hinblick auf die Eigenschaften des benötigten Dienstes. Beispielsweise kann er ein bestimmtes Maß an Erreichbarkeit oder Performanz einfordern bzw. hat spezifische Preisvorstellungen.
- **Provider:** Der Provider bzw. Anbieter definiert das Gegenstück zum Konsumenten. Er bietet Cloud-Dienste an und steht in der Verantwortung, die vom Konsumenten geforderte Leistung zu erbringen. Der Anbieter kann dabei eine beliebige Anzahl an Konsumenten jeweils mit gleichen oder unterschiedlichen Diensten bedienen.
- **Auditor:** Der Auditor bzw. Prüfer ist eine unabhängige Instanz, welche Gutachten bezüglich der erbrachten Leistungen seitens des Anbieters erstellt. Sind diese Leistungen unterhalb der zwischen Konsument und Anbieter vertraglich festgelegten Mindestleistungen befindlich, erstattet der Prüfer Meldung an den Konsumenten. Anschließend kann dieser beispielsweise vom Anbieter Schadensersatz verlangen.
- **Carrier:** Der Carrier bzw. Bote regelt und vollzieht die Übermittlung von Daten zwischen dem Anbieter und Konsumenten. Hierunter ist in den meisten Fällen der Internet-Provider zu verstehen. Seine erbrachten Leistungen haben erheblichen Einfluss auf die Zufriedenheit der Konsumenten. Können Daten nicht zeitnah und vollständig ausgeliefert werden, gefährdet dies ihre Wirtschaftlichkeit.
- **Broker:** Der Broker bzw. Vermittler ist ein Intermediär, welcher als Schnittstelle zwischen Konsument und Anbieter agieren kann. Ist aus spezifischen Gründen ein direkter Austausch von Leistungen zu unterlassen, trägt er dafür Sorge, dass Dienste von Teilnehmern gefunden sowie von bzw. an diese überbracht werden können.

Schlüsselmerkmale

Es existieren diverse Arten von Cloud-Computing, welche im Lauf dieses Kapitels näher erläutert werden. Allen Arten liegen jedoch im Regelfall dieselben Merkmale zugrunde, welche somit die Grundlage einer jeden Cloud bilden. Die Schlüsselmerkmale werden im Folgenden dargelegt.

- **On-demand self-service:** Jeder Konsument kann zu jeder Zeit und ohne eine weitere menschliche Interaktion seitens des Anbieters einen Dienst wahrnehmen.
- **Broad network access:** Alle Dienste sind über das Internet und unabhängig von der Plattform des Endgeräts erreichbar.
- **Resource Pooling:** Der Anbieter legt seine gesamten Hard- und Softwareressourcen zusammen. Alle Konsumenten kommunizieren hierdurch mit demselben IT-System. Da dieses System mandantenfähig ist, arbeiten die einzelnen Nutzer jedoch logisch gesehen auf verschiedenen Systemen und haben keine Einsicht in die Bereiche anderer Konsumenten.
- **Rapid elasticity:** Der Anbieter kann seinen Konsumenten dynamisch Ressourcen zuteilen und wieder entziehen. Der Bedarf des Konsumenten kann also stetig variieren, ohne dass ihm jemals Ressourcen fehlen bzw. Ressourcen von ihm ungenutzt sind. Die Ressourcenzuteilung seitens des Anbieters geschieht dabei im Idealfall automatisch.
- **Measured Service:** Die Auslastung des Cloud-Services seitens des Konsumenten ist sowohl für Anbieter als auch Konsument jederzeit einsehbar. Hierzu zählen beispielsweise der genutzte Speicher, die Anzahl der übermittelten Daten oder die Anzahl der aktiven Benutzeraccounts. Durch dieses Vorgehen besitzt der Konsument sowohl Transparenz über sein Nutzungsverhalten als auch über die daraus resultierenden Kosten.

Dienstdimensionen

Ein typisches Cloud-Szenario umfasst grob betrachtet zwei Rollen. Ein Anbieter stellt einen Dienst zur Verfügung, den ein Konsument wiederum nutzt. In dieser Konstellation hat jede Partei Anforderungen an die andere Partei. Der Konsument möchte beispielsweise eine bestimmte Performanz der angebotenen Cloud-Services gewährleistet haben, während der Anbieter auf eine pünktliche und leistungsgerechte Bezahlung Wert legt. Die Gesamtheit der Anforderungen, welche die beiden Parteien jeweils an ihr Gegenüber stellen, lassen sich fünf verschiedenen Bereichen bzw. Dimensionen zuordnen.

- **Contractual:** Contractual bzw. vertragliche Anforderungen sind die grundlegenden Richtlinien, die der Dienst-Vertrag zwischen Anbieter und Konsument umfasst. Hierin wird festgelegt, welche Parteien in welcher Rolle an dem Geschäft beteiligt sind. Die Aufführung von Vermittlern, Prüfern und Boten sei an dieser Stelle möglich. Weiterhin ist definiert, wie lang der Vertrag gültig ist bzw. wann die Nutzung des Services seitens des Konsumenten startet und endet. Letztlich kann auch spezifiziert werden, wann die Löschung der Anbieterdaten nach Ablauf des Vertrages frühestens erfolgen darf bzw. spätestens erfolgen muss. Die vertraglichen Anforderungen sind meistens statisch und ändern sich während der Vertragslaufzeit demnach nur selten.
- **Financial:** Financial bzw. finanzielle Anforderungen werden hauptsächlich vom Anbieter

gestellt und dienen seiner gerechten Entlohnung. Zum einen ist hier das Modell zu spezifizieren, nach dem die Bezahlung erfolgen soll. Beim ***tiered pricing*** erfolgt die Abrechnung anhand der Zeit, in der der Konsument den Dienst tatsächlich genutzt hat. ***Per-unit pricing*** bedeutet, dass der Konsument einen festen Preis für jede genutzte Einheit zahlt. Eine Einheit sei beispielsweise ein Gigabyte an Speicherplatz, welches der Konsument auf Seiten des Anbieters belegt hat. Beim ***subscription-based pricing*** bezahlt der Konsument ähnlich dem ersten Modell für einen Zeitraum. Der Unterschied liegt jedoch darin, dass es nicht von Relevanz ist, ob bzw. wie oft er den Service in diesem Zeitraum tatsächlich wahrgenommen hat. Dieses Bezahlmodell kann heutzutage mit einer Flatrate verglichen werden. Neben dem Bezahlmodell umfassen die finanziellen Anforderungen zudem die Höhe der Bezahlung sowie den Bezahlrhythmus. Weiterhin ist festgelegt, mit welcher Methode die Zahlung erfolgen muss, beispielsweise per Pay-Pal oder normaler Überweisung.

- **Compliance:** Compliance-Anforderungen betreffen die Einhaltung von Vorgaben. Insbesondere der Konsument muss sicherstellen, dass seine ausgelagerten Dienste bestimmte Gesetze, Normen und Standards einhalten. Andernfalls kann dies für ihn rechtliche bzw. wirtschaftliche Konsequenzen nach sich ziehen. Beispielhaft sind hierfür rechtliche Vorgaben zur Lagerung sensibler Daten bzw. strikte Zugangskontrollen zu bestimmten Ressourcen. Neben den eigentlichen Vorgaben beinhaltet diese Dimension zudem Strafen bzw. Entschädigungen, die bei Nichteinhaltung von vertraglichen Bestimmungen an den Anbieter verhängt werden können.
- **Operational:** Operational bzw. betriebliche Anforderungen sollen die Erreichbarkeit eines Cloud-Dienstes sicherstellen. Es wird initial festgelegt, wie lange der Dienst pro Zeiteinheit maximal ausfallen darf. Des Weiteren sind Maßnahmen zur Instandhaltung und Wiederherstellung der Systeme angegeben. Auch eine geeignete Bewertung bzw. Behandlung von Ausfällen wird vereinbart, um den bestmöglichen Betrieb zu gewährleisten. Letztlich sind geeignete Skalierungsvorgaben auszumachen, da die Bereitstellung von Ressourcen seitens des Anbieters ab einem bestimmten Punkt limitiert ist.
- **Technical:** Technical bzw. technische Anforderungen beziehen sich maßgeblich auf die Leistung des Dienstes. Der Konsument will sicherstellen, dass seine Bedürfnisse zu jeder Zeit in ausreichendem Maß befriedigt werden. Hierfür werden diverse Kennzahlen festgelegt, welche seitens des Anbieters zu erfüllen sind. Beispielhaft hierfür sind die Performanz und die Speicherkapazität seines IT-Systems sowie der Datendurchsatz des Netzwerks. Abhängig vom tatsächlichen Dienst sind weiterhin Schnittstellen oder vorinstallierte Betriebssysteme sowie Softwarebibliotheken aufzuführen, welche das IT-System umfassen muss.

Dienstmodelle

Der bisherige Dienst-Begriff ist allgemein formuliert und umfasst alle Arten von Leistungserbringungen. Das National Institute of Standards and Technology vollzieht hier eine weitere Spezifizierung, um verschiedenartige Dienste effektiver voneinander abgrenzen zu können. Im Folgenden werden die drei verschiedenen Dienstmodelle erläutert.

- **Infrastructure as a Service (IaaS):** Das IaaS-Modell zielt auf die Bereitstellung gesamter Rechnerinstanzen ab. Hierunter sind sowohl Server als auch Datenbanken bzw. Speichermedien im Allgemeinen zu verstehen. Der Konsument hat grundlegenden Zugriff auf die Hardware-Komponenten, die ihm der Anbieter bereitstellt. Es ist ihm somit möglich, seine eigenen Konfigurationen vorzunehmen sowie eigene Anwendungen aufzuspielen. Die Aufgabe seitens des Anbieters besteht lediglich darin, einen störungsfreien Betrieb der Hardware sicherzustellen. Hierzu zählt auch die kontinuierliche Überwachung der Performanz, Speicherbelegung sowie Datentransferrate. Das Dienstmodell bietet sich für solche Konsumenten an, die bereits über stabil laufende Anwendungen verfügen und lediglich eine flexible Skalierung der Rechenleistung benötigen.
- **Platform as a Service (PaaS):** Das PaaS-Modell baut auf dem des IaaS auf. Zusätzlich zu den Rechnerinstanzen stellt der Anbieter hier noch komplett Systemumgebungen bereit. Je nach Wunsch des Konsumenten sind auf den Hardware-Knoten bestimmte Betriebssysteme, Entwicklungsumgebungen sowie Programmbibliotheken installiert. In manchen Fällen können zudem grundlegende Prozeduren bezüglich Zugriffskontrolle, Synchronisierung oder Datenhaltung vorgefunden werden. Die bereits vorkonfigurierten Container erleichtern den Konsumenten die Software-Entwicklung, da keine zusätzlichen Anstrengungen für die Konfiguration einer Systemumgebung nötig sind.
- **Software as a Service (SaaS):** Beim SaaS-Modell gestattet der Anbieter den Konsumenten die Nutzung einer speziellen Software, welche auf seinem IT-System arbeitet. Der Konsument hat von jedem Endgerät, zu jeder Zeit und an jedem Ort Zugriff auf die Applikation. Er besitzt jedoch keinen Einfluss auf die darunterliegenden Hardware-Komponenten sowie deren Konfiguration. Dieses Recht bleibt dem Anbieter vor, der für den reibungsfreien Betrieb der Software verantwortlich ist. Die Hauptvorteile des Dienstmodells liegen in den geringeren Investitionskosten sowie in der schnelleren Einsatzfähigkeit der Software.

Deployment-Modelle

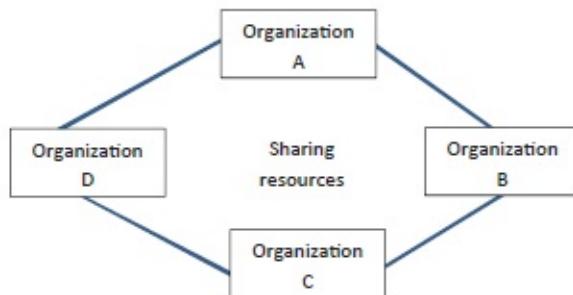
Innerhalb des Cloud-Computing existieren vier unterschiedliche Deployment-Modelle. Diese definieren jeweils, welche Zugriffsmöglichkeiten auf die Cloud-Dienste herrschen.

- **Private cloud:** Die Dienste einer privaten Cloud sind auf einzelne Organisationen bzw. Individuen beschränkt. Typischerweise hat ein Unternehmen Zugriff auf seine eigene

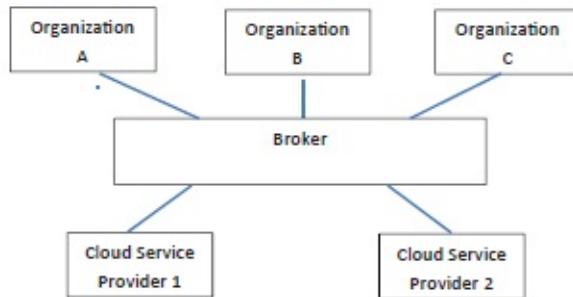
private Cloud, welche ausschließlich ihm zur Verfügung steht. Im Fall einer **typical private cloud** hostet das Unternehmen die Cloud in ihrem eigenen Rechenzentrum und ermöglicht den Zugriff per Internet. Eine Firewall trägt dafür Sorge, dass nur unternehmensinterne Mitarbeiter Zugang zu den Applikationen bzw. Informationen haben. Die **managed private cloud** ähnelt dem Vorgänger, mit Ausnahme dass das Rechenzentrum des Unternehmens von einem externen Anbieter verwaltet wird. Dies hat jedoch keinen Einfluss auf die Lokalität der Hardware. Bei einer **hosted private cloud** laufen die Cloud-Anwendungen direkt auf den Hardware-Komponenten des externen Anbieters. Weiterhin hat ausschließlich das spezifische Unternehmen Zugriff. Die letzte Variation ist die **virtual private cloud**, welche im eigentlichen Sinn eine öffentliche Cloud ist, auf die mehrere Unternehmen Zugriff haben. Durch VPN wird jedoch in dieser öffentlichen Cloud eine virtuelle private Cloud erzeugt, auf der ausschließlich das Unternehmen arbeiten kann.

- **Public Cloud:** Die öffentliche Cloud ist das Gegenstück zur privaten Cloud. Sie erlaubt verschiedenen Konsumenten den Zugriff auf ihre Funktionen. Dem Anbieter gehören dabei sämtliche Anwendungen, welche öffentlich zur Verfügung stehen. Typischerweise verfügt dieses Deployment-Modell über einfache Anmeldestrukturen und standardisierte Services, um möglichst vielen Konsumenten gerecht zu werden. Die öffentliche Cloud ist für viele Benutzer der Inbegriff von Cloud Computing.
- **Community Cloud:** In einer Community Cloud schließen sich verschiedene Organisationen zusammen, welche ein gemeinsames Ziel verfolgen. Beispielsweise sind verschiedene Krankenhäuser, die dieselben medizinischen Geräte verwenden oder ihre Erkenntnisse untereinander teilen wollen. Demnach ist die Community Cloud eine öffentliche Cloud für eine homogene Gruppe von Konsumenten. Bei einer **federated community cloud** greifen die Organisationen innerhalb der Gruppe gegenseitig auf ihre eigenen Ressourcen zu (siehe Abb. 2). Bei einer **brokered community cloud** erfolgt der Zugriff über einen Vermittler, welcher die Anfragen an einen zentralen Anbieter weiterleitet.

Federated community cloud:



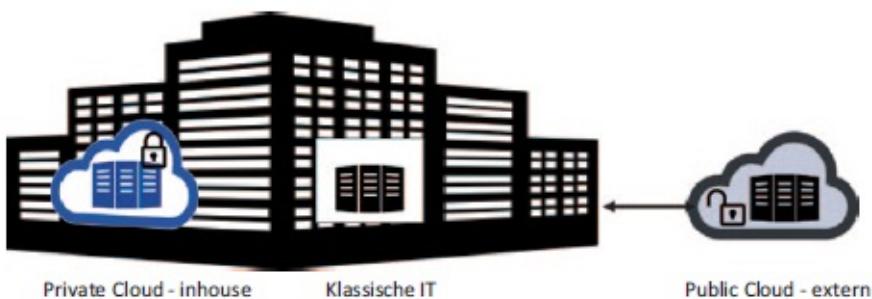
Brokered community cloud:



(Abbildung 1: Vergleich von Federated und Brokered Community Cloud)

(Quelle: *Cloud Computing Basics* von Srinivasan 2014, S. 36)

- **Hybrid Cloud:** Eine hybride Cloud ist der Zusammenschluss von mindestens zwei der vorangegangenen Deployment-Modelle. Beispielsweise kann ein Unternehmen sensible Daten in eine private Cloud auslagern, während rechenintensive und unsensible Dienste in einer öffentlichen Cloud ablaufen (siehe Abb. 3).



(Abbildung 3: Beispiel für hybride Cloud)

(Quelle: *Cloud Computing als neue Herausforderung für Management und IT* von Münzl et al. 2015, S. 13)

Das Cloud-Computing Definition Framework trägt zur Präzision und Vereinheitlichung des Cloud-Begriffs bei. Insbesondere die Kommunikation sowie Entwicklung im Cloud-Umfeld wird hierdurch erleichtert.

Sicherheit

Sicherheit ist für ein effektives Cloud Computing von existenzieller Bedeutung. Nur wenn Risiken minimiert bzw. vollkommen vermieden werden können, ist es Organisationen möglich, Potenziale im Hinblick auf Wirtschaftlichkeit und Flexibilität auszuschöpfen. Der allgemeine Begriff Sicherheit lässt sich in puncto Cloud in zwei Unterkategorien unterteilen, welche im Folgenden erläutert werden.

Compliance

Nahezu jedes Unternehmen unterliegt externen Vorgaben, welche als Compliance bezeichnet werden. Hierunter fallen zum einen staatliche Vorgaben, beispielsweise bezüglich des Datenschutzes. Zum anderen existieren brachenspezifische bzw. - unabhängige Normen und Standards, welche zur Aufrechterhaltung des Geschäfts einzuhalten sind. Mit jeder Hard- bzw. Software-Komponente, die der Konsument zu einem Anbieter auslagert, erhöht sich das Risiko, dass relevante Vorgaben fortan nicht mehr eingehalten werden. Zwangsläufig muss der Konsument sicherstellen, dass der Anbieter ohne jede Ausnahme nach seinen vorgegebenen Maßstäben arbeitet. Dieser Prozess beginnt im Idealfall schon bei der Anbieterauswahl.

Das wohl wichtigste Sicherheitsthema im Hinblick auf die Cloud ist der Schutz von sensiblen Unternehmensdaten. Hierunter sind zum einen Personendaten zu verstehen, wie beispielsweise Namen, Geburtsdaten und Adressen von Mitarbeitern sowie Kunden. Des Weiteren sind auch Finanzdaten hinzuzuzählen, welche den Fiskus betreffen. Bevor ein Konsument diese Daten in ein entferntes Cloud-Rechenzentrum auslagert, muss er sich zunächst über die geltenden rechtlichen Gegebenheiten informieren. In Abhängigkeit davon, in welchem Land der Konsument ansässig ist und in welchem Land sich das entfernte Rechenzentrum befindet, kann eine Auslagerung von sensiblen Unternehmensdaten von vornherein untersagt werden. Ist die Auslagerung rechtens, muss sich der Konsument sicher sein, dass der Anbieter die ihm anvertrauten Daten gewissenhaft und vertraulich behandelt. In Deutschland kann zu diesem Zweck beispielsweise eine Auftragsdatenverarbeitung beschlossen werden. Dieses Schriftstück legt fest, dass der Anbieter die sensiblen Daten nicht für seine eigenen Zwecke benutzt bzw. der Konsument das volle Recht auf die Daten beibehält.

IT-Sicherheit

Die IT-Sicherheit beschäftigt sich mit den technischen Maßnahmen, die für einen sicheren Cloud-Betrieb vorgenommen werden müssen. Schutzziele sind beispielsweise Vertraulichkeit, Integrität oder Authentizität. Eine sichere Nutzung lässt sich dabei nur gewährleisten, wenn Konsument, Anbieter und die zwischen ihnen herrschende Verbindung ganzheitlich geschützt sind.

- **Sicherheit beim Anbieter:** Der Anbieter von Cloud-Diensten steht in der Verantwortung, seine Rechenzentren gegenüber Missbrauch zu schützen. Er verfügt in vielen Fällen über sensiblen Daten diverser Organisationen bzw. Individuen und muss dafür Sorge tragen, diese vor Dritten geheim zu halten. Dabei kann er auf eine Vielzahl von Techniken zurückgreifen, die im Allgemeinen dem Schutz von Rechenzentren dienen und nicht spezifisch auf die Cloud-Architektur ausgelegt sind. Beispieldhaft sei hierfür eine sichere Authentifizierung der Konsumenten nach dem Zweifaktor-Prinzip. Weiterhin muss er Vorkehrungen treffen, welche im Detail an das Cloud-Szenario angepasst sind. Beispielsweise ist die logische Trennung von mehreren Konsumenten (Resource Pooling) abzusichern, sodass keine Zugriffe auf fremde Ressourcen möglich sind. Auch die Anwendung von speziellen Cloud-Patterns kann die Sicherheit verbessern.
- **Sicherheit beim Konsumenten:** Obwohl oft davon ausgegangen wird, dass die Verantwortung für einen sicheren Cloud-Betrieb allein beim Anbieter liegt, steht auch der Konsument in der Pflicht. Wie groß sein Beitrag zur Sicherheit dabei ausfallen kann, hängt vom Dienstmodell ab. IaaS erlaubt dem Konsumenten die ausführliche Absicherung seiner genutzten Rechenknoten. Bei SaaS fallen die Möglichkeiten deutlich geringer aus, da in der Regel keine detaillierten Konfigurationsmöglichkeiten tieferliegender Hard- bzw. Softwareschichten vollzogen werden können. Unabhängig vom Dienstmodell sollte der Konsument jedoch darauf achten, seine für den Zugriff genutzten Endgeräte frei von Schadsoftware zu halten und nur ausgewählten Benutzern zur Verfügung zu stellen.
- **Sicherheit bei der Übertragung:** Ein Schlüsselmerkmal der Cloud ist die Erreichbarkeit von Diensten über das Internet. Sensible Daten werden vom Konsumenten an den Anbieter verschickt bzw. vom Konsumenten angefragt. Dies impliziert, dass die Daten für eine bestimmte Zeitspanne auf einem öffentlichen und ggf. unsicheren Verbindungskanal befindlich sind. Zur Wahrung von Integrität und Vertraulichkeit müssen daher Mechanismen im Hinblick auf die Ver- und Entschlüsselung von Daten angewendet werden.

Cloud-Dienste am Beispiel

Die schnelle Verbreitung von Cloud Services ist vor allem durch die großen Provider wie Amazon, Google, Microsoft, Rackspace oder Terremark ermöglicht worden. Diese investieren große Summen in ihre Infrastruktur und errichten weltweit Datenzentren, die redundant und nach den Gesetzen des jeweiligen Standortes, die Daten ihrer Kunden verwalten und ihnen entsprechende Services zur Verfügung stellen.

Neben den großen Cloud Service Providern gibt es auch kleinere Angebote wie Apple, Salesforce, VMware oder Dropbox, welche sich auf bestimmte Services spezialisiert haben, z.B. die Distribution von Musik oder CRM.

Im Folgenden werden die Angebote von Amazon und Google genauer betrachtet.

Amazon Web Services

Amazon ist der größte der Clouddienstleister und übertrifft in Sachen Kapitalausstattung seine Konkurrenten um das Fünffache. Dabei bietet Amazon neben kostenpflichtigen Premiumservices auch kostenlose Angebote, mit denen Kunden die Möglichkeiten der AWS testen können. Nutzern wird ein einziger Einstiegspunkt für alle Services der Cloud geboten, was es für Einsteiger leichter macht sich zu orientieren und das genutzte Portfolio zu erweitern.

Seit dem Start der AWS im Jahr 2006 hat Amazon schätzungsweise 12 Milliarden Dollar in seine Services investiert und liegt damit hinter den Investitionen mancher Konkurrenten. Die AWS werden von Unternehmen in über 190 Ländern verwendet, entsprechend verteilen sich die Datenzentren über alle Kontinente, mit einem Fokus auf Asien und Nordamerika. Zusätzlich bietet Amazon seinen Kunden an, verschiedene Standorte je nach Service zu wählen, wo beispielsweise Daten verwaltet und von welchem Standort Rechenleistung bezogen werden soll.

AWS bietet dabei SaaS, PaaS, IaaS Services, die in Form von privaten und öffentlichen Clouds angeboten werden. Amazon garantiert dabei eine Verfügbarkeit von 99.9% der Zeit, was umgerechnet maximal 52 Minuten Ausfallzeit im Jahr bedeutet. Selbst kurze Ausfallspannen können bereits enorme Folgen aufgrund der schieren Größe und Verbreitung der AWS bedeuten, wie Vorfälle in den vergangenen Jahren zeigten.

Elastic Compute Cloud (EC2)

Mit der EC2 können Nutzer virtuelle Server verwalten, dabei kann man zwischen 4 Standorten wählen. In einem nächsten Schritt wählt der Benutzer dann eine Verfügbarkeitsregion für den gewählten Standort aus, auf der sein Server laufen wird. Weiterhin müssen Arbeitsspeicher, Prozessorleistung und Festplattenspeicher spezifiziert werden.

Die virtuellen Server werden aus Amazon Machine Images (AMI) erzeugt. Amazon bietet verschiedene vorgefertigte AMIs an, die sich in Betriebssystem und vorinstallierter Software unterscheiden. Sie stellen es dem Benutzer aber auch frei selber eine wiederverwendbare AMI zu erzeugen. Diese kann sogar veröffentlicht und vom Kunden als eigenes Produkt vertrieben werden.

Zur Identifikation gegenüber einer gehosteten Instanz wird ein Public Key Verfahren eingesetzt. Für weitere sicherheitsrelevante Einstellungen befindet sich jede Instanz in einer sogenannten Security Group, die frei konfigurierbare Regeln anbietet.

Nach erfolgreicher Einrichtung wird einer Instanz eine private sowie eine öffentliche IP eingerichtet. Über die öffentliche IP ist der Server aus dem Web erreichbar, die Private dient zur Kommunikation zwischen Instanzen in der AWS. Beide IPs sind jedoch dynamisch und werden nach Neustart der Instanz neu vergeben, daher muss man zusätzlich eine statische elastische IP Adresse reservieren.

Änderungen an der laufenden Instanz werden nicht gespeichert, dafür müssen Zustandsänderungen entweder bei großen Datenmengen Amazons S3 verwendet werden oder Amazons Elastic Block Store (EBS). EBS funktioniert in dem Fall wie eine externe Festplatte die an die Instanz angeschlossen wird.

Alle Vorgänge lassen sich über Kommandozeilenbefehle realisieren, es gibt jedoch auch weitere Werkzeuge wie die AWS Management-Konsole sowie das AWS Toolkit für Eclipse, um mit seinen Instanzen zu interagieren.

Amazon Simple Storage Service (S3)

S3 ist ein Massenspeicher mit einem eigenen Dateisystem. Die bis zu 5 GB großen Web-Objekten liegen hier in sogenannten Buckets und können über einen Bucket-Namen angesprochen werden. Der Zugriff auf die Objekte erfolgt standardmäßig über Web Services und ist über eine SOAP oder REST API möglich. Hierfür stehen ebenfalls komfortable, grafische Werkzeuge zur Verfügung. Zusätzlich gibt es auch Tools von Drittanbietern, die einen Zugriff auf den Speicher, wie auf weit entfernte Festplatten ermöglichen.

Amazon Elastic Block Store (EBS)

Mit dem Speicherdiensst können Datenspeicher von 1GB bis 1TB erzeugt werden, sogenannte Volumes. Ein Volumen kann ein beliebiges Dateisystem beinhalten und verhält sich dabei wie ein USB Stick. Ein Volumen kann lediglich an einer Instanz (AMI) angehängt werden, die in derselben Verfügbarkeitszone angelegt wurde wie das Volumen. Der Speicher ist persistent.

Amazon Simple Queue Service (SQS)

Die Nachrichtenwarteschlange SQS wird zur Skalierung von Anwendungen in der AWS eingesetzt. Sender können hier Nachrichten in die Warteschlange einstellen, welche von registrierten Empfängern dann abgearbeitet werden. Eine dienstnehmende Komponente kann ihre Arbeitsaufträge in die Warteschlange einstellen, wo sie eine dienstgebende Komponente abholt. So können bei Engpässen der kritischen Komponenten, diese parallel auf mehreren EC2 Instanzen laufen.

Amazon SimpleDB

SimpleDB ist auf eine einfach strukturierte, dafür hoch effiziente und zuverlässige Datenhaltung ausgelegt. Die transaktionalen Möglichkeiten sind hierbei stark eingeschränkt, werden jedoch für ein breites Spektrum an Anwendungen als ausreichend erachtet.

Amazon Relational Database Service

Als Gegenwurf zur SimpleDB steht das Amazon RDS, ein Plattformdienst zur einfachen Einrichtung, Betreiben und Skalieren von relationalen Datenbanken in der Cloud. Dabei gewährt Amazon Zugriff über die bekannten Funktionen einer MySQL Datenbank. Bei der Einrichtung wählt RDS automatisch die optimalen Parameter, unter Berücksichtigung von Rechenressourcen und Speicherkapazitäten. Diese lassen sich jedoch im Nachhinein auch entsprechend anpassen.

RDS stellt elastische Kapazitäten zur Verfügung und erledigt gleichzeitig zeitraubende Datenbankverwaltungsaufgaben. Automatisierte Backups und Snapshots können in individuell definierten Intervallen durchgeführt werden und wodurch eine hohe Zuverlässigkeit erreicht wird.

Bei Bedarf kann die Datenbank vertikal skaliert werden, bei sehr aufwendigen Leseoperation kann durch sogenannte Read-Replika Instanzen horizontal repliziert werden. Ein kostenloses Hochverfügbarkeitsangebot bietet die Möglichkeit synchron replizierte Instanzen in mehreren Verfügbarkeitszonen bereit zu stellen. Bei Ausfällen oder Wartungsfenstern in einem Standort wird automatisch zwischen den replizierten Instanzen gewechselt. Über Amazon CloudWatch kann man schließlich die Auslastung der Datenbank überwachen.

Google Apps

Das App-Angebot von Google ist stark gewachsen in den vergangenen 5 Jahren. Das erste Angebot war Google Docs, was zum direkten Konkurrenten Microsoft Office, die Probleme von verschiedenen Versionen und inkompatiblen Formaten untereinander umgehen sollte. Zusätzlich wurden Features wie das kollaborative Arbeiten an einem Dokument hinzugefügt, sowie die Möglichkeit geboten Dokumente in verschiedenen Formaten jederzeit hoch und runterzuladen.

In den nächsten Jahren wurden die Services mit Gmail, Google Drive, Google Talk, Google Calendar, Google Video, Google Labs und Google Play erweitert. Da alle diese Services kostenfrei verfügbar sind und Drive beispielweise großzügige 30 GB zur Verfügung stellt, verbreiteten sie sich entsprechend schnell.

Google hat schätzungsweise bis zu 21 Milliarden in ihre Services investiert und somit deutlich mehr als Amazon, bietet seine spezialisierten Services jedoch kostenlos an und finanziert diese über Werbeangebote.

App Engine

Googles App Engine ist ein PaaS Service und dient als Programmierumgebung und Ausführungsumgebung für neu entwickelte Apps. Unterstützt werden Java und Python. Zum Testen kann die Anwendung auf einer lokalen Laufzeitumgebung gestartet werden, schließlich kann man eine fertige Version auf die Google Infrastruktur übertragen und ausführen lassen.

Zusätzlich zu den Funktionalitäten der Java Bibliotheken bietet die Plattform weitere Services, die über die Standardschnittstellen von Java angesprochen werden können und die Portabilität sowie die Benutzbarkeit des Codes erweitern.

- Zur Speicherung von Daten verwendet die Plattform den sogenannten DataStore, eine schemalose objektorientierte Datenbank. Angesprochen werden kann sie mit Hilfe von JDO und JPA.
- Des Weiteren können über die JavaMail Schnittstelle Funktionen zum Senden und Empfangen von Emails via Google-Mail Konten implementiert werden.
- Zur Authentifizierung werden Google Konten verwendet, welche mit Hilfe eines rudimentären Rechte-Managements zwischen Administratoren und Nutzern der App unterschieden werden können.

Für Entwickler gibt es ein kostenloses Tageskontingent ans Nutzbarer CPU Leistung und Speicher, welches für die meisten kleineren Applikationen ausreichend ist. Erst bei größeren Projekte werden teurere Premiummodelle notwendig.

Google Storage

Der Speicherdienst funktioniert ähnlich dem S3 Service von Amazon und ermöglicht das Speichern von Web-Objekten in den Datenzentren von Google, die über REST abgerufen werden können. Auch hier werden Objekte in sogenannte Buckets gespeichert. Mit Hilfe des Kommandozeilenwerkzeugs GSUtil können Objekte hochgeladen und Buckets angelegt werden., sowie Zugriffsrechte verwalten.

Google Cloud Print

Durch den Cloud Print Service können sich netzwerkfähige, kompatible Drucker in der Cloud registrieren lassen und anschließend von jeder Applikation, die Google Cloud Print unterstützt, über das Internet angesprochen werden. So fallen die Installation von einzelnen Treibern und eventuelle Kompatibilitätsprobleme weg. Drucker können mit anderen Google Konten geteilt werden und so den Zugriff gestatten oder gleich als Dienstleister auftreten.

MS Cloud Design Patterns

Zusammenfassung des Buches:

Titel: MS Cloud Design Patterns - Prescriptive Architecture Guidance for Cloud Applications

Verfasser: Alex Homer, John Sharp, Larry Brader, Masashi Narumoto, Trent Swanson

Verlag: Microsoft

Jahr: 2014

ISBN: 978-1-62114-036-8

Zusammenfassung von: Nils Kohlmeier, Daniel Beneker, Jonas Wiese, Malte Berg, Tolga Aydemir, Sven Schirmer, Yannick Kloss, Niklas Harting, Timo Rolfsmeier, Alexander Schwietert, Lukas Taake, Fabian Lorenz, Philipp Viertel, Benjamin Schmidt, Lutz Winkelmann, Tim Jastrzembski, Gamze Soylev Oektem, Justin Jagnieniak, Christian Holzberger, Marcel Dzaak, Andrei Günther, Oliver Nagel, Marvin Schirrmacher, Jonathan Jansen

Einleitung

Dieses Dokument stellt eine Zusammenfassung von Design-Pattern im Hinblick auf Cloud Computing dar und richtet sich dabei an das Buch "Cloud Design Patterns - Prescriptive Architecture Guidance for Applications" (ISBN: 978-1-62114-036-8). Im Buch werden acht grundlegende Aspekte für Probleme in der Entwicklung im Bereich Cloud Computing betrachtet:

- **Verfügbarkeit:**

Die Verfügbarkeit ist definiert als die Zeit, indem das System funktions- und arbeitsfähig ist. Für gewöhnlich wird diese Zeit prozentual angegeben und mit Benutzern in Form eines Service-Level-Agreements (SLA) vereinbart.

- **Datenmanagement:**

Datenmanagement ist eines der Schlüsselaspekte im Bereich des Cloud-Computings. Im Hinblick auf Anforderungen bezüglich Performance, Skalierbarkeit oder Verfügbarkeit liegen Daten an mehreren Orten und auf mehreren Servern gestreut vor. Hier sind bspw. Datensynchronisation und Datenintegrität gefordert.

- **Design und Implementierung**

Entscheidungen in der Design- und Entwicklungsphase haben immense Auswirkungen auf die Qualität und die Kosten einer Applikation. Das Design hat bspw. Einfluss auf die Konsistenz und Kohärenz zusammenhängender Komponenten, die Wartbarkeit eines Systems und die Wiederverwendbarkeit einzelner Komponenten.

- **Benachrichtigungen**

Die dezentrale Natur von Cloud-Applikationen erfordert eine Infrastruktur für Benachrichtigungen, um verteilte Komponenten und Dienste koppeln zu können.

- **Management und Überwachung**

Cloud-Applikationen werden in Datenzentren betrieben, dadurch besteht oft keine volle Kontrolle über die Infrastruktur. Somit gestaltet sich das Management und die Überwachung oft schwieriger als bei einem "on-premises" Betrieb. Applikationen müssen zur Laufzeit Informationen freigeben, um Administratoren die Möglichkeit zu bieten das System zu verwalten und zu überwachen.

- **Performanz und Skalierbarkeit**

Performanz beschreibt die Fähigkeit eines Systems eine Aktion in einem bestimmten Zeitintervall ausführen zu können, während die Skalierbarkeit die Fähigkeit eines Systems beschreibt erhöhte Lasten verwalten zu können, ohne dass sich Auswirkungen in der Performanz oder bei anderen verfügbaren Ressourcen bemerkbar machen.

- **Elastizität**

Elastizität beschreibt die Fähigkeit eines Systems auf Fehler reagieren zu können, sowie sich von Fehlern erholen zu können. Da typischerweise mehrere Kunden einen Dienst auf einem Server in Anspruch nehmen, besteht eine erhöhte Fehlerwahrscheinlichkeit.

- **Sicherheit**

Sicherheit beschreibt die Fähigkeit eines Systems böswillige oder unbeabsichtigte Aktionen zu vermeiden und nur Aktionen innerhalb des System-Designs zu genehmigen. Cloud-Applikationen sind oft öffentlich zugänglich und müssen deshalb vor böswilligen Angriffen und unautorisiertem Zugriff geschützt werden.

Im Folgenden werden diverse Design-Pattern vorgestellt, welche Hilfestellungen für Herausforderungen in der Entwicklung cloudbasierter Systeme bieten und dabei die oben genannten Aspekte einbeziehen.

Cache-Aside Pattern

Das Cache-Aside Pattern beschreibt ein Verfahren, nach dem häufig benötigte Daten für den schnellen Zugriff in einem Cache zwischengespeichert und bei Bedarf aktualisiert werden. Der Ablauf ist wie folgt: Das System greift zuerst auf den Cache zu und prüft, ob die gewünschten Daten im Cache vorhanden sind. Ist dies nicht der Fall, so werden die Daten aus dem eigentlichen Speicher (Datebank, etc.) geladen und im Cache für weitere Zugriffe gespeichert. Werden Änderungen an den Daten durchgeführt, so werden die Änderungen im Speicher übernommen und der entsprechende Cache-Eintrag wird als ungültig markiert, damit bei einem weiteren Zugriff die Daten neu aus dem Speicher geladen werden.

Beim Umsetzen dieses Patterns sollte eine Gültigkeitsdauer der Daten im Cache festgelegt werden. Daten, die eine längere Zeit nicht abgerufen wurden, müssen aus dem Cache entfernt werden, um Platz für Daten zu schaffen, die häufiger benötigt werden. Diese Gültigkeitsdauer sollte nicht zu kurz gewählt werden, sonst werden die Daten ständig neu aus dem Datenspeicher geladen. Bei der Wahl der Gültigkeitsdauer sollte aber auch die Zeit, die benötigt wird, um ein bestimmtes Datum aus einem Datenspeicher zu holen, beachtet werden. So sollte zum Beispiel ein Datum, dass in einer externen Datenbank liegt, die sehr langsam ist, länger im Cache verbleiben als ein Datum, dass in einer Datenbank auf demselben Server liegt und somit schneller geladen werden kann. Ein sehr wichtiger Punkt ist, dass Daten im Datenspeicher durch eine externe Anwendung geändert werden können und die Kopie der Daten im Cache damit nicht mehr gültig ist. Die Anwendung, die den Cache verwendet, merkt die Änderung erst beim erneuten Laden der Daten aus dem Datenspeicher. Verwenden mehrere Anwendungen die gleichen Daten, so sollte man sich die Verwendung von geteilten Caches überlegen, bei der mehrere Anwendungen gemeinsam auf einen Cache zugreifen. Beim Start der Anwendung kann außerdem der Cache mit Daten gefüllt werden, die wahrscheinlich während des Startens benötigt werden.

Das Cache-Aside-Pattern sollte eingesetzt werden, wenn der Bedarf von bestimmten Daten nicht vorhersagbar ist und quasi zufällig ist. Für statische Daten oder für Session-Daten in Client-Server-Anwendungen sollte das Pattern nicht eingesetzt werden.

Circuit Breaker Pattern

Mit diesem Pattern lässt sich auf Fehler, die unterschiedlich lange dauern, angemessen reagieren. Die richtige Anwendung des Pattern kann die Stabilität und Elastizität eines Programms verbessern.

Kontext und Problem

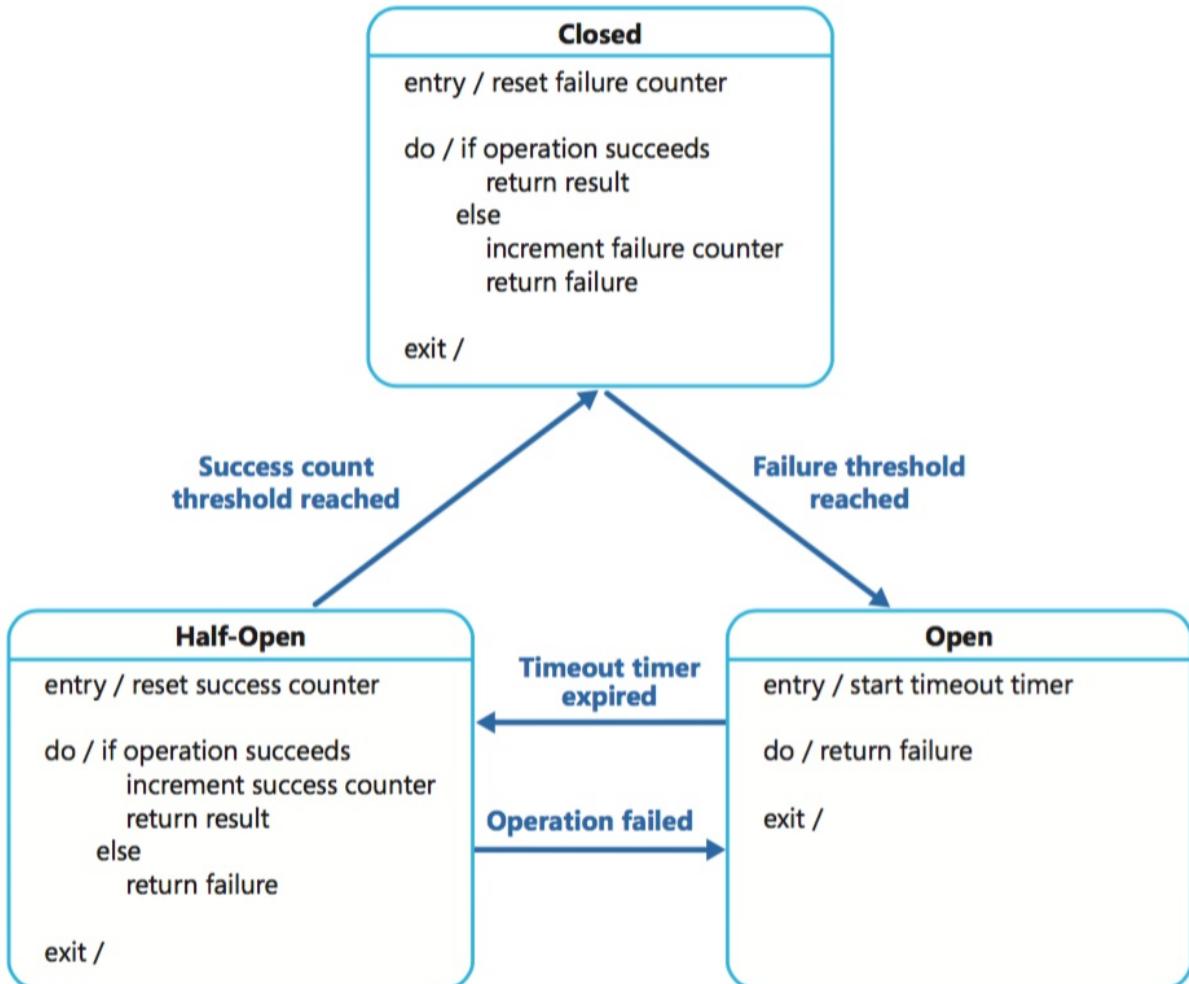
Es gibt zwei Arten von Fehlern: Kurzlebige Fehler, die sich oft nach kurzer Zeit auflösen, wie beispielsweise ein kurzzeitig langsames Netzwerk und größere Fehler, die aus unerwarteten Ereignissen resultieren. Im letzteren Fall ist es sinnlos den ausgefallenen Service weiter anzusprechen. In diesem Fall sollte das Programm den Fehler schnell akzeptieren und entsprechend damit umgehen.

Lösung

Die Idee ist, das Programm davon abzuhalten Operationen, die sehr wahrscheinlich fehlschlagen, (erneut) auszuführen. In Abgrenzung dazu führt das „[Retry Pattern](#)“ Operationen erneut aus, wenn man davon ausgeht, dass sie funktionieren.

Das Circuit Breaker Pattern ist ein Proxy für Operationen die fehlschlagen können. Der Proxy misst die Anzahl vorheriger Fehler und entscheidet auf der Grundlage, ob weitere Operationen zugelassen werden oder nicht.

Implementieren lässt sich dieses Verhalten als State Machine. Die folgende Abbildung stellt die State Machine mit den drei Zuständen Closed, Open und Half-Open dar.



Status: Closed

Dies ist der initiale Zustand. Alle Anfragen durchlaufen den Proxy und werden zugelassen. Kommt es bei einer Anfrage zu einem Fehler, wird er mit Hilfe eines Zählers registriert. Erreicht dieser Zähler einen Schwellwert, wird in den Status „Open“ gewechselt. Der Zähler sollte periodisch zurück gesetzt werden, damit gelegentliche Fehler nicht dazu führen, dass immer in den „Open“ Status gewechselt wird.

Status: Open

Beim Wechsel in diesen Status wird ein Timeout-Timer gestartet. Der Proxy blockiert alle Anfragen und gibt entsprechende Exceptions zurück. Die Länge desTimeouts entscheidet darüber, wie lange alle Anfragen blockiert werden sollen. Ist der Timer abgelaufen, wird in den Status „Half-Open“ gewechselt.

Status: Half-Open

In diesem Status wird ein Teil der Anfragen zugelassen. Mit diesem Status wird versucht zu erkennen, ob der entsprechende Service wieder erreichbar ist oder nicht. Es wird nur ein Teil der Anfragen zugelassen, um zu verhindern, dass ein Service, der wieder erreichbar ist, plötzlich mit so vielen Anfragen überflutet wird, dass er sofort wieder abstürzt. Alle erfolgreichen Anfragen werden gezählt. Schlägt eine Anfrage fehl, wird sofort wieder in den Status „Open“ zurück gewechselt. Wurde eine bestimmte Anzahl von Anfragen erfolgreich durchgeführt, ist davon auszugehen, dass der Fehler behoben ist. Es wird wieder in den Status „Closed“ gewechselt.

Weitere Aspekte

- Exception Handling: Falls vom Circuit Breaker eine Exception geworfen wird, sollte der Empfänger entsprechend reagieren, wie beispielsweise dem Benutzer eine Meldung anzuzeigen, die Oberfläche für eine gewisse Zeit zu deaktivieren oder alternative Operationen anbieten.
- Verschiedene Arten von Exceptions: Circuit Breaker selbst kann verschiedenste Exception erhalten, wenn er eine Anfrage ausführt und diese fehlschlägt. Auf verschiedene Exceptions könnte unterschiedlich reagiert werden. So könnten beispielsweise mehr Timeout Fehler eintreten, bis in den „Open“ Status gewechselt wird, als bei einer allgemeinen Exception, die auf einen schwerwiegenderen Fehler hinweist.
- Logging: Fehlgeschlagene Anfragen sollten gespeichert werden. Außerdem sollte die State Machine an sich überwacht werden. Bei einem Wechsel in den Status „Open“ kann beispielsweise der Systemadministrator benachrichtigt werden.
- Fehlgeschlagene Operationen testen: Statt eines Timeout-Timers im „Open“ Status, kann der nicht erreichbare Service testweise angesprochen werden, um festzustellen, wann der Service wieder erreichbar ist. Dies kann mit einem einfachen Ping-Befehl oder über einen speziellen Health Endpoint realisiert werden.
- Manuelle Steuerung: Es sollte möglich sein, dass beispielsweise der Systemadministrator manuell einen bestimmten Status setzen kann.
- Overhead: Ein Circuit Breaker kann Ziel sehr vieler Anfragen werden und sollte daher nur einen kleinen Overhead erzeugen.
- Ressourcen differenzieren: Angenommen in einem Data-Store-Service ist eine Datei nicht mehr erreichbar, dann könnte es sein, dass der Circuit Breaker in den Status „Open“ wechselt und alle Anfragen blockiert, obwohl alle anderen Dateien des Data-Stores problemlos erreichbar sind.
- Fehlgeschlagene Anfragen erneut ausführen: Statt in einem Fehlerfall direkt eine Exception zu werfen, können auch alle Anfragen gesammelt werden und ausgeführt werden, sobald der Service wieder erreichbar ist.

Competing Consumers Pattern

Falls eine Cloud basierte Anwendung viele Anfragen bearbeiten muss, die Anzahl der Anfragen aber stark schwankt, kann es sinnvoll sein, eine „Message Queue“ zu verwenden. Mit Hilfe dieser Queue kann die Anwendung (Producer) die Nachrichten zu anderen Anwendungen (Consumer Services) umleiten. Diese Consumer Services können die Nachrichten dann Asynchron und frei Skalierbar abarbeiten.

Vorteile

Da die Anzahl der Consumer Services variierbar ist, ermöglicht dieses Pattern eine inhärente Last Verteilung. Falls viele Anfragen eintreffen können weitere Instanzen des Consumer Service gestartet werden. Diese können, wenn die Anzahl der Anfragen abnimmt, abgeschaltet werden. Da die Message Queue als Buffer dient, ist eine Synchronisierung zwischen Producer und Consumer nicht nötig. Die Queue führt außerdem dazu, dass jede Anfrage mindestens einmal bearbeitet wird. Falls ein Consumer während der Bearbeitung einer Anfrage abstürzt, kann die Queue so konfiguriert werden, dass eine andere Instanz die Anfrage erneut bearbeitet.

Nachteile

Durch die Verwendung einer Message Queue ist nicht garantiert, dass die Nachrichten in der gleichen Reihenfolge, wie sie eingetroffen sind bearbeitet werden. Da die Queue nur dafür sorgt, dass jede Nachricht mindestens einmalig bearbeitet wird, kann es vorkommen, dass eine Nachricht mehrfach abgearbeitet wird. Da durch die Queue keine direkte Kommunikation zwischen Producer und Consumer erfolgt, sie also füllig entkoppelt sind, ist es schwierig die Ergebnisse des Consumers zurück an den Producer zu liefern. Auch die Message Queue kann bei zu vielen Nachrichten zu Flaschenhals werden.

Fazit

Das Pattern sollte verwendet werden, wenn die Anzahl der Anfragen stark variiert und die Aufgaben der Anwendung gut Asynchron zu verarbeiten sind. Außerdem müssen die Aufgaben unabhängig voneinander sein und parallel verarbeitet werden können. Falls die Aufgaben nicht gut voneinander getrennt werden können oder die Reihenfolge der Aufgaben wichtig ist, ist das Pattern nicht sinnvoll zu verwenden.

Compute Resource Consolidation Pattern

Dieses Pattern kann verwendet werden, um Berechnungen mehrerer Operationen oder Aufgaben in einer zusammenhängenden Einheit durchzuführen. Hierdurch kann die Ressourcennutzung optimiert und die Kosten gesenkt werden.

Hintergründe und Probleme

Eine Cloud Anwendung enthält häufig viele Operationen und Aufgaben, welche meist standartmäßig in getrennten Einheiten, beispielsweise auf verschiedenen virtuellen Maschinen, verarbeitet werden. Dieses Konzept verursacht eine große Menge an verarbeitenden Einheiten, welche das System sehr komplex machen. Außerdem verursachen diese Einheiten Kosten, auch wenn sie inaktiv oder unausgelastet sind. Ein weiteres Problem sind die erschwerten Kommunikationswege, welche durch die Aufteilungen entstehen.

Lösung

Um den genannten Problemen entgegenzuwirken, können mehrere Aufgaben und Operationen in eine verarbeitende Einheit zusammengelegt werden. Diese müssen dafür zuallererst gruppiert werden. Eine verbreitete Möglichkeit ist es zu prüfen, welche Aufgaben und Operationen ähnliche Anforderungen und Verarbeitungszeiten besitzen.

Zusammengefügt können diese Einheiten je nach Auslastung skalieren und somit mehr Rechenleistung, in Form von weiteren berechnenden Einheiten, hinzufügen oder bei geringer bzw. gar keiner Nutzung Rechenleistung abtreten. Wichtig ist es jedoch darauf zu achten, wie die Aufgaben und Operationen, welche gruppiert werden sollen während der Nutzung skalieren. Werden beispielsweise Operationen gruppiert, welche teilweise viele neue Einheiten für die Verarbeitung benötigen und der andere Teil nur auf bestimmte Ereignisse wartet, könnten unnötigerweise bei Ressourcenanforderungen auch dem Warteprozess erhöhte Ressourcen zugeschrieben werden, obwohl diese nicht benötigt werden. Verarbeitende Einheiten können unterschiedliche Rechenleistungen wiederspiegeln. Daher sollte darauf geachtet werden, dass Einheiten mit hohen Rechenleistungen, welche dementsprechend teurer sind, eine durchgängige und nicht zu schwache Auslastung haben. Daher sollten Operationen und Aufgaben welche viel Leistung in kleinen Schüben benötigen zusammengelegt und einer leistungsstarken Einheit zugewiesen werden. Dagegen sollten langwierige und viel Ressourcen benötigende

Aufgaben und Operationen nicht zusammengefasst werden, da es zu einer Überlastung der Einheit führen kann. Eine kontinuierliche Analyse der Gruppierungen hilft dabei diese optimal anzutragen.

Berücksichtigungen

Bei der Implementierung des Patterns sollten unbedingt vorher bestimmte Aspekte berücksichtigt werden. Hierzu zählt wie bereits beschrieben die Entscheidung der Gruppierung anhand ähnlicher Skalierungsanforderungen und Verarbeitungszeiten. Jedoch sollten Operationen mit hohen Ressourcenanforderungen nicht zusammen in eine Einheit gruppiert werden. Ebenfalls sollte darauf geachtet werden, dass wenn eine Operation oder Aufgabe regelmäßig gewartet wird und dies ein Neuaufsetzen der verarbeitenden Einheit erfordert, alle gruppierten Operation der Einheit ebenfalls gestoppt und neu geladen werden müssen. Hinzukommend kann es sein, dass die Operationen sich denselben Kontext in der Einheit teilen und somit auf Ressourcen des anderen zugreifen können. Die hierdurch mögliche Sicherheitslücke weitet sich mit der Anzahl der sich teilenden Operationen in einer Einheit aus. Ebenso können Fehler die von einer Operation ausgelöst wurden, das Verhalten anderer Operationen in derselben Einheit einschränken. Die Komplexität steigt durch das Gruppieren von Operationen zu einer Einheit, was die Wartbarkeit und Testbarkeit verschlechtert. Außerdem ist zu berücksichtigen, dass die Cloud Infrastruktur regelmäßige Aufräumarbeiten tätigt. Einheiten mit mehreren lang laufenden Operationen sollten daher so konfiguriert werden, dass diese nicht vorzeitig unterbrochen werden. Beim Entwickeln einer Operation sollte darauf geachtet werden, diese so zu entwerfen, dass eine Änderung der Umgebung keine Auswirkung auf die Operation hat und weiterhin funktioniert.

Anwendungsfälle

Dieses Pattern sollte für wenig bzw. regelmäßig keine Leistung benötigende Operationen und Aufgaben verwendet werden, welche in einer eigenen Einheit nicht kosteneffektiv sind. Sobald eine Operation Ausfallsicher sein soll oder strikten Sicherheitsanforderungen entsprechen muss ist von einer Anwendung des Pattern abzusehen und eine eigene Einheit zu verwenden.

Command and Query Responsibility Segregation (CQRS) Pattern

Das CQRS-Pattern führt eine Teilung zwischen den `data transfer objects` (DTO) ein, sodass das Schreiben von Daten (Command) und Lesen von Daten (Query) über zwei verschiedene Interfaces - respektive DTOs - geschieht.

Problemstellung

Klassisch geschehen alle Schreib- und Leseoperationen auf ein Entitiy (in der Datenbank) über ein und das selbe DTO und dabei wird immer mit ein und der selben Datenbank kommuniziert. Das heißt, alle CRUD-Operationen haben das selbe Zielobjekt. Zum Beispiel wird ein Kunden-DTO über den `data access layer` (DAL) angefordert und auf dem Bildschirm angezeigt. Ein Nutzer der Applikation aktualisiert nun einige Felder des Kunden-DTO und speichert. Die Applikation wird nun diesen DTO über den DAL auf die Datenbank speichern. Ein DTO wurde also sowohl für das Lesen, als auch für das Schreiben genutzt.

Diese Art und Weise mit den CRUD-Operationen und DTOs umzugehen, funktioniert gut, solange die Business Logik, die auf diese Daten und Operationen angewandt werden soll, gering ist.

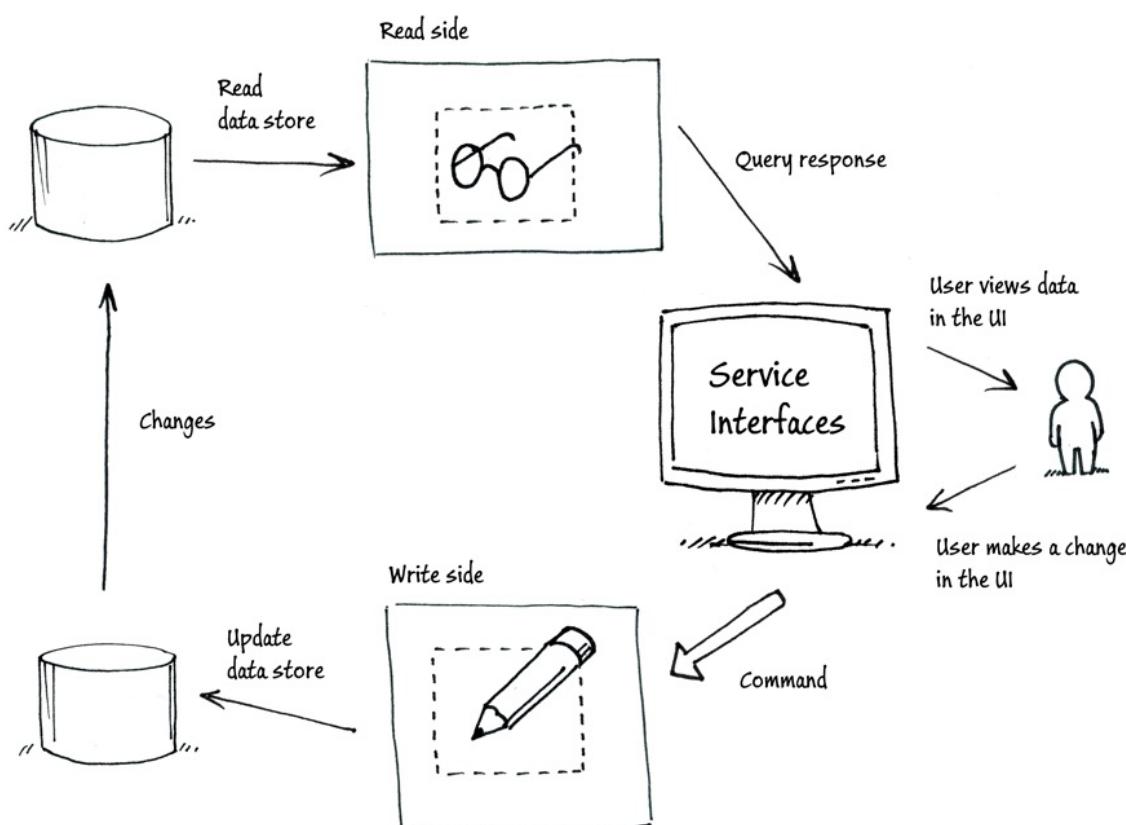
Die Nachteile:

- Oft kommt es zu Fehlanpassungen zwischen den Lese- und Schreibdarstellungen der Daten; zum Beispiel zusätzliche Spalten oder Eigenschaften, die korrekt aktualisiert werden müssen, obwohl sie nicht als Teil einer Operation benötigt werden.
- Viele Nutzer, die parallel auf die Datenbank zugreifen, steigern die Gefahr, dass die Integrität der Daten nicht gewährleistet werden kann. Die ausgiebige Nutzung von Transaktions-Mechanismen des RDBMS, sind unumgänglich. Die Performance der Applikation wird dadurch nicht steigen und die Komplexität der Querys nimmt zu.
- Die Verwaltung der Sicherheit und Berechtigungen werden umständlicher, da jede Entität sowohl Lese- als auch Schreiboperationen unterliegt, die versehentlich Daten im falschen Kontext ausliefern können.

Lösung

CQRS trennt die Lese- und Schreiboperationen und nutzt dafür unterschiedliche Interfaces. Dadurch kann sogar eine Trennung der `data models` erfolgen (nicht verwechseln mit den DTOs). Durch Datenmodelle, die für Lese- bzw. Schreiboperationen optimiert wurden, kann die Komplexität der Implementierung reduziert werden. Ein Nachteil für einen Entwickler ist, dass mit diesen CQRS-Modellen nicht einfach Code generiert werden kann (getter/setter).

Diese Datenmodelle können die selbe physische Datenbank ansprechen, jedoch macht es aus Sicht der Sicherheit, der Performance und der Skalierbarkeit Sinn, auch die Datenbanken zu trennen. Die Lese-Datenbank kann dann ein `read-only`-Duplikat der Schreib-Datenbank sein. Dadurch kann die Skalierung der Infrastruktur deutlich vereinfacht werden, da (meistens) deutlich mehr Leseoperationen in einer Applikation stattfinden, als Schreiboperationen. Das heißt, es können mehrere Lese-Datenbanken aufgesetzt werden, um so die Arbeitsgeschwindigkeit der Applikation zu erhöhen. Außerdem kann die Lese-Datenbank und die Schreib-Datenbank dahingehend optimiert werden, dass diese verschiedene Normalisierungsgrade haben. Die Schreib-Datenbank kann zum Beispiel nach der dritten Normalform optimiert werden. Die Lese-Datenbank kann wiederum eine denormalisierte Datenbank verwenden, um möglichst schnelle Abfragen zu ermöglichen.



(Abbildung 1: Eine mögliche Implementierung des CQRS-Pattern. Quelle: <https://msdn.microsoft.com/en-us/library/jj591573.aspx>)

Wann sollte CQRS verwendet werden

- Wenn die Applikation den parallelen Zugriff mehrerer Nutzer auf die selben Daten ermöglicht und es häufig vorkommt.
- Wenn bei den Schreiboperationen viel Business-Logik auf die Daten angewandt wird und diese Daten immerzu validiert werden, um die Konsistenz der Daten zu gewährleisten. Das Lese-Modell gibt nur eine einfache DTO wieder und besitzt keinerlei Business-Logik in diesem Fall.
- Wenn die Leseoperationen deutlich öfter stattfinden als Schreiboperationen und das Ziel der schnellere Zugriff auf die Daten ist.
- Wenn bei dem Entwicklungsprozess Entwickler-Teams mit unterschiedlichen Erfahrungswerten arbeiten. So kann das erfahrenere Team die komplexen Schreib-Modelle implementieren und das weniger erfahrenere Team implementiert die Lese-Modelle und kümmert sich um das User Interface.

Wann sollte CQRS nicht verwendet werden

- Wenn die Business-Logik relativ einfach ist.
- Wenn eine einfache Bedienoberfläche des CRUD-Typs und die damit verbundenen Datenzugriffsoperationen ausreichen.
- Wenn es sich um die Umsetzung durch das gesamte System handelt. Es gibt Komponenten in einem System, da kann die Implementierung des CQRS leichter sein. Diese sollten bevorzugt umgesetzt werden. Unnötige Komplexität sollte vermieden werden.

Event Sourcing Pattern

Bei dem Event Sourcing Pattern werden alle Veränderungen des Zustands eines Systems oder der Daten als Sequenz von Events gespeichert.

Problematik

Das herkömmliche [create, read, update, and delete \(CRUD\)](#) Model birgt einige Nachteile: Zunächst werden bei einem Update die alten Werte mit neuen Werten überschrieben. Das konventionelle Update impliziert demnach eine Löschung (Delete) der alten Daten, sofern kein Logfile existiert, welches alle Änderungen aufzeichnet. Bei einem Updatevorgang werden die Daten meist gesperrt, wodurch andere Benutzer und Anwendungen keine Änderungen an den Daten machen können. Des Weiteren werden bei einem CRUD System die Operationen direkt auf dem Data Store durchgeführt. Hieraus resultierte eine schlechtere Performance und Reaktion, sowie Skalierbarkeit des Systems.

Lösung

Die Anwendung beschreibt die Änderungen der Daten als Events. Auch das Löschen der Daten ist ein Event. Diese Events werden asynchron an den Event Store verschickt und dort persistent abgelegt. Eine Löschung der Events im Event Store ist nicht vorgesehen, die Events sind immutable. Der Event Store publishes die Events an einen Topic, sodass alle Consumer die den Topic abonniert haben, die Events bekommen und verarbeiten können. Typische Abonnenten sind z.B. Controller für die Materialized View, die daraufhin die View aktualisieren. Für die Vereinfachung der Präsentationsschicht wird meist für jede Entität der derzeitige Datenstand als materialized View gespeichert.

Das Event Sourcing Pattern wird meist zusammen mit dem [CQRS-Pattern](#) eingesetzt.

Vorteile

Der Prozess, der die Events verarbeitet, kann von dem Prozess, welcher das Event erstellt hat, entkoppelt werden, sodass die Verarbeitung später erfolgen kann. Außerdem entstehen durch die Entkopplung neue Skalierungsmöglichkeiten. Der gesamte Systemänderungsverlauf bzw. die Datenhistorie kann durch eine erneute Betrachtung der

Events erfolgen. Dies ist hilfreich z.B. für das Debugging und Testen der Systeme. Zudem kann hier auch die Business Logik einer Historisierung der Entitäten erfolgen. Generell hat der Event Log einen hohen Business Value.

Berücksichtigungen

Durch eine fehlende Allokation der Daten, kann es zu Misständen kommen. Besonders wenn mehrere verschiedene Anwendungen oder auch Instanzen Events publishen wollen. Die korrekte Reihenfolge kann so auch nicht immer gewährleistet werden. Ein Timestamp der Events oder eine autoinkrementelle ID-Nummer kann hier für eine ACID-Fähigkeit Abhilfe schaffen. Außerdem ist immer eine Verzögerung der zwei entkoppelten Prozesse zu bemerken, wenn das Event gepublished wurde wartet es noch auf seine Verarbeitung.

Der aktuelle Datenstand wird durch die Verarbeitung aller Events bestimmt. Bei einer Vielzahl an Events empfiehlt es sich Snapshots zu machen, um eine Verarbeitung aller Events zu umgehen, sodass nur die Events ab dem Snapshot verarbeitet werden müssen.

Empfehlungen für die Benutzung des Patterns

Die Vorteile des Event Sourcing Patterns überwiegen, wenn

- die beiden Prozesse (Datenveränderung, Datenverarbeitung) entkoppelt werden sollen, Flexibilität des Systems
- Datenzustandverläufe von Interesse sind.
- eine hohe Skalierbarkeit erwartet wird.
- schnelle und einfache Anpassung an neue Business Logik gefordert wird.

Eine Verwendung des Patterns ist nicht empfehlenswert, wenn

- die traditionellen CRUD Mechanismen ausreichend sind, Historisierung der Daten nicht benötigt wird.
- ACID Fähigkeit vorhanden sein muss.
- Echtzeitupdates der Views gefordert werden.
- Veränderungen der Datenbestände nicht erfolgen.

External Configuration Store Pattern

Definition

Mit dem *External Configuration Store Pattern* werden Konfigurationsinformationen an einer zentralen Stelle verwaltet. Diese werden von allen Anwendungen und den Instanzen von Anwendungen verwendet. Dieses Pattern soll für ein vereinfachtes Management und eine bessere Steuerung der Konfigurationen beitragen.

Probleme mit herkömmlichen Lösungen

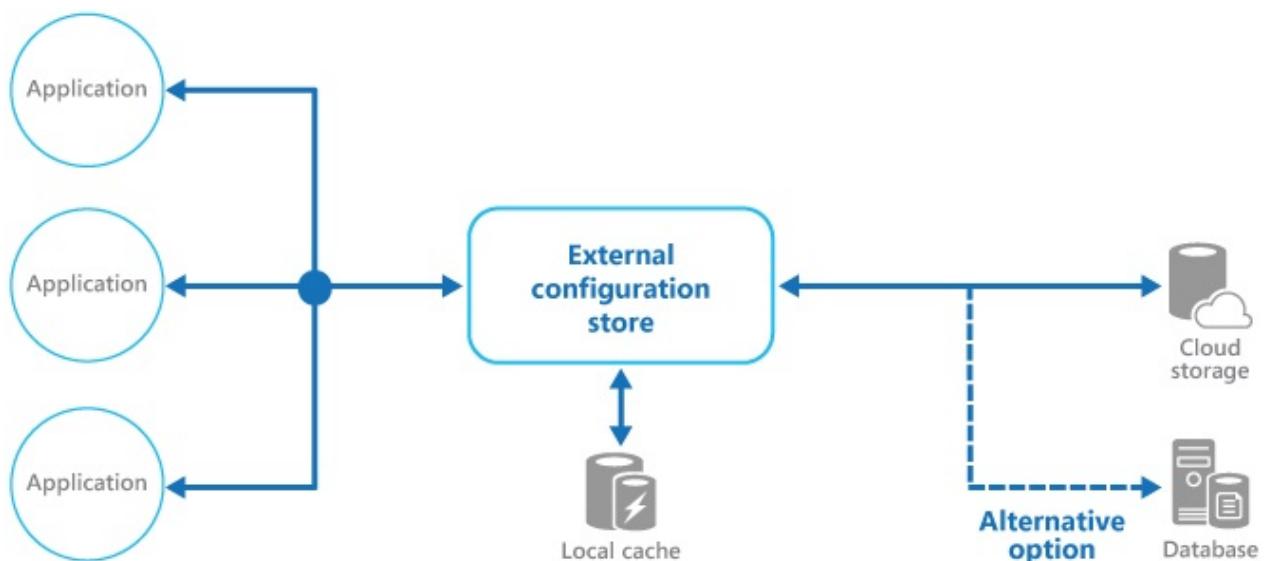
Eine Vielzahl von Anwendungen haben Konfigurationsinformationen in Dateien gespeichert, welche in Laufzeitumgebungen einer Cloud ausgeführt werden. Bei manchen Anwendungen lassen sich diese verändern und die Anwendung reagiert sofort darauf. In anderen Fällen müssen diese jedoch neu gestartet werden, was die Anwendung in dieser Zeit unerreichbar macht und zusätzlichen Administrationsaufwand erfordert. Ein weiteres Problem ist, wenn in einer Cloud alle Instanzen einer Anwendung geupdated werden und diese während dem Vorgang unterschiedliche Konfigurationen verwenden. Die Authentifizierung von Benutzern muss berücksichtigt werden.

In vielen Anwendungsfällen ist es von Vorteil, wenn Konfigurationen von einer zentralen Stelle für Anwendungen gemeinsam genutzt werden.

Das Pattern

Das *External Configuration Store Pattern* sieht es vor, dass Konfigurationsinformationen an einer externen, zentralisierten, Stelle von einem nicht-flüchtigen Speichertyp gespeichert werden. Über definierte Schnittstellen können Konfigurationen gelesen und verändert werden. In einem typischen Cloud Szenario könnte der Speicher beispielsweise ein Cloud-basierter Speicherservice sein. Die Schnittstellen müssen konsistent und einfach zu verwenden sein. Die Konfigurationsinformationen sollten in einem strukturierten Format wiedergegeben werden.

Je nach Typ des Speichers kann es von Vorteil sein, wenn die Softwarekomponente, welche für Konfigurationsspeicher zuständig ist, die Konfigurationsinformationen in einen lokalen Cache einliest um die Performance zu erhöhen.



Überblick des External Configuration Store pattern mit optionalem lokalen Cache. (Quelle: <https://docs.microsoft.com/en-us/azure/architecture/patterns/external-configuration-store>)

Wann das Pattern verwendet werden kann

- Wenn sich Anwendungen und Instanzen einer Anwendung Konfigurationsinformationen teilen oder wenn Standardeinstellungen auf mehreren Anwendungen und Instanzen einer Anwendung erzwungen werden müssen
- Wenn das Standard Konfigurationssystem nicht alle geforderten Konfigurationseinstellungen unterstützt
- Wenn ein ergänzender Speicher für einige Konfigurationen benötigt wird
- Wenn ein vereinfachter Mechanismus für die Administration von mehreren Anwendungen benötigt wird

Federated Identity Pattern

Durch die Verwendung dieses Patterns, kann die Entwicklung sowie die Administrierung von Anwendungen vereinfacht werden, indem die Authentifizierung an einen externen Anbieter weitergereicht wird.

Problematik

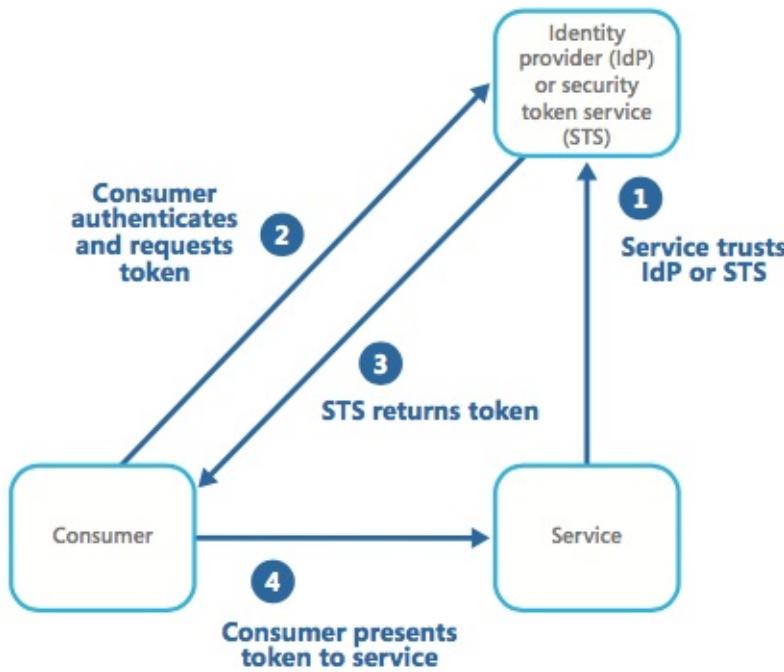
Anwender müssen im Alltag mit vielen verschiedenen Anwendungen arbeiten, für welche sie unterschiedliche Anmeldeinformationen haben. Dies führt zu negativen Erfahrungen für die Anwender, da sie sich viele Passwörter merken müssen. Des Weiteren ist es für die Administratoren ein sehr hoher Aufwand alle Anmeldeinformationen zu verwalten. Neben dem Punkt des hohen Verwaltungsaufwands, führt es auch zu einem Sicherheitsrisiko falls ein Mitarbeiter die Firma verlässt, müssen viele einzelne Anmeldeinformationen gelöscht werden.

Lösung

Um der vorangegangenen Problematik entgegenzuwirken, sollte eine Authentifizierungsfunktion implementiert werden, welche Federated Identity's unterstützt. Das bedeutet, dass die Authentifizierungsfunktion vom Anwendungscode getrennt wird und die Authentifizierung an einen Identity Provider (IdP) wie z.B. Microsoft, Google, Yahoo! oder Facebook weitergereicht wird.

Dies vereinfacht die Entwicklung sowie die Verwaltung der Nutzerdaten. Außerdem hat der Benutzer dadurch die Möglichkeit, sich neben der Standard Authentifizierung über einen frei wählbaren IdP zu authentifizieren. Dadurch muss sich der Anwender keine neuen Anmeldeinformationen merken. Des Weiteren wird durch die Verwendung eines IdP die Authentifizierung von der Autorisierung entkoppelt.

Abbildung 1 veranschaulicht die Funktionsweise des Federated Identity Pattern, wenn eine Anwendung auf einen Dienst zugreifen muss, der eine Authentifizierung mittels eines IdP erfordert.



(Abbildung 1: Übersicht über

eine Authentifizierung)

Berücksichtigungen

Wenn man dieses Pattern verwendet, sollte man verschiedene Aspekte beachten:

- Wenn die Anwendung auf mehrere Rechenzentren verteilt werden soll, muss der Identitätsmanagement Mechanismus in den Rechenzentren ebenfalls bereitstehen, um die Zuverlässigkeit und Verfügbarkeit der Anwendung zu gewährleisten.
- Wenn man einen Identity Provider bereitstellt muss beachtet werden, dass in der Regel keine Informationen über den authentifizierten Benutzer außer einer E-Mail-Adresse und einem Namen zur Verfügung stehen.
- Wenn mehr als ein Identity Provider für die Anwendung bereitsteht muss erkannt werden, zu welchem Identity Provider der Anwender weitergeleitet werden muss.

Anwendungsfälle

Dieses Pattern wird sehr häufig in SaaS (Software as a Service) Anwendung verwendet, um den Nutzern neben der Standard Anmeldung, eine Anmeldung über ihre Accounts für die Sozialen Netzwerke bereitzustellen.

Weitere Anwendungsfälle sind Enterprise Anwendungen, Business-to-Business Anwendungen, Anwendungen welche mit dritt Anbieter Software interagieren sowie Unternehmen, die ihre IT-Systeme zusammengefasst haben.

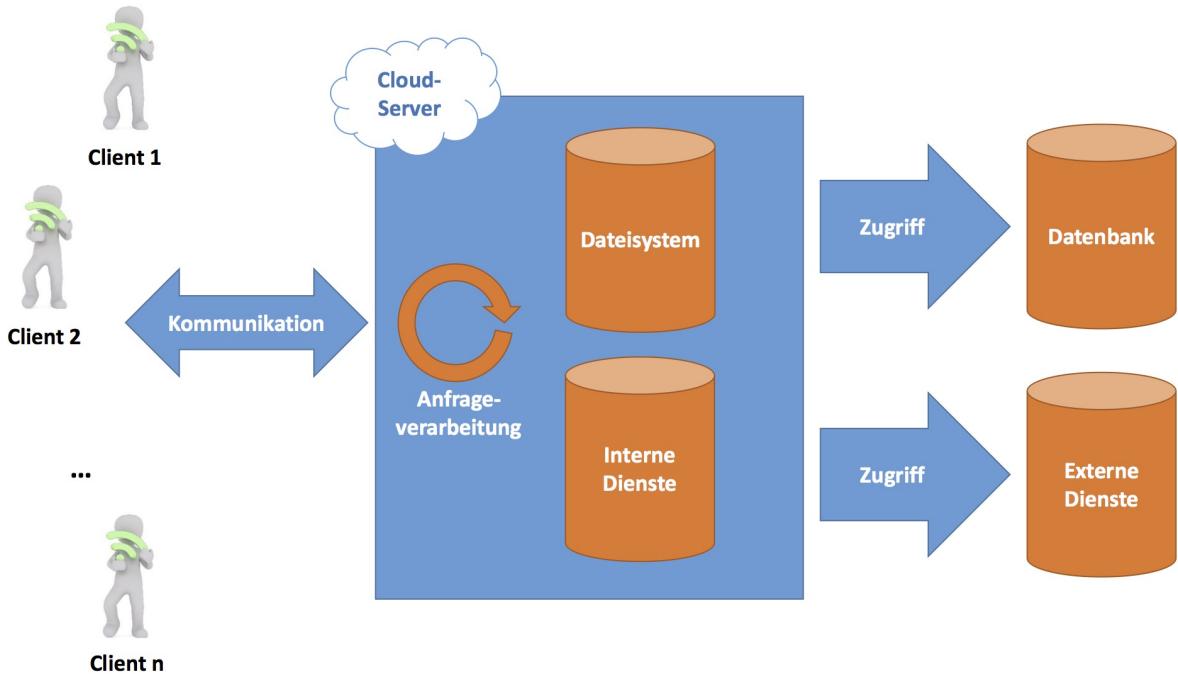
Gatekeeper Pattern

Cloud Computing erfreut sich zunehmender Beliebtheit. Das Auslagern von Diensten und Daten an einen externen Ort bietet diverse Vorteile im Hinblick auf Kosteneffizienz und Skalierbarkeit. Auch die zentrale Zugänglichkeit ist ein entscheidender Faktor, der die fortlaufende Verbreitung des Trends begünstigt. Jeder Benutzer kann zu jeder Tages- und Nachtzeit und an jedem Ort der Erde auf die ihm zugehörigen Informationen und Dienste zurückgreifen. Alles, was er dazu benötigt, ist eine Netzwerkverbindung sowie ein netzwerkfähiges Endgerät. Die zentrale Lage von Ressourcen macht jedoch auch einen potentiellen Schwachpunkt der Cloud-Architektur aus.

Problematik

Ein cloud-typisches Interaktionsszenario beinhaltet zwei verschiedene Akteure. Vom Client bzw. Endnutzer geht eine initiale Anfrage aus. Beispielsweise möchte er sich authentifizieren, vertrauliche Auskünfte beantragen oder eine Anwendung nutzen. Die Anfrage wird ohne Umwege an einen zentralen Server gesendet und anschließend von diesem verarbeitet. In Abhängigkeit von der Forderung des Clients führt der Server verschiedene Aktionen aus. Beispielhaft sei hierfür die Kommunikation mit Datenbanken oder das Anstoßen von externen Diensten. Die gesammelten Ergebnisse werden letztlich vom Server wieder an den Client übermittelt, sodass die Anfrage abgeschlossen werden kann.

Eine Problematik, die sich durch diese Kommunikationsarchitektur ergibt, beruht auf der fehlenden Trennung der Programmlogik innerhalb des Servers (siehe Abb. 1). Dieser dient zum einen als direkter Kontaktspunkt für eine beliebige Anzahl an Clients. Hierunter können sich sowohl rechtmäßige Benutzer, welche einen tatsächlichen Anspruch auf ihre jeweilige Forderung haben, als auch unrechtmäßige Benutzer befinden, denen keine Leistung erbracht werden darf. Der Server ist dementsprechend öffentlich und von jedem Endgerät aus kontaktierbar. Zum anderen ist derselbe Rechenknoten für die Orchestrierung und Instanziierung von Diensten sowie das Auslesen und die Übermittlung von vertraulichen Daten zuständig. Hier lässt sich ein Widerspruch erkennen. Auf derselben Rechnerinstanz findet gleichzeitig die Verarbeitung von unbekannten Anfragen sowie der Umgang mit sensiblen Informationen bzw. Diensten statt.



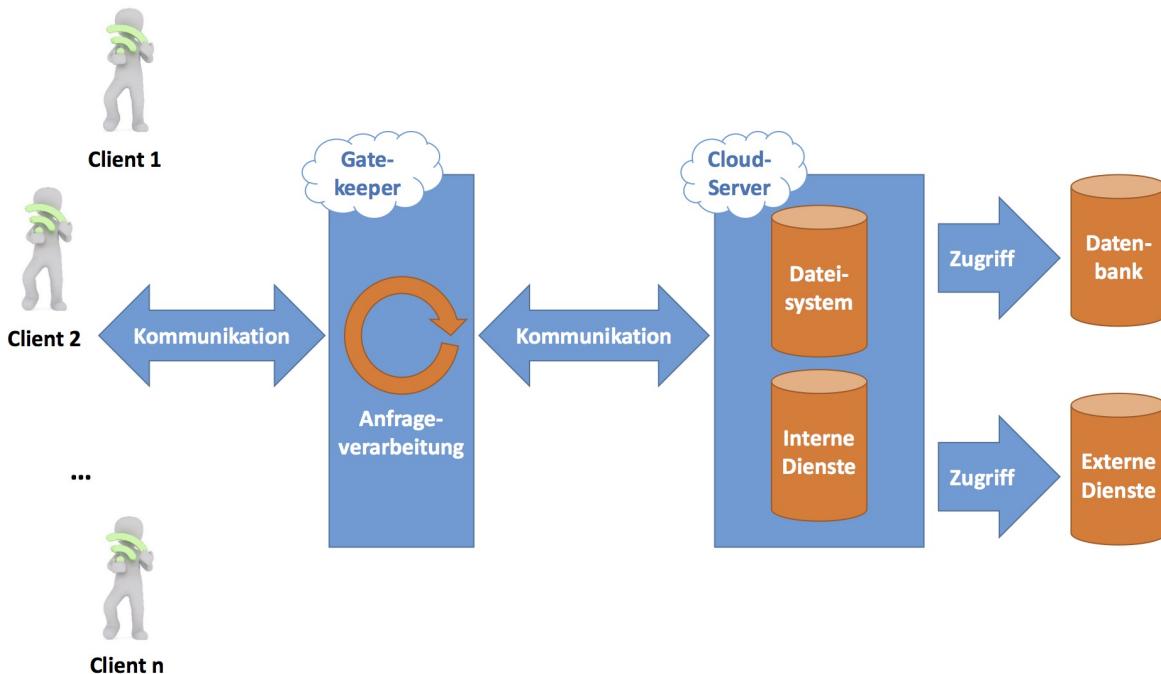
(Abbildung 1: Klassische Cloud-Architektur)

Potentielle Sicherheitslücken, die die beschriebene Architektur beinhaltet, lassen sich im Wesentlichen dem Bereich Security zuordnen. Aufgrund der direkten Kommunikationsmöglichkeit wird es Angreifern erleichtert, sich in das System einzuschleusen. Anfragen können beispielsweise so formuliert werden, dass der Zugriff auf eine dahinterliegende Datenbank ohne Umwege möglich ist. Das Auslesen oder Manipulieren von gesamten Informationsbeständen wäre eine denkbare Folge. Auch das Anstoßen sicherheitskritischer Dienste seitens eines Eindringlings kann ernsthafte Konsequenzen für unschuldige Benutzer sowie den Cloud-Betreiber nach sich ziehen. Neben der Sicherheit wird auch die Verfügbarkeit von Cloud-Diensten durch die direkte Kommunikationsmöglichkeit zwischen Client und Server gefährdet. Das kontinuierliche Senden von Anfragen mit dem Ziel einen Serverausfall herbeizuführen (Denial of Service-Angriff), ist effektiv möglich.

Lösung

Zur Lösung der vorangegangenen Problematik dient das sogenannte Gatekeeper Pattern. Die Idee, die diesem Architekturmuster zugrunde liegt, ist keinesfalls neu, sondern beruht auf zwei bestehenden Techniken. Zum einen wird auf das Prinzip der Firewall zurückgegriffen, welche Netzwerkzugriffe auf Systeme beschränkt. Des Weiteren dient das Fassade-Pattern aus der objektorientierten Programmierung als Grundlage. Der Kerngedanke dieses Entwurfsmusters besteht darin, eine Vielzahl von möglicherweise komplexen Schnittstellen vor dem Benutzer zu verbergen und ihm einen unkomplizierten Zugang zu einem System bereitzustellen.

In der Praxis kann der Gatekeeper als eine zusätzliche Serverinstanz verstanden werden, welche eine Vermittlerfunktion zwischen den Clients und dem Cloud-Server einnimmt (siehe Abb. 2). Anfragen seitens der Benutzer werden initial an den Gatekeeper geleitet und von diesem verarbeitet. Anschließend stellt der Gatekeeper eine Anfrage an den Cloud-Server, der ihm im Bestfall ein passendes Ergebnis liefert. Dieses übermittelt der Gatekeeper letztlich an den Benutzer und schließt somit die Anfrage ab.



(Abbildung 2: Anwendung des Gatekeeper Pattern)

Die Aufgabe des Gatekeepers besteht darin, die direkte Kommunikation von Client und Server zu unterbinden. Der Benutzer soll demnach keine Kenntnis vom Cloud-Server besitzen, sondern lediglich von der zwischengelagerten Schnittstelle. Des Weiteren trägt der Gatekeeper für eine geeignete Verarbeitung der Benutzeranfragen Sorge. Idealerweise werden nur solche Anfragen an den Cloud-Server weitergeleitet, die von rechtmäßigen Benutzern stammen bzw. keine Gefahr für die Sicherheit des Systems sowie die Integrität der Daten darstellen. Ggf. kann auch eine Überarbeitung der Anfragen dabei helfen, dass keine Sicherheitsrisiken eingegangen werden. Es ist hervorzuheben, dass der Gatekeeper ausschließlich Anfragen behandelt und übermittelt. Er hat beispielsweise keinen direkten Zugriff auf die Daten bzw. Dienste des Servers. Hierin liegt zugleich die Kernidee des Patterns. Sollte der Intermediär von Angreifern befallen werden, resultiert keine Gefahr für die Benutzer bzw. den Cloud-Betreiber. Der Eindringling besitzt mithilfe des übernommenen Gatekeepers keine Kontrolle über den Cloud-Server und hat demnach auch keinen Zugriff auf vertrauliche Ressourcen.

Nachteile

Die Anwendung des Gatekeeper Patterns hat positive Auswirkungen auf die Sicherheit von Cloud-Diensten. Jedoch ergeben sich auch Nachteile im Bezug auf Performance und Erreichbarkeit. Die Kommunikation über einen Vermittler hat zwangsläufig einen negativen Einfluss auf die Übermittlungsgeschwindigkeit von Informationen. Es ist nicht mehr ausreichend, Daten einmalig zu übermitteln. Dieser Vorgang, der auch das ver- und entpacken von Dateien beinhaltet, muss nun zweimal vollzogen werden. Insbesondere bei zeitkritische Anwendungen kann dies zu Problemen führen. Ein weiterer Nachteil ist die geringere Ausfallsicherheit, die sich durch den Einsatz eines zusätzlichen Rechenknotens ergibt. Sollte der Gatekeeper versagen, kann auch der Cloud-Server nicht mehr angesprochen werden. Um dieser Gefahr entgegenzuwirken, ist es sinnhaft, mehrere Instanzen des Gatekeepers zu betreiben. Hierdurch kann auch gleichzeitig ein erhöhtes Lastaufkommen bewältigt werden.

Health Endpoint Monitoring Pattern

Ziel des Patterns ist es funktionale Checks innerhalb einer Applikation auszuführen. Diese werden von externen Kontroll-Tools über entsprechende Zugangspunkte angestoßen und in bestimmten Intervallen abgerufen. Das Pattern hilft somit die korrekte Ausführung verschiedener Applikationen und Services zu gewährleisten.

Einsatzgebiet

- Webseiten und Webapplikationen auf Verfügbarkeit und korrekte Funktionsweise überwachen
- Middle-tier oder geteilte Services überwachen, um Fehlerquellen zu isolieren, die andere Applikationen beeinflussen könnten
- Zum Erweitern von bereits vorhandenen Überwachungselementen wie dem Logging und protokolierte Performanzkennzahlen einer Applikation

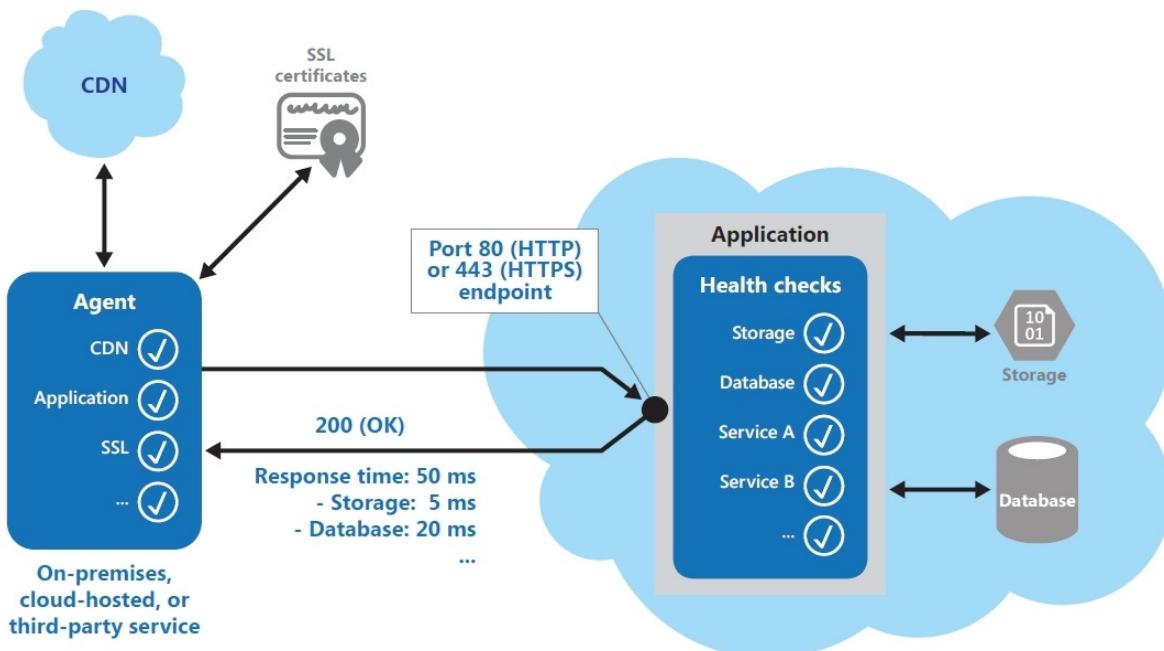
Problematik

Es ist üblich bzw. in vielen Fällen eine feste Voraussetzung, dass Webapplikationen oder Services auf ihre korrekte Ausführung überprüft werden. Bei einem Service in der Cloud kommen jedoch neue Abhängigkeiten und potenzielle Problemquellen dazu. Der eigene Service hängt potenziell von weiteren Services des Plattformanbieters oder anderen ab und über die Hostingumgebung hat man keine volle Kontrolle.

Auch hängt die Performance und Verfügbarkeit letzten Endes von der Hardware und der Netzwerkverbindung des Clouddienstleisters ab. Es ist also zwingend notwendig in regulären Abständen die Verfügbarkeit der eigenen Services in der Cloud zu überprüfen.

Das Pattern

Das Pattern besteht aus 2 wesentlichen Vorgängen bzw. Elementen. Zum einen aus dem Kontroll-Tool, das die Kontrollanfrage an einen frei konfigurierbaren Endpunkt einer Cloud Applikation stellt. Zum anderen die Cloud Applikation, die auf Anfrage ihre jeweiligen Checks durchführt. Diese können Kontrollen von weiteren genutzten Services inkludieren, sowie Kontrollen der Datenbankanbindungen und des Cloudspeichers. Ein entsprechender Response Code wird an das Kontroll-Tool zurückgesendet (siehe Abbildung 1).



(Abbildung 1: Übersicht des Patterns)

Das Kontroll-Tool analysiert anschließend den gesendeten Response Code gegen ein Set von frei konfigurierbaren Regeln, die festlegen wie welche Ergebnisse zu bewerten sind.

Zusätzlich können weitere Checks durchgeführt werden.

- Detaillierte Analyse des Inhalts der Antwort abseits des Codes, der z.B. weitere Information über teilweise fehlgeschlagene Tests geben könnte, selbst wenn der gegebene Response Code 200(OK) war
- Das Messen der Antwortzeit zur Überprüfung der Performance der Applikation und der Netzwerkgeschwindigkeit
- Auslaufende SSL Zertifikate abfragen
- Messen eines DNS Lookups auf die URL der Applikation, sowie Kontrolle der zurückgegebenen URL des DNS Lookups auf ihre Korrektheit

Es ist empfehlenswert die Tests von verschiedenen Kontroll-Tools von verschiedenen Standorten abfragen zu lassen um evtl. Unterschiede in der Verbindung ausfindig zu machen. Dies kann potenziell auch die Wahl beeinflussen wo eine Applikation deployt wird. Die Tests sollten auch gegen die Service-Instanzen von Kunden laufen, um zu überprüfen ob die Applikation für alle Kunden korrekt funktioniert. Wenn also ein Kunde seinen Cloudspeicher auf mehrere Standorte verteilt hat müssen alle Standorte kontrolliert werden.

Implementierung: Was man beachten sollte

- Wie genau sollten die **Antwort der Cloud Applikation** aussehen? Der minimalste Ansatz, ein simpler HTML Response Code, könnte potenziell nicht genug Informationen übermitteln.

- **Mehrere Endpunkte** für unterschiedliche Services innerhalb der Applikation konfigurieren, so sollte jeder Check (Datenbank, Storage..) einzeln angesprochen werden können oder über einen weiteren Endpunkt übergreifend. Für den jeweiligen Endpunkt können unterschiedliche Regeln (z.B. erwartete Antwortzeiten) definiert werden.
- Man kann den gleichen Endpunkt benutzen, der auch für den generellen Zugriff genutzt wird um über die jeweiligen Pfade auch direkt **funktionale Tests** auszuführen (z.B. einen User anlegen mit /applikation-url/create & /applikation-url/healthcheck/id für den regulären Check)
- Sollten zu viele **Ressourcen der Cloud Applikation** für die Checks verwendet werden, könnte dies die Usererfahrung beeinflussen. In der Regel können Logs über Fehler und Performanzzähler bereits genug aussagen und machen ausgiebige Performance Checks daher unnötig.
- **Sicherheit** der Endpunkte: Zugriff nur mit Authentifikation / 'versteckte' Endpunkte über unübliche Ports & verschiedene IP Adressen / Endpunkte so konfigurieren das sie bestimmte Informationen über die gewünschten Tests benötigen, andernfalls werden Anfragen abgewiesen
 - Zugriff auf gesicherte Endpunkte: Nicht alle Tools Unterstützen in ihre eingebauten health-verification Features auch Authentifizierung. Drittparty Anbieter wie Pingdom, Panopta, NewRelic oder Statuscake helfen hier.
- Auch die **Kontrolltools müssen getestet werden**, z.B. indem eine Cloud Applikationen ein festen OK Response auf einem Endpunkt sendet, den das Kontrolltool auslesen korrekt auslesen sollte.

Microsoft Azure

Applikationen die auf einer Microsoft Azure Umgebung gehostet werden, können bereits eingebaute Services der Plattform nutzen.

Der **Azure Management Service** bietet dabei die Möglichkeit bis zu 10 Regeln festzulegen, die durch einen Grenzwert definiert werden. Dieser kann sich z.B. auf die CPU Auslastung oder die Fehler pro Sekunde bei Anfragen beziehen. Bei Überschreitung der festgelegten Werte wird Alarm gegeben und der Nutzer per Email informiert.

Welche Konditionen überwacht werden können hängt von der Applikation und der Hostumgebung ab, nutzen jedoch in jedem Fall vom Nutzer definierte Endpunkte um die Kontrollen abzufragen.

Für virtuelle Maschinen oder Webapplikationen kann ein sogenannter **Traffic Manager** zusätzlich HTML Request stellen, die z.B. die Verfügbarkeit von Webseiten checken.

Index Table Pattern

Das Index Table Pattern dient zur Performance-Optimierung in Anwendungen, die mit großen Mengen an persistierten Daten arbeiten.

Da insbesondere Abfragen anhand anderer Werte als des Primärschlüssels besonders langsam werden können, bietet es die Möglichkeit, weitere Schlüssel anlegen zu können.

In relationalen Datenbanken kann dieses Verhalten meist mittels zusätzlicher Index-Spalten gelöst werden.

Das Index Table Pattern dient in anderen Fällen als eine Möglichkeit, diese Funktionalität nachzubilden.

Bei der Umsetzung gibt es drei unterschiedliche Strategien:

- **Vollständige Duplizierung**

Sämtliche Daten pro zusätzlichem Schlüssel in einer zusätzlichen Tabelle abgelegt. Dabei findet eine *vollständige Denormalisierung* statt. Problematisch bei häufigen Änderungen und im Bezug zur Speichermenge.

- **Normalisierte Index Tabellen**

Die neuen Tabellen enthalten jeweils nur den zusätzlichen Schlüssel und eine Referenz auf die ursprüngliche Tabelle, den *Fact Table*. Problematisch, da in jedem Fall zwei Abfragen erfolgen müssen.

- **Teilnormalisierte Index Tabellen**

Ähnlich den normalisierten Tabellen, allerdings werden häufig verwendete Daten dupliziert und nur für die fehlenden eine Anfrage an den *Fact Table* benötigt. Durch dieses Vorgehen erreicht man eine Balance zwischen den beiden anderen Fällen.

Darüber hinaus können auch Variationen umgesetzt werden, wie ein Schlüssel, der aus mehreren Feldern des Fact Table besteht, ein Ersatz für *Composite Keys*.

Ein Index Table kann auch verwendet werden, um den Zugriff auf Daten in Shards zu optimieren. Dabei wird der für das Sharding gehashte Primär Schlüssel als Referenz auf den geshardeten Fact Table verwendet. Der Index Table bleibt dabei zusammenhängend.

Probleme und Nachteile dieses Patterns sind zum einen der zusätzliche Aufwand, die Daten über mehrere Tabelle hinweg konsistent zu halten, bzw. die Notwendigkeit zweier Abfragen, um einen Datensatz auszulesen; je nach verwendeter Strategie.

Darüber hinaus muss auch bei Datenänderungen Konsistenz gewährleistet sein.

Sinnvoll ist die Anwendung des Patterns, wenn häufig Daten anhand von Nichtschlüsselattributen abgefragt werden. Dabei bestehen allerdings die folgenden Ausnahmen:

- **Häufig ändernde Daten**, durch die der Mehraufwand zum Pflegen der Index Tabellen die Performance-Gewinne überschreitet.
 - **Kleine Wertemenge** als neuer Schlüssel, wie z.B. Geschlechterinformationen (m/w)
 - **Unbalancierte Daten**, bei denen z.B. in 90% der selbe Wert enthalten ist. Eine Ausnahme dieser Ausnahme ist es, wenn häufig die verbleibenden 10% abgefragt werden.
-

Verwandte Entwursmuster sind:

- Data Consistency Primer
- Sharding Pattern
- Materialized View Pattern

Leader Election Pattern

Das Leader Election Pattern hat zum Ziel, die Aktionen mehrerer verteilter Komponenten, die an einer gemeinsamen Aufgabe arbeiten, zu koordinieren. Dafür wird aus allen Komponenten eine Komponente ausgewählt, die nun die Arbeit der anderen Komponenten managt. Somit wird verhindert, dass sich die Komponenten in ihrer Arbeit gegenseitig behindern, beispielsweise wenn versucht wird auf dieselbe Ressource zur gleichen Zeit zuzugreifen.

Problemhintergrund

Typischerweise ist es bei verteilten- oder Cloud-Anwendungen so, dass es mehrere Komponenten/Instanzen gibt, die beispielsweise denselben Programmcode beinhalten und auf gemeinsame Ressourcen zugreifen müssen. Dies kann bei skalierbaren Anwendungen der Fall sein, wenn beispielsweise jede Komponente für genau einen Benutzer zuständig ist, aber auf gemeinsame Ressourcen zugegriffen werden muss. Damit beim Zugriff auf diese gemeinsamen Ressourcen nicht unter anderem die Änderungen von einer Ressource von einer anderen stumpf überschrieben werden, muss dieser Zugriff koordiniert werden. Auch ist es möglich, dass jede Komponente nur einen kleinen Teil einer großen Aufgabe bearbeitet und am Ende alle Teilergebnisse zusammengeführt werden müssen. Da in beiden Fällen jede Komponente gleichgestellt ist (Peer), gibt es keine direkte Komponente die speziell für die Koordination zuständig ist.

Lösung

Um dieses Problem nun zu lösen, muss eine Instanz aus allen Instanzen gewählt werden, die nun die Leitung der anderen übernimmt. Da alle Instanzen dieselbe Code-Basis aufweisen, ist es somit möglich, dass jede beliebige Instanz zum Leiter ernannt wird. Der Prozess zur Wahl des Leiters muss also genau koordiniert werden, damit es nicht zu Fehlern kommt und es beispielsweise zwei oder mehr Instanzen gibt, die die Leitung übernehmen. Des Weiteren muss darauf geachtet werden, dass im Fehlerfall, beispielsweise bei einem Netzwerkausfall, ein neuer Leiter ernannt wird, wenn der alte seine Aufgabe nicht mehr absolvieren kann. Häufig wird dies so gelöst, dass jede Instanz den Leiter überwacht, beispielsweise durch einen Herzschlag-Mechanismus oder einfachem Abfragen des Leiters. Tritt dabei ein Fehler auf, wählen die verbleibenden Instanzen unter sich einen neuen Leiter aus. Die Wahl eines Leiters kann dabei beispielsweise folgendermaßen ablaufen:

- Es wird die Instanz mit der niedrigsten Prozess ID gewählt.
- Es existiert ein Mutex, der von allen versucht wird zu erreichen. Die Instanz die den Mutex zuerst bekommt wird zum Leiter. Dabei muss allerdings sichergestellt werden, dass der Mutex freigestellt wird, wenn der Leiter beispielsweise durch einen Fehler vom Netzwerk getrennt wird.
- Implementierung verschiedenster Leader Election Algorithmen, beispielsweise Bully- oder Ring-Algorithmus. Diese setzen voraus, dass jede Instanz eine eigene einzigartige ID besitzt und das die Instanzen untereinander kommunizieren können.

Probleme und Überlegungen

Folgende Punkte müssen bei der Anwendung dieses Patterns beachtet werden:

- Der Prozess zum Wahl eines Leiters muss Fehlerresistent sein.
- Es muss möglich sein den Ausfall eines Leiters festzustellen und entsprechend zu handeln, beispielsweise indem ein neuer Leiter gewählt wird.
- In skalierbaren Systemen ist es möglich, dass die Instanz die zum Leiter gewählt wurde beendet wird, wenn weniger Ressourcen benötigt werden und das System runter skaliert.
- Wird ein Mutex zur Wahl eines Leiters genutzt, ist der gesamte Prozess auch abhängig davon. Fällt der Zugriff zu dieser Mutex-Ressource aus, so kann auch kein neuer Leiter gewählt werden und das gesamte Peer System kann zusammenbrechen.
- Die manuelle Implementierung von einem der vorhandenen Leader Election Algorithmen bietet die größte Flexibilität und Anpassbarkeit.

Wann sollte es genutzt werden

Dieses Pattern sollte genutzt werden, wenn es in einem Verteilten (Cloud) System mehrere gleichgestellte Instanzen/Komponenten gibt, deren Zusammenarbeit sorgfältig koordiniert werden muss und es keinen direkten eigenständigen Leiter gibt. Der gewählte Leiter sollte dabei nur die Koordination der Arbeit der anderen Instanzen übernehmen und nicht noch selbst diese Arbeit ausführen, um zu verhindern, dass die Kommunikation verlangsamt wird. Dieses Pattern sollte nicht verwendet werden wenn:

- Es einen speziellen Prozess gibt, der die Aufgabe der Koordination übernimmt und somit immer als Leiter fungiert. Dies könnte beispielsweise durch ein Singleton Prozess umgesetzt werden, welcher im Fehlerfall neugestartet werden kann.
- Die Koordination durch deutlich einfachere Verfahren abgewickelt werden kann. Beispielsweise kann im Fall, dass mehrere Instanzen den Zugriff auf eine gemeinsame Ressource benötigen ein optimistischer oder pessimistischer Kontrollzugriff (Locking)

realisiert werden.

- Es entsprechende Drittanbieter Lösungen/Frameworks gibt, die diese Koordination der einzelnen Prozesse übernehmen können.

Materialized View Pattern

Context und Problem

Die Speicherung von Daten geschieht oft unter der Perspektive der reinen Datenspeicherung und berücksichtigt dabei nicht die Frage, wie die Daten gelesen werden sollen. Die Daten werden gemäß ihrer Typen gespeichert (z.B. Zahlen, Strings, usw.) und nach dem Schema des Datenmodells wie z.B. NoSQL oder relational.

Bei einer Datenabfrage ("Query"), welche nur eine Teilmenge der Daten benötigt, die jedoch über verschiedene Entitäten hinweg gespeichert wurden, müssen all diese Entitäten ausgelesen werden, um die gewollte Information zu erlangen.

Lösung

Das Materialized View Pattern beschreibt die Erzeugung von im Voraus erzeugten vorgefüllten Views. So eine View "materialisiert" die Daten in einer Form, welche am besten geeignet ist für das Ergebnis der Datenabfrage.

Eine materialisierte View ("materialized View") enthält nur die relevanten Daten für die entsprechende Datenanfrage, was einen schnelleren und einfacheren Informationsgewinn bedeutet. Sie können auch aktuelle, berechnete Werte enthalten, welche in der Abfrage definiert sind. Diese Views und die Daten die sie enthalten, können jederzeit auf Basis der Quelldaten wieder neu erstellt werden. Sie wird daher in keinem Fall direkt von einer Applikation aktualisiert, sondern eher als Cache behandelt. Wenn sich die Quelldaten ändern, muss die View jedoch aktualisiert werden, was entweder vom System automatisiert durchgeführt wird oder manuell.

Probleme und Überlegungen

Folgendes muss bei der Implementierung dieses Pattern beachtet werden:

- Wann und wie wird die View aktualisiert? (z.B. reagieren auf Veränderungen)
- Manche Systeme benötigen Materialized View Patterns. (z.B. bei Event Sourcing Patterns)
- Die Konsistenz der Daten bei Generierung und Aktualisierung der View
- Wo soll die View gespeichert werden? Sie muss nicht am selben Ort wie die

eigentlichen Daten liegen

- Ist die View nur zur Leistungsverbesserung da, kann sie z.B. im Cache gespeichert werden
- Materialized Views sollten ausgereizt werden (Berechnungen in die Datenabfrage einbinden)
- Wenn es möglich ist, Indexierung der View vornehmen um die Leistung zu steigern in relationalen Datenbanken

Wann wird dieses Pattern verwendet?

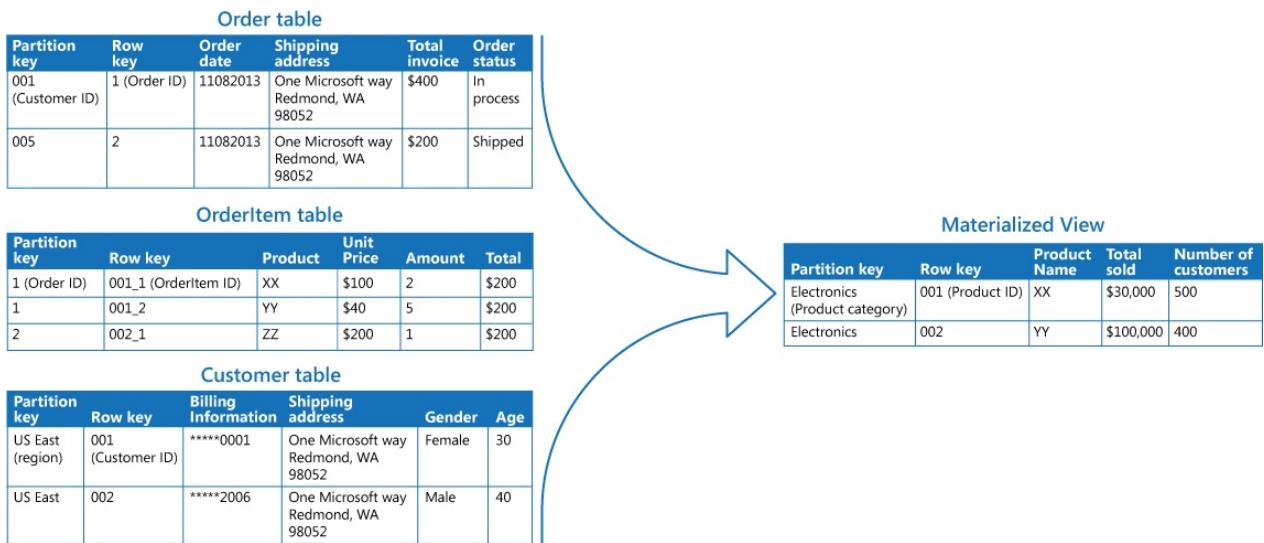
- Wenn Datenabfragen sehr komplex sind oder eine direkte Abfrage zu schwierig ist (z.B. unstrukturierte Daten, keine oder wenig Struktur vorhanden)
- Temporäre Views um die Leistung zu verbessern
- Lokal verfügbare Views, falls die Verbindung zu den Daten nicht besteht
- Vereinfachte Datenabfragen und Daten-Views zum experimentieren
- Sicherheit: Nur bestimmte Daten bestimmten Nutzern zur Verfügung stellen
- Als Brücke bei der Nutzung unterschiedlicher Datenspeichermethoden (z.B. eine Cloud zum Schreiben und eine relationale Datenbank für schnelle Datenabfragen)

Wann wird dieses Pattern nicht verwendet?

- Die Daten sind einfach abzufragen
- Die Quelldaten ändern sich sehr oft
- Konistenz ist ein wichtiger Faktor

Beispiel

Im folgenden Beispiel wird mittels anspruchsvollen Datenabfragen eine "materialized View" erstellt. Über verschiedene Entitäten wird eine View erstellt, die die gesamten Verkaufszahlen elektronischer Produkte enthält, zusammen mit der Anzahl der Kunden. Nutzer können mit Hilfe dieser View einfach bestimmte Resultate abfragen oder in andere Datenabfragen miteinbeziehen.



(Abbildung 1: Beispielhafte Anwendung um eine Zusammenfassung der Verkäufe darzustellen. Quelle: <https://docs.microsoft.com/en-us/azure/architecture/patterns/materialized-view>)

Ähnliche Patterns und Richtlinien

Folgende Patterns und Richtlinien könnten auch von Interesse sein:

- Data Consistency Primer
- Command and Query Responsibility Segregation (CQRS) Pattern
- Event Sourcing Pattern
- Index Table Pattern

Pipes and Filters Pattern

Eine Aufgabe sollte aus mehreren einzelnen Schritten bestehen. Diese einzelnen Schritte sollten wiederwendbar sein. Diese Wiederverwendbarkeit verbessert die Leistungsfähigkeit, Skalierbarkeit und Wiederverwendbarkeit der einzelnen Schritte deutlich, da die einzelnen Schritte unabhängig und skalierbar untereinander sind.

Problem

Für die Entwicklung einer Anwendung kann man einzelne Verarbeitungsschritte zu einem monolithischen Modul zusammenfassen. Aber wenn die gleichen Verarbeitungsschritte in mehreren verschiedenen Modulen vorkommen, dann verbessert sich der Code nicht. Der Code wird dupliziert statt reduziert, da der Code nicht wiederverwendet wird, sondern nur kopiert wird. Der Code wird nicht gut optimiert, da der Code in verschiedenen Modulen dupliziert wird und keine Regeln der Wiederverwendbarkeit genutzt wurden. Die Anwendung ist dadurch nicht besonders skalierbar. Wenn sich einige Verarbeitungsschritte in den Modulen ändern z.B. der Ablauf der Schritte ändert sich oder weitere Schritte werden hinzugefügt, dann muss der Code an mehreren Stellen überarbeitet werden.

Lösung

Eine Lösung für diese Probleme wäre, wenn man die Verarbeitungsschritte zu einem Prozess zusammenfasst. Dieser Prozess verarbeitet die einzelnen Schritte linear. Die Verarbeitungsschritte erledigen immer eine Aufgabe. Jeder Verarbeitungsschritt empfängt Daten und sendet Daten an den nächsten Verarbeitungsschritt weiter. Die einzelnen Verarbeitungsschritte sind Komponenten. Diese Komponenten können einfach ausgetauscht werden, wenn sich der Prozess verändert. Außerdem wird duplizierter Code vermieden, da der Code in einer Komponente gekapselt ist. Ändert man den Code in der Komponente, so würgt sich diese Änderung auf alle Prozesse aus, wo diese Komponente verwendet wird. Das Prinzip führt zu einer besseren Wiederverwendung des Codes (siehe Abbildung 1).

Die Verarbeitungsgeschwindigkeit des Prozesses hängt von den einzelnen Komponenten ab. Wenn einige Komponenten zu langsam sind, dann kann das zu einer Verlangsamung des ganzen Prozesses führen. Aus diesem Grund sollte man parallele Prozesse verwenden, um die langsamen Komponenten zu verteilen. Diese Verteilung erhöht die Verarbeitungsgeschwindigkeit der Anwendung. Gerade bei Serveranwendungen führt, dass zu einer besseren Performance für das gesamte System. Die Komponenten (oder Filters)

sind untereinander unabhängig und können beliebig skaliert werden. Sie eignen sich gut für Cloud-Server, da die Komponenten auf verschiedene Maschinen verteilt werden können. Rechenintensivere Komponenten können auf leistungsfähigeren Maschinen verteilt werden als weniger rechenintensivere Komponenten, dass spart Kosten und führt zu einer besseren Performance für die Serveranwendung. Man kann von überall auf der Welt auf die Komponenten und Prozesse (oder auch Pipeline) zugreifen, da sich diese in der Cloud befindet.

Der vorige Filter kann seine Ergebnisse an den nächsten Filter in der Pipeline weitergeben, bevor der vorige Filter zu Ende ist. Es können so mehrere Filter parallel arbeiten. Sollte ein Filter oder eine Maschine von der Cloud ausfallen, dann kann die Pipeline diesen Fall kompensieren. Die Pipeline leitet die Verarbeitungsschritte an einem anderen Filter weiter. Dieser Filter erledigt dann diese Aufgaben und gibt das Ergebnis an der Pipeline zurück. Das System ist dadurch sehr gut gegen Systemausfälle geschützt, da bei Ausfällen einer oder mehrerer Filters nicht diese ganze Pipeline mit ausfällt. Wenn man verteilte Transaktionen entwickeln will, dann kann man das „Pipes and Filters Patterns“ mit dem „Compensating Transaction Pattern“ kombinieren. Die verteilte Transaktion wird in mehrere Aufgaben geteilt und jede Aufgabe bekommt einen Filter. Der Filter benutzt das „Compensating Transaction Pattern“.

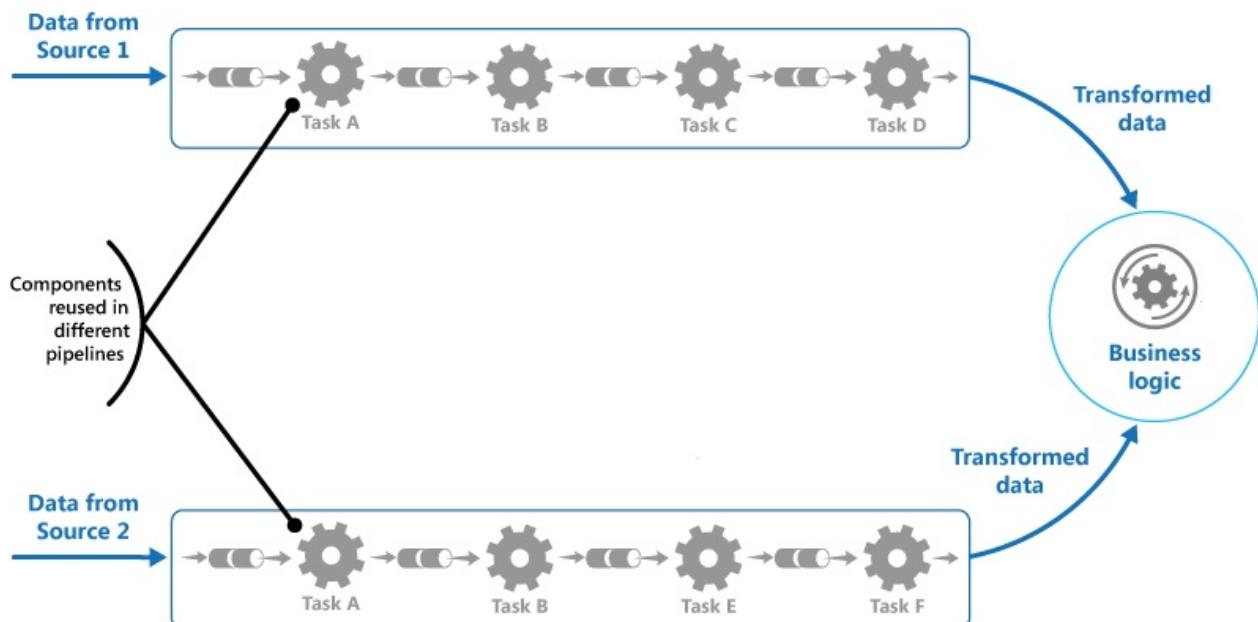


Abbildung 1: Grafische Darstellung des Patterns.

Benutzung

Wenn man das „Pipes and Filters Patterns“ benutzen will, dann sollte man darauf achten, dass die Anwendung nicht komplexer dadurch wird, immer noch zuverlässig läuft, das keine Idempotenz entsteht, keine wiederholten Nachrichten verschickt werden und der Zustand

und Kontext der Filter erhalten bleibt. Eine Komplexität kann dadurch entstehen, wenn man die Aufgaben an verschiedene Server verteilt. Die Zuverlässigkeit muss sicherstellen, dass keine Daten zwischen den einzelnen Filters verschwinden. Filters sollten Idempotenz sein, da sonst Ergebnisse von verschiedenen oder gleichen Filtern mehrfach auftreten könnten. Diese Daten würden dann mehrfach verarbeitet werden, dass führt zu Leistungseinbrüchen der Serveranwendung. Die Pipeline muss überprüfen, ob doppelte Nachrichten an die Filters geschickt werden und gegebenenfalls diese entfernen. Jeder Filter muss seinen Zustand speichern und seinen eigenen Kontext besitzen, da sie getrennt voneinander arbeiten.

Anwendungsbeispiele

Zum Beispiel geeignet für folgende Aufgaben:

- Ein Problem in mehreren unabhängigen Schritte lösen
- Die Verarbeitungsschritte der Serveranwendung sind unterschiedlich skalierbar
- Wenn etwas Flexibel ist, dann kann man die Verarbeitungsschritte anders anordnen und untereinander verändern
- Die Verteilung der Serveranwendung und der Verarbeitungsschritte auf viele verschiedenen Servers verbessert die Leistung für die Serveranwendung
- Wenn ein System vor Ausfällen geschützt werden soll

Es ist nicht so gut geeignet für folgende Aufgaben:

- Wenn die Verarbeitungsschritte nicht unabhängig voneinander sind, sondern zu einem großem Zusammenhang gehören.
- Wenn die Speicherung der Zustandsinformation Probleme mit der Datenbank verursachen würde

Bei der Implementierung sollte eine Warteschlange für die Pipeline verwendet werden. Die Filter bekommen die Daten und verarbeiten die Daten danach. Wenn die Daten verarbeitet wurden, dann werden die Daten an dem nächsten Filter in der Warteschlange weitergegeben. Bis das Ende der Warteschlange erreicht wurde. Der Anfang macht das erste Objekt der Filter Klasse in der Warteschlange.

Priority Queue Pattern

Das Priority Queue Pattern arbeitet Abfragen mit einer höheren Priorität schneller ab, als Abfragen mit einer geringen Priorität.

Konzept und Problem

Beispielsweise bei der Nutzung einer Cloud werden Abfragen an den Server über eine Message-Queue verwaltet. Da einige Abfragen so zeitnah wie möglich behandelt werden müssen, um dem Nutzer das Gefühl von "Echtzeit" zu vermitteln müssen diese durch Priorisierung vorgeschaltet werden.

Lösung

Um das Konzept von First-in First-out zu modifizieren bekommt jede eingehende Abfrage eine Priorität zugewiesen. Die neue Abfrage wird dann hinter das letzte Element der gleichen Priorität gesetzt. Somit hat man mehrere First-in First-out Queues innerhalb einer nach Prioritäten sortierten Queue.

Sollte ein System das Prinzip der Priority Queue nicht unterstützen, so kann durch die Erstellung mehrerer Queues für jede einzelne Priorität Abhilfe geschaffen werden.

Demnach werden erst alle Abfragen aus der Queue mit der höchsten Priorität abgearbeitet. Ist diese leer, so wird auf die nächsthöhere Priorität eingegangen.

Der hierbeschriebene single-pool-Ansatz führt dazu, dass eine Abfrage mit einer niedrigen Priorität womöglich nie abgearbeitet wird, da neue Abfragen mit einer höheren Priorität dazu kommen. Abhilfe soll hier der multi-pool-Ansatz sein, welcher die niedriger priorisierten Abfragen in Abhängigkeit seiner verfügbaren Ressourcen auf jeden Fall abarbeitet.

Die Nutzung der Priority Queue hat den Vorteil Business-Anforderungen welche die Priorisierung betreffen, wie beispielsweise die Bevorzugung von Pro-Kunden gegenüber normalen Kunden durchsetzen zu können.

Probleme und Überlegungen

Die Prioritäten müssen in Abhängigkeit zur Lösung implementiert werden. Wenn alle Abfragen die höchste Priorität haben, so muss unter Umständen auch hierbei lange auf eine Antwort gewartet werden.

Bei einer hohen Auslastung der Queue muss für den single-pool-Ansatz eine Lösung für die niedrig Priorisierten Abfragen gefunden werden.

Beim multi-pool-Ansatz auf einer single-pool-Basis muss ein Algorithmus zum Abrufen der korrekten Abfragen in den unterschiedlich priorisierten Queues implementiert werden.

Die Raten der behandelten niedrig- und hochpriorisierten Abfragen müssen überwacht und verglichen werden, um die Priorisierungen möglichst effizient anzupassen.

Abschließend kann man sagen, dass die Priority Queue bei Abfragen mit verschieden gewichteter Priorität oder bei Abfragen von Kunden mit verschieden gewichteter Priorität zum Einsatz kommen sollte.

Queue-Based Load Leveling Pattern

Das **Queue-Based Load Leveling Pattern** besteht darin, dass man Services um eine asynchrone Queue erweitert, welche die Anfragen der Tasks entgegennimmt und dem Service weiterreicht. Das soll verhindern, dass bei zu vielen Anfragen der Tasks an den Service dieser überlastet und seine Funktionalität verzögert oder gar nicht erfüllen kann. Das Pattern erhöht somit die Verfügbarkeit und Verlässlichkeit der Services.

Kontext und Problem

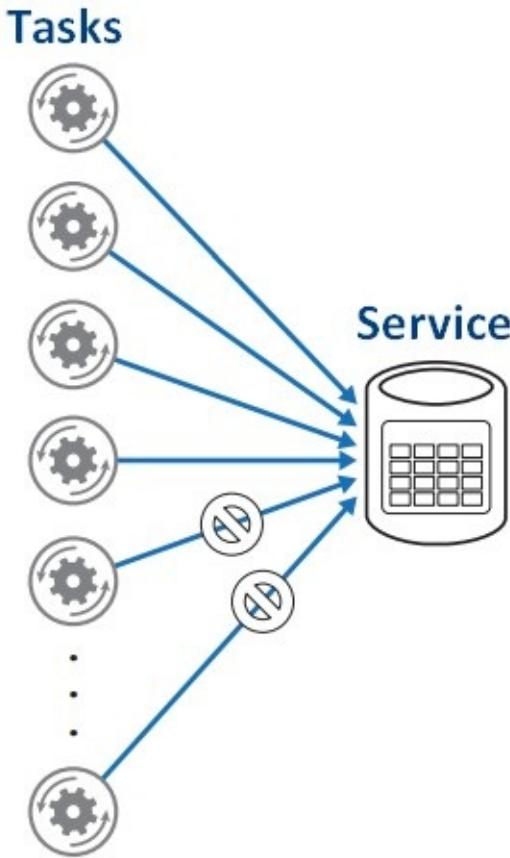
Viele Programme in der Cloud basieren darauf, dass sie Tasks ausführen, welche Services aufrufen. Wenn aber zu viele Tasks auf denselben Service zur gleichen Zeit zugreifen wollen, kann dies zur einer Überlastung des Service führen, was sich in Verzögerungen oder nicht Erreichbarkeit äußern kann (siehe Bild).



(Quelle: Cloud Design Patterns)"

Lösung

Der Einsatz einer asynchronen Queue kann zwar die Verzögerung der Anfrage der Tasks nicht vollständig verhindern, sorgt aber dafür, dass die Anfragen nicht unbeantwortet bleiben.



(Quelle: Cloud Design Patterns)"

Die Anfragen der Tasks werden in die Queue gesteckt und geordnet nach Zeitpunkt des Eintreffens abgearbeitet.

Das bietet folgende Vorteile:

- Die Verfügbarkeit der Services wird maximiert, da Verzögerungen weniger Auswirkungen auf die Funktionalität haben. Selbst wenn der Service temporär nicht erreichbar ist, können die Anfragen nachgearbeitet werden.
- Der Service kann skaliert werden, da je nach Anforderung die Anzahl der Queues und Services angepasst werden können.
- Es unterstützt bei der Kostenkontrolle bzgl. Ressourcen, da durch die Queue die Anzahl an Services anstatt für den Worst Case nur ausreichend für den Durchschnittsarbeitsfluss sein muss. Das Pattern eignet sich ideal für Applikationen, bei denen hohe Zugriffszahlen in kurzer Zeit entstehen könnten. Es wäre entsprechend nicht effektiv, dies bei Applikationen einzusetzen, bei denen das Gegenteil der Fall ist.

Man sollte folgende Punkte beachten, sofern man dieses Pattern implementieren will:

- Es ist von Vorteil, eine Logik für die Applikation zu entwickeln, welche die Abarbeitungsrate der Services kontrolliert, damit der Service nicht überlastet. Es sollten grundsätzlich Arbeitsflussspitzen vermieden werden. Zur Sicherstellung kann das System unter realistischen Arbeitsfluss getestet werden, um bei Unstimmigkeiten die

Anzahl der Queues und Services anzupassen.

- Die Queue funktioniert nur in einer Richtung. Wenn der Task eine Rückmeldung erwartet, müssen weitere Mechanismen implementiert werden, die dies sicherstellen (z.B. [Asynchronous Messaging Primer](#)).

Retry Pattern

Dieses Muster (Retry Pattern) ermöglicht, dass eine Anwendung erwartete und temporäre Fehler behandelt. Wenn eine Anwendung versucht, sich mit einem Service oder Netzwerk zu verbinden, wiederholt sie die Operation, weil die Ursache des Fehlers vorübergehend (transient) sein kann.

Problem

Wenn eine Anwendung mit Elementen in einer Cloud kommuniziert, können vorübergehende Fehler auftreten: der kurzzeitige Untergang der Netzwerkkonnektivität, die kurzzeitige Nichtverfügbarkeit eines Services... Die Aktion, die diese Fehler auslöst, kann erfolgreich sein, wenn sie mit einer Zeitverzögerung wiederholt wird.

Lösung

Eine Anwendung kann diese Fehler folgendermaßen behandeln:

- Wenn der Fehler nicht vorübergehend ist (z.B. Authentifizierungsfehler), sollte die Anwendung die Operation abbrechen und eine 'Exception' werfen.
- Wenn der bestimmte Fehler ungewöhnlich oder selten ist, sollte die Anwendung die Operation sofort wiederholen. (Wahrscheinlich wird der Fehler nicht erneut auftreten.)
- Wenn der Fehler ein 'busy' Fehler ist, sollte die Anwendung warten, bevor die Operation wiederholt wird. Wenn die Operation nach einer bestimmten Anzahl von Versuchen immer noch nicht erfolgreich ist, sollte die Anwendung den Fehler als eine Ausnahme (Exception) behandeln. Es ist hilfreich Fehler, die mit Retry lösen kann, in einem log detailliert aufzuzeichnen.

Was es zu beachten gilt

- Die Retry policy sollte an die Art der Anwendung und des Fehlers angepasst werden. Wenn die Operation für die Anwendung nicht essentiell ist, kann es manchmal für die Anwendung besser sein nicht die Operation zu wiederholen, sondern abzubrechen.
- Wenn man zu viel und zu schnell erneut versucht, kann man sowohl den 'busy' service, als auch die Anwendung überlasten.
- Nach einer Vielzahl von Wiederholungen kann man erstmal einen Fehlerbericht

ausgeben, dann eine Weile warten und erst dann ein oder zwei Mal erneut die Operation versuchen.

- Es ist wichtig, dass die Zeitverzögerung zwischen erneuten Anfragen an die Art des Fehlers angepasst ist.
- Retry Code muss die Zuverlässigkeit und Performance des gesamten Codes nicht beeinträchtigen. Man muss den Retry Code vielfältig testen.
- Es soll keine verschachtelten Retry Codes geben. Ein Retry Code ist z.B. verschachtelt, wenn eine Aufgabe, die Retry Code hat, eine andere Aufgabe, die auch Retry Code hat, aufruft.
- Man sollte versuchen herauszufinden, ob die Fehler lang andauernd sind. In diesem Fall sollte man den Fehler als Ausnahme behandeln.

Wann sollte man Retry Pattern benutzen?

Retry Pattern sollte benutzt werden:

- Wenn bei einer Anwendung, die mit einem fernen Service interagiert, vorübergehende Fehler auftreten können, sollte man Retry Pattern benutzen.

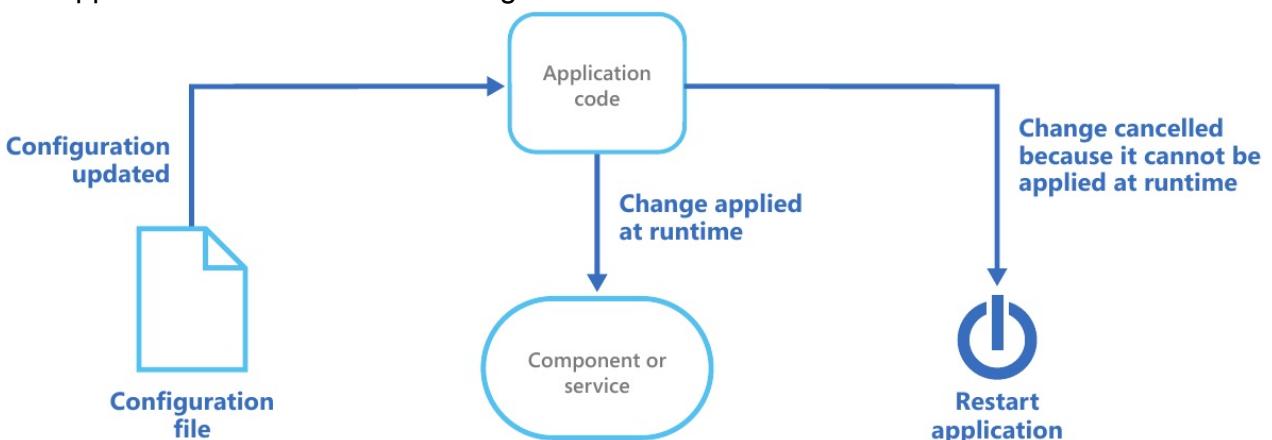
Retry Pattern sollte nicht benutzt werden:

- Wenn ein Fehler lang andauernd ist.
- Für nicht vorübergehende Fehler
- Wenn ein 'busy' Fehler bei einem bestimmten Service häufig ist.

Runtime Reconfiguration Pattern

Das Runtime Reconfiguration Pattern soll eine Rekonfiguration einer Applikation während der Laufzeit ermöglichen, so dass der Nutzer oder Kunde nur eine minimale Downtime oder daraus resultierende Unterbrechung der Applikation erfährt.

Dieses Pattern ist erstmal abhängig von den Features des Hosts der Applikation. Diese könnten aus dem Hochladen einer Konfigurationsdatei oder dem Zugriff über ein Administrationsportal oder einer API bestehen. Der Applikationscode kann dabei die Änderung an der Konfiguration untersuchen und sie auf die dazu gehörigen Komponenten der Applikation anwenden. Die Komponenten müssen dabei die Änderung erkennen und sie übernehmen. Wenn die Komponenten das nicht während der Laufzeit können, ist ein Neustart der Applikation dafür notwendig. Dies kann dadurch geschehen, dass der Host eine Änderung erkennt und der Applikation mitteilt sich neu zu starten oder es muss ein Code implementiert werden, der Änderungen an der Einstellung erkennt und gegebenenfalls die Applikation zum Neustarten zwingt.



(Abbildung 1 - Schema eines Runtime Reconfiguration Patterns)

Viele Umgebungen arbeiten mit Events als Antwort auf eine Konfigurationsänderung. Die Umgebungen, die es nicht tun, erfordern ein zyklisches Absuchen nach Änderungen der Konfiguration und eine Anwendung dessen falls notwendig. Es kann dabei auch nötig sein, das Datum und die Zeit einer Konfigurationsdatei zu vergleichen, um eine eventuelle neue Version festzustellen. Alternativ kann die Umgebung auf bestimmte Änderungen reagieren. Zum Beispiel könnte beim Auftreten eines Runtime-Error ein automatisches Sammeln von zusätzlichen Informationen erfolgen.

Folgende Punkte sollte man beim Implementieren dieses Patterns beachten:

- Die Konfigurationsdateien sollten abgetrennt von einer entwickelten Applikation gespeichert werden.
- Wenn Konfigurationsänderungen nicht automatisch erkannt werden, sollte man einen

alternativen Mechanismus zum Erkennen und Anwenden von Änderungen erstellen.

- Wenn das zyklische Absuchen nach Änderungen erforderlich ist, sollte man abwägen in welchen Abständen man dies tut.
- Wenn es mehr als eine Instanz der Anwendung gibt, sollte man beachten, dass die Änderung an alle Instanzen der Anwendung nach einer gewissen Zeit kommuniziert wird.
- Wenn eine Konfigurationsänderung einen Neustart des Hostsystems erfordert, sollte man die Art der Änderung identifizieren. Es könnte eine Änderung sein, die einen automatischen Neustart des Systems erfordert oder eine, die die Verantwortung dem Administrator zuschiebt, um die Applikation je nach Zeit oder Performance des Systems neu zu starten.
- Es ist abzuwägen wie man die Konfiguration zurücksetzen will, für den Fall, dass die Änderung Fehler verursacht hat.
- Auch die Herkunft der Konfigurationseinstellung sollte beachtet werden, weil es die Performance der Applikation beeinflussen könnte. Zum Beispiel ein Fehler, der aufgetreten ist, wenn ein externer Speicher nicht verfügbar ist.
- Das Cachen kann dabei helfen Verzögerungen zu reduzieren, wenn Komponenten wiederholt auf Konfigurationen zugreifen müssen.

Dieses Pattern ist hilfreich für:

- Applikationen, welche unnötige Downtimes vermeiden müssen.
- Umgebungen, die automatisch Events auslösen, wenn die Hauptkonfiguration sich geändert hat.
- Applikationen bei denen sich die Konfigurationen oft ändern und die Änderung nicht den Neustart der Applikation benötigen sollte.

Nicht hilfreich wäre es für Laufzeitkomponenten, die so aufgebaut sind, dass sie die Konfiguration nur in ihrer Startzeit laden und der Aufwand diese Komponenten anzupassen im Vergleich zum Neustarten und einer kurzen Downtime nicht gerechtfertigt ist.

Schedular Agent Supervisor Pattern

Das Pattern dient dazu Aufgaben und Abfolgen von Aufgaben in der Cloud auszuführen und zu Koordinieren.

Cloud bedeutet dabei, dass mehrere Systeme am Ausführen von Aufgaben beteiligt werden. Aufgaben sind dabei Prozesse die auf andern Systemen ausgeführt werden. Die Aufgaben werden nicht direkt innerhalb der Komponenten des Patterns eingebettet sondern durch z.B. URLs in externen System aufgerufen.

Durch den Einsatz des Patterns soll insbesondere im Fehlerfall, beim Absturz des Supervisors oder beim Fehlschlagen einzelner Aktionen, ein definierter Zustand wiederhergestellt werden (Self-Healing).

Das Pattern beinhaltet 3 Zuständigkeiten (Actors):

- Scheduler
- Agent
- Supervisor

Durch den **Scheduler** wird die Reihenfolge der Ausführung der Aufgaben festgelegt. Weiterhin wird der derzeige Zustand der Aufgabe festgehalten, welche Teilschritte ausgeführt wurden und in welchem Zustand sich die derzeitige Teilaufgabe befindet. Der Supervisor speichert den Zustand der aktuellen Ausführung in einer Datenbank dem "State-Store".

Dieser State-Store bietet ein Protokoll über den Zustand der der aktuellen Aufgabe, er bietet aber auch die Möglichkeit die Ausführung bestimmter Aufgaben nach einem Absturz oder Neustart fortzusetzen.

Der **Agent** koordiniert den Aufruf der konkreten Aufgabe. Er Prüft den Zustand einer Aufgabe und kann den Scheduler anweisen bestimmte Aktionen erneut auszuführen. Er entspricht im wesentlichen dem Proxy-Pattern mit der Erweiterung von Timeouts und Kommunikation mit einem bekannten Scheduler. Der Agent ist möglich generisch zu halten er sollte keine Kenntnis des aktuellen Geschäftsvorgangs haben.

Der **Supervisor** benutzt den Agent um den Status einer Teilaufgabe abzufragen und dem Scheduler zugänglich zu machen.

Im Kontext von Cloud-Anwendungen ist die Koordinierung der Aufgaben besonders wichtig, insbesondere ist es wichtig, dass Aufgaben nicht mehrfach ausgeführt werden. Um dies sicherzustellen muss zusätzlich zum Scheduler Agent Supervisor Pattern auf das Leader

Election Pattern zurückgegriffen werden. Pro Aufgabe muss ein Scheduler ausgemacht werden der für diese spezifische Aufgabe verantwortlich ist.

Um korrekte Ergebnisse der ausgeführten Aktionen zu erhalten sollten diese Indempotent sein. Sollte dies nicht möglich sein muss ein Mechanismus für Transaktionen (Ein Rollback im Fehlerfall muss gewährleistet sein) innerhalb der Aufgaben oder der Aktoren integriert sein.

Sharding Pattern

Die Idee ist es große Datenmengen in mehrere horizontale "Splitter"(Shards) aufzuteilen, um die Skalierbarkeit zu verbessern.

When to use

Wenn die Daten die Ressourcen einer einzelnen Datenbank übersteigen werden. Um die Performance zu steigern, indem die Zugriffskonflikte minimiert werden.

Problemstellung

Eine Datenbank auf einem einzigen Server wird sich früher oder später mit den Folgenden Themen befassen müssen: Speicherplatz Rechenleistung Netzwerkbandbreite Datenverteilung

Diese Punkte durch "aufrüsten" auszubauen, verschiebt lediglich das Problem in die Zukunft.

Sharding als Lösung

Die Datenbank wird in mehrere Splitter aufgeteilt. Jeder Splitter verfügt über das gleiche Schema, enthält jedoch nur eine individuelle Menge der Gesamten Datenmenge.

Für das Aufteilen von Daten auf die einzelnen Splitter werden "shard keys" oder "partition keys" verwendet. Dieser Schlüssel muss statisch sein.

Sharding organisiert die Daten, die Logic sorgt bei einem Zugriff der Applikation dafür, dass die angeforderten Daten aufgefunden werden.

Vorteile

Jederzeit erweiterbar, durch das Hinzufügen von weiteren Splittern. Zugriffskonflikte können reduziert und Performance gesteigert werden, durch ein Ausbalancieren des Workloads der einzelnen Splitter. In einer Cloudlösung können die Splitter "nah" am User platziert werden.

Nachteile

Daten der Splitter müssen ausgeglichen werden, um für eine hohe Performance zu sorgen.

Sharding Strategies

Für die Vergabe von "shard keys" und das Verteilen der Daten, gibt es 3 gängige Strategien, welche im Folgenden skizziert werden.

The Lookup strategy

Hier liegen alle benötigten Daten (für einen Teil einer Abfrage) auf einem einzigen Shard.
Vorteile: Mehr Kontrolle über die Konfigurationen der Shards. Nachteile: Overhead durch das aufsuchen mehrere Speicherorte(Shards)

The Range strategy

Häufig zusammen abgefragte Daten, bspw. in einem Query, werden zusammen in einem Splitter gespeichert. Vorteile: Einfach zu implementieren. Einfaches Datenmanagement
Nachteile: Keine optimale Balance zwischen den Shards. Rebalancing ist schwer.

The Hash strategy

Aus mehreren Attributen werden Hashes gebildet, welche abgefragt werden können. Durch diese Strategie soll ein Balance der Zugriffe unter den Shards erreicht werden. Dies ist vor allem sinnvoll in Umgebungen, mit sehr vielen Zugriffen. Vorteile: Hashermittlung kann Overhead erzeugen. Rebalancing ist schwer.

Static Content Hosting Pattern

Im Folgenden wird das Static Content Hosting Pattern beschrieben, welches dazu dient Ressourcen effizienter zu verwalten, wenn statische Elemente von Servern zur Verfügung gestellt werden.

Problemstellung

Web-Applikationen beinhalten im Normalfall einige Elemente mit statischem Inhalt. Dieser statische Inhalt kann in Form von HTML-Seiten und anderen Ressourcen, wie Bildern oder Dokumenten vorliegen. Clients können entweder über den Aufruf der HTML-Seite oder über einen direkten Download auf diese statischen Elemente zugreifen. Dies führt dazu, dass der Webserver Anfragen zum Bereitstellen statischer Inhalte verarbeiten muss und somit Rechenzeit nutzt, um die Inhalte zu instanziieren. Diese Rechenzeit lässt sich minimieren, indem statische Inhalte ausgelagert werden.

Abhilfe

In vielen cloudbasierten Systemen ist es möglich die zur Laufzeit erstellten statischen Instanzen zu minimieren, indem statische Instanzen auf einem Speicherdiensst zum Abruf bereitgestellt werden. Die Kosten für Speicherdiensste sind im Gegensatz zu den Kosten für Rechenzeit auf cloudbasierten Systemen wesentlich geringer. Das folgende Schaubild bietet einen Überblick in das Konzept des Static Content Hosting Pattern:

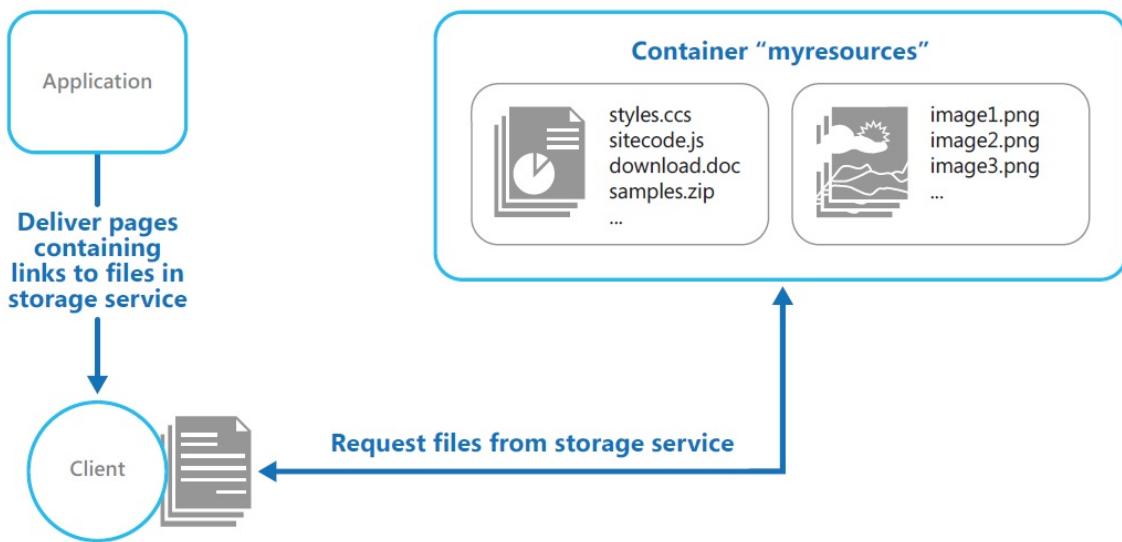


FIGURE 1
Delivering static parts of an application directly from a storage service

Herausforderungen und Überlegungen

Beim Implementieren dieses Design-Patterns ist folgendes zu beachten:

- Der gehostete Speicherdiensst muss einen HTTP Endpunkt auflösen können, damit Benutzer auf den statischen Inhalt zugreifen können. Und Einige Speichergeräte unterstützen HTTPS, falls statische Ressourcen SSL erfordern.
- Für eine maximale Performance und Verfügbarkeit sollte ein "Content Delivery Network" in Betracht gezogen werden. Dieses nutzt eine Zwischenspeicherung der Inhalte an mehreren Datenzentren und ist dadurch allerdings mit zusätzlichen Kosten verbunden.
- Durch das Duplizieren der Daten an verschiedenen Datenzentren kann sich die IP-Adresse ändern, die URL bleibt jedoch erhalten.
- Die Komplexität der Applikation steigt, da Daten gestreut vorzufinden sind und damit ist auch ein erhöhter Aufwand für das Verwalten der Daten verbunden.
- Speicherdiensste
- Die Rechte für den öffentlichen Zugriff in "Storage Container" müssen korrekt verwaltet werden um bspw. unbefugte Uploads zu verhindern. Um anonyme Zugriffe zu verhindern sollte ein Valet Key oder ein Valet Token genutzt werden: [Valet Key Pattern](#).

Wann sollte das Pattern genutzt werden?

Das Pattern ist ideal geeignet, um:

- Kosten für das Hosting von Webseiten und Applikationen mit statischen Inhalten zu minimieren.

- Kosten für das Hosting von Webseiten zu minimieren, wenn diese ausschließlich aus statischen Inhalten bestehen.
- statische Inhalte für Applikationen bereitzustellen, welche auf anderen Hosts (oder "on-premises") betrieben werden.
- Inhalte an mehreren geographisch verteilten Standorten mit Hilfe eines "Content Delivery Networks" zu speichern.
- Kosten und Nutzung der Bandbreite zu überwachen, da sich Host- und Laufzeitkosten von Speicherkosten der statischen Inhalte trennen lassen.

Das Pattern ist nicht geeignet, wenn:

- Applikationen die statischen Inhalte manipulieren, bevor die Inhalte an Clients ausgeliefert werden. Beispielsweise wenn Applikationen einen Zeitstempel auf ein Dokument aufsetzen, direkt bevor es heruntergeladen wird.
- nur sehr wenig statischer Inhalt zu verwalten ist.

Throttling Pattern

Problemstellung

Die Auslastung einer Cloud unterliegt zeitlichen Schwankungen. Sie ist abhängig von der Anzahl der aktiven Benutzer und deren Tätigkeiten. Daher kann es zu plötzlichen und unerwarteten Anstiegen der Auslastung kommen, die das System an die Grenzen seiner Ressourcen bringt. Die vereinbarten Service-Level können auf Grund der resultierenden Performanceeinbußen nicht mehr eingehalten werden. Um schwankende Auslastungen zu bewältigen, können Strategien wie Autoscaling oder Throttling angewendet werden.

Strategien

Beim Autoscaling kann die Cloud die Bereitstellung weiterer Ressourcen anfragen, wenn die eigenen Ressourcen nicht ausreichen. Bei einem unerwartet starken Anstieg der Auslastung kann es dennoch zum Defizit kommen, da diese Bereitstellung nicht unmittelbar erfolgt. Bei der Throttling-Strategie dürfen Anwendungen die Ressourcen bis zu einem Softlimit verbrauchen. Wenn dieses Softlimit überschritten wird, werden Anwendungen gedrosselt, sodass das System weiterhin seine Funktion erbringen kann und die vereinbarten Service-Level eingehalten werden. Das Drosseln kann durch Ablehnen von Benutzeranfragen über eine Zeitperiode oder durch das Deaktivieren / Einschränken von Funktionalität bestimmter nicht essentieller Services erreicht werden (z.B. Begrenzung der Auflösung bei Videostreaming). Dabei können die Ressourcen aller Benutzer gleichermaßen (Queue-based Load Leveling Pattern) oder bei Benutzer mit einem höheren Service-Level geringer eingeschränkt werden (Priority Queue Pattern). Im Folgenden soll der Ablauf der zwei Strategien Throttling und Autcoscaling in Kombination gezeigt werden. Die Abbildung 1 zeigt dazu den Ressourcenverbrauch aller Anwendungen über die Zeit an. Bis zu dem Zeitpunkt T1 ist der Ressourcenverbrauch unterhalb des Softlimits. Ab dem Zeitpunkt T1 wird dieses Limit überschritten und das System fragt die Bereitstellung weiterer Ressourcen an (Autoscaling). Bei einem schnellen Anstieg des Ressourcenbedarfs kann das System die maximale Belastung erreichen, da das Autoscaling Zeit in Anspruch nimmt. Um dies zu verhindern, kann eine Throttling-Strategie zwischen der Zeitspanne T1 und T2 eingesetzt werden. Das Throttling kann zum Zeitpunkt T2, wenn die angefragten Ressourcen zur Verfügung stehen, wieder verringert werden.

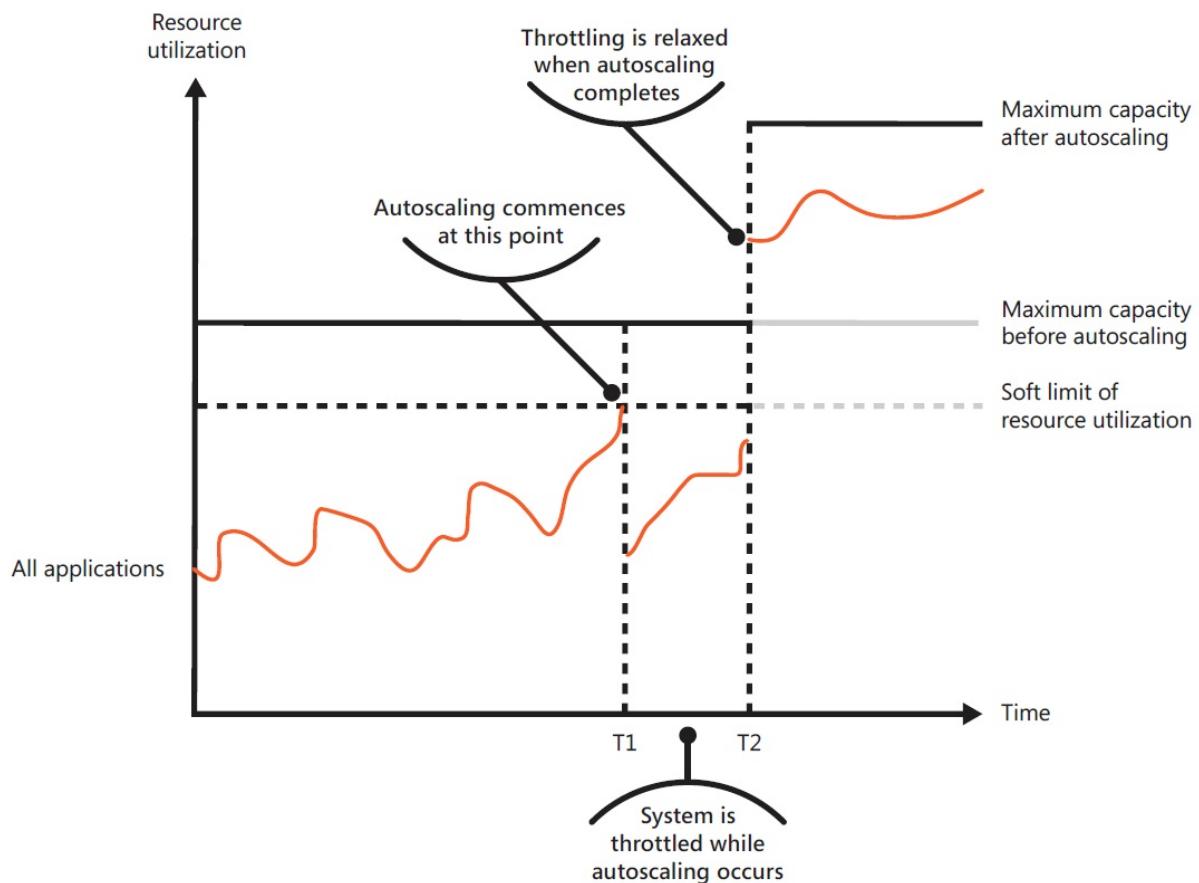


Abbildung 1: Kombination der Throttling- und Autoscaling-Strategie [Cloud Design Patterns, S. 157].

Weitere Betrachtungen

- Eine Throttling-Strategie hat Auswirkungen auf das Systemdesign und muss daher früh betrachtet werden.
- Das System muss das Drosseln schnell umsetzen können.
- Das System muss einer gedrosselten Client-Applikation eine aussagekräftige Fehlermeldung zurückgeben, sodass diese entsprechend reagieren kann.
- Bei kurzen Anstiegen des Ressourcenbedarfs sollte auf Autoscaling verzichtet werden, da diese Strategie kostspielig ist.
- Eine aggressivere Throttling-Strategie oder eine größere Ressourcenreserve sollte betrachtet werden, wenn das System trotz einer aktiven Drosselung an die Grenzen geht und dies nicht tolerierbar ist.

Einsatz des Pattern

- Einhaltung von Service-Level-Vereinbarungen.
- Verhindern, dass eine Teilnehmer-Applikation große Mengen an Ressourcen für sich

selbst beansprucht.

- Abfangen von Spitzen des Ressourcenverbrauchs.

Valet Key Pattern

Das Valet Key Pattern wird bei Server-Client-Systemen angewendet, um Clients den direkten, aber eingeschränkten, Zugriff auf dedizierte Ressourcen zu ermöglichen. Das soll den Datenumfang der Serverapplikation reduzieren, ohne die Datensicherheit zu gefährden. Das Pattern ist besonders bei Applikationen nützlich, die Cloud-basierte Speichersysteme nutzen, da so Kosten minimiert und die Skalierbarkeit und Performanz des Gesamtsystems maximiert werden können.

Kontext und Problemstellung

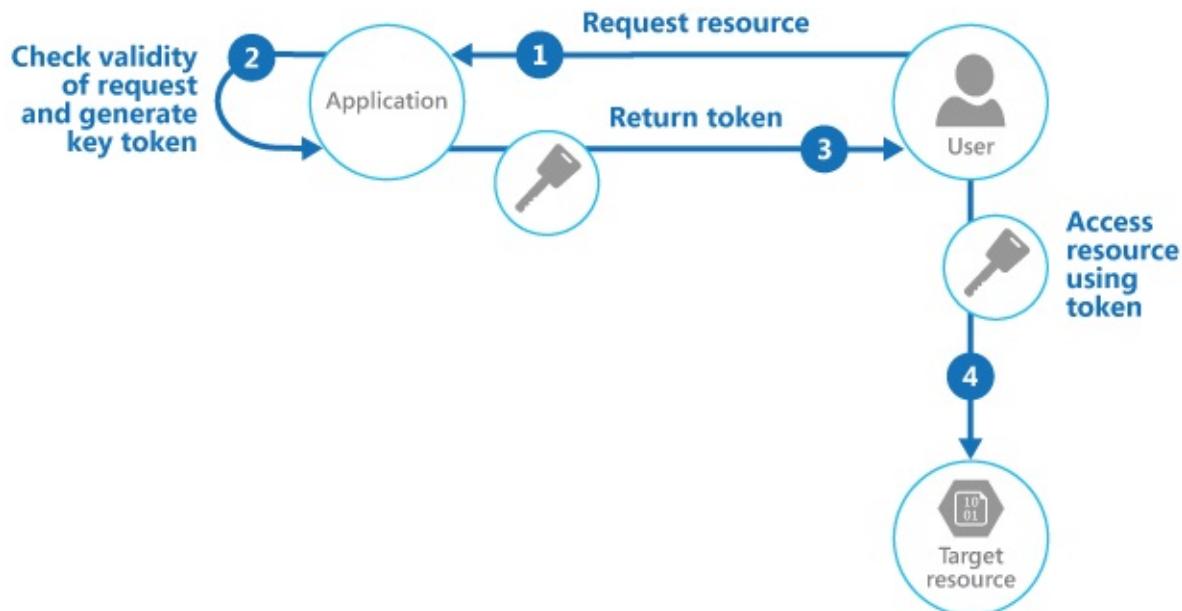
Bei Server-Client-Systemen ist typischerweise die Server-Applikation dafür zuständig Daten zwischen Clients und dem eigenen Datenspeicher hin- und herzutransferieren. Dieses Vorgehen bindet jedoch wertvolle Ressourcen, wie Rechenzeit, Arbeitsspeicher und Bandbreite. Dabei sind Datenspeicher eigentlich in der Lage den Datenaustausch mit dem Client direkt zu regeln, ohne die Applikation zu belasten.

Dafür braucht ein Client Zugriff auf die Sicherheitsschlüssel des Datenspeichers. Sobald der Client diesen Zugriff hat, hat die Applikation jedoch keine Handhabe mehr über den Datenaustausch zwischen Client und Datenspeicher; sie kann also nicht mehr als Gatekeeper agieren. Dieser direkte und damit unkontrollierte Zugriff ist bei verteilten Systemen, die nicht-vertrauenswürdige Clients ausschließen können müssen, kritisch und deshalb nicht gangbar. Applikationen müssen stets in der Lage sein den direkten Datenzugriff durch den Client granular zu steuern und ggf. zu verwahren, um Datensicherheit zu garantieren und gleichzeitig die eigenen Ressourcen zu schonen.

Lösungsansatz

Es bedarf der dedizierten Zugriffskontrolle auf den Datenspeicher. Letzterer kann die Authentifizierung und Authorisierung der Clients aber nicht selber durchführen. Eine typische Lösung für dieses Problem ist es, dem Client, durch die Verwendung von Schlüsseln oder Tokens, nur eingeschränkten Speicherzugriff zu geben. Diesen Schlüssel nennt man Valet Key, welcher dem Client temporären Zugriff ausschließlich auf benötigte Speicherbereiche gewährt. Die Ausstellung der Valet Keys zur Laufzeit durch die Applikation ist dabei schnell und einfach, wordurch die Auslastung von Applikation und Server durch Datentransferprozessen effektiv gesenkt wird.

Nach der Ausstellung eines Tokens benutzt der Client diesen, um auf eine dedizierte Ressource zuzugreifen; und zwar nur für einen beschränkten Zeitraum und mit restriktiven Zugriffsrechten (vgl. Abbildung). Zum einen kann so nur auf bestimmte Speicherbereiche zugegriffen werden (Tabellen, Tabellenzeilen, Container, Container-Elemente). Zum anderen verliert der Token nach einer bestimmten Zeit seine Gültigkeit und dem Client wird der Ressourenzugriff wieder verwehrt. Zusätzlich kann die Applikation die Gültigkeit vorzeitig beenden, wenn z.B. der Client den erfolgreichen Abschluss eines Datentransfers gemeldet hat.



Probleme und Überlegungen

Wenn es um die Implementierung des Patterns geht, sind folgende Aspekte zu berücksichtigen:

- Verwaltung des Gültigkeitsstatus' und -zeitraums: Für den Fall, dass ein Sicherheitsschlüssel durchgesickert ist und die Gefahr der boshaften Verwendung besteht, gibt es verschiedene Gegenmaßnahmen. Der Schlüssel kann als ungültig markiert, die Server-Rechte angepasst oder der korrespondierende Server-seitige Schlüssel deaktiviert werden. Präventiv kann bereits bei der Schlüsselgenerierung die Gültigkeitsdauer möglichst kurz gesetzt werden; dabei darf sie aber nicht so kurz gewählt werden, dass der Client nicht genügend Zeit zur kompletten Datenübertragung hätte.
- Kontrolle des Zugrifflevels: Der Sicherheitsschlüssel sollte dem Nutzer ausschließlich den Ressourenzugriff ermöglichen, der für die aktuelle Operation notwendig ist. Soll der Nutzer nichts hochladen dürfen, genügt einfacher Leserzugriff. Soll er lediglich Daten hochladen, genügt ausschließlich Schreibzugriff in einen bestimmten leeren Bereich.

- Lenken des Benutzerverhaltens: Die granulare Ressourcenkontrolle ist durch die Funktionalitäten der Applikationsdienste und des Datenspeichers beschränkt, weshalb die Client-Operationen nur eingeschränkt lenkbar sind. So hat man keine Handhabe über die Anzahl der Uploads oder die Dateigrößen, wodurch große unerwartete Kosten entstehen können. Um die Anzahl der Uploads zu kontrollieren, unterstützen Datenspeicher jedoch oft die Möglichkeit erfolgreichen Uploads zu melden, was die Applikation zur Überwachung nutzen kann.
- Validierung und optionale Zensierung hochgeladener Daten: Ein boshafter Nutzer kann kompromitierte Daten hochladen oder ein autorisierter Nutzer lädt versehentlich defekte oder manipulierte Daten hoch. Deshalb müssen alle Uploads validiert und überprüft werden.
- Alle Operationen können durch Protokollfunktionen der Schlüsselbasierten Mechanismen protokolliert werden, was für die Prozessoptimierung oder Rechnungsstellung gegenüber dem Nutzer verwendet werden kann.
- Sichere Übermittlung des Schlüssels durch Einbettung in die URL, automatischen Download und der ausschließlichen Übermittlung via HTTPS.
- Sensitive Daten bei der Übermittlung schützen durch die verpflichtende Verwendung von SSL oder TLS.

Anwendungsszenarien

Nützlich bei folgenden Anwendungen:

- Minimierung der Ressourcenlast und Maximierung der Performanz und Skalierbarkeit, da die Schlüsselerzeugung typischerweise eine einfache kryptografische Operation ist.
- Minimierung der Operationskosten, da der direkte Datenzugriff die Netzwerklast reduzieren kann.
- Wenn Clients regelmäßig Daten hochladen, besonder bei großen Speichern oder großen Dateien.
- Wenn die Applikation nur limitierte Hardwareressourcen zur Verfügung hat.
- Wenn Daten dezentral gespeichert sind und die Applikation häufig vermitteln muss.

Nicht nützlich bei folgenden Anwendungen:

- Wenn die Applikation Daten bearbeiten oder validieren muss, bevor sie an den Client gesendet werden können.
- Wenn das Design der Applikation die Implementierung des Patterns schwierig macht; das Valet Key Pattern sollte bei der initialen Architektur berücksichtigt werden.
- Wenn die Anzahl der Up- und Downloads genau überwacht werden muss und der Datenspeicher über keinen Mitteilungsmechanismus verfügt.
- Wenn die Größe der Uploads begrenzt werden muss. Alternativ kann nach dem Upload

die Dateigröße kontrolliert werden oder zyklisch während des Uploads.

Asynchronous Messaging Primer

Der Nachrichtenaustausch ist einer der Schlüsselfunktionen von verteilten Systemen. Dieser ermöglicht es Applikationen und Diensten miteinander zu Kommunizieren und zu kooperieren und hilft dabei skalierbare und belastbare Systeme zu entwerfen. (vgl. S. 166)

Queue-Grundlagen

Innerhalb von Cloud-Technologien ist der Nachrichtenaustausch häufig in Form von Queues (Warteschlagnen) implementiert. Diese unterstützen meist folgende drei Operationen: Ein Absender kann eine Nachricht in die Queue legen, ein Empfänger kann diese Nachricht untersuchen oder sie aus der Queue entnehmen. (vgl. S. 166)

Senden und Empfangen von Nachrichten mithilfe einer Queue

Eine Queue kann man sich tatsächlich wie eine Warteschlange vorstellen. Der Absender hängt hinten an die Warteschlange neue Nachrichten an, die der Empfänger von der anderen Seite abarbeitet (FIFO-Prinzip). Versucht ein Empfänger aus einer leeren Queue eine Nachricht zu entnehmen, so wird diese Operation geblockt. Häufig ist es den Empfängern auch möglich zu Prüfen ob Nachrichten zur Verfügung stehen, sodass das Blocken umgangen werden kann. (vgl. S. 166)

Die Infrastruktur der Queue muss dafür sorgetragen, dass eine einmal hineingelegte Nachricht nicht verloren geht. (vgl. S. 166)

Queues sind ideal für das asynchrone Arbeiten zweier oder mehrerer Dienste. Nach dem Ablegen einer Nachricht in einer Queue, kann sich der Absender anderen Aufgaben widmen. Zusätzlich werden Queues häufig in Verbindung mit mehreren Sendern und Empfängern eingesetzt.

Grundlegende Message-Queue-Patterns

One-way messaging ist die einfachste Form der Implementierung. Absender legen Nachrichten in eine Queue die anschließend von Empfängern bearbeitet werden. Es findet keine Überprüfung statt, ob die Nachrichten tatsächlich irgendwann von einem Empfänger bearbeitet werden. (Vgl. S. 168)

Beim **request/response messaging** erwartet der Absender eine Rückmeldung des Empfängers, dass die Nachricht verarbeitet wurde. Die Antwort der Empfänger wird in eine Absender-Spezifische Queue gelegt (jeder Absender hat eine eigene Queue für die Antworten). Erfolgt innerhalb einer bestimmten Zeit keine Antwort, kann ein Empfänger entsprechende Maßnahmen einleiten (z.B. erneutes Senden der Nachrichten). (Vgl. S. 168)

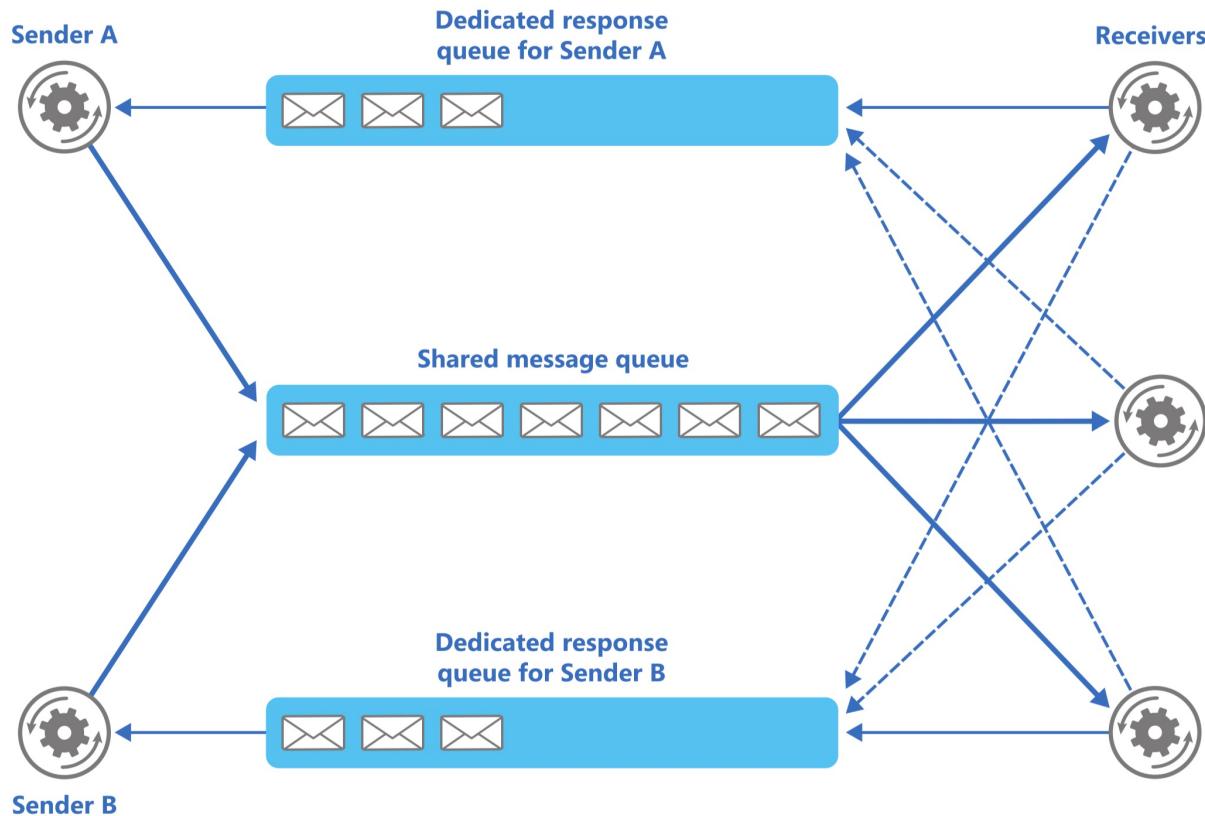


Abbildung 1: Request/Response Messaging [Cloud Design Patterns, S. 168].

Broadcast messaging wird eingesetzt wenn mehrere Empfänger die gleiche Nachricht erhalten sollen. Der Absender legt seine Nachrichten in eine Haupt-Queue, von der diese Nachrichten dann in Empfängerspezifische Queues umkopiert werden.

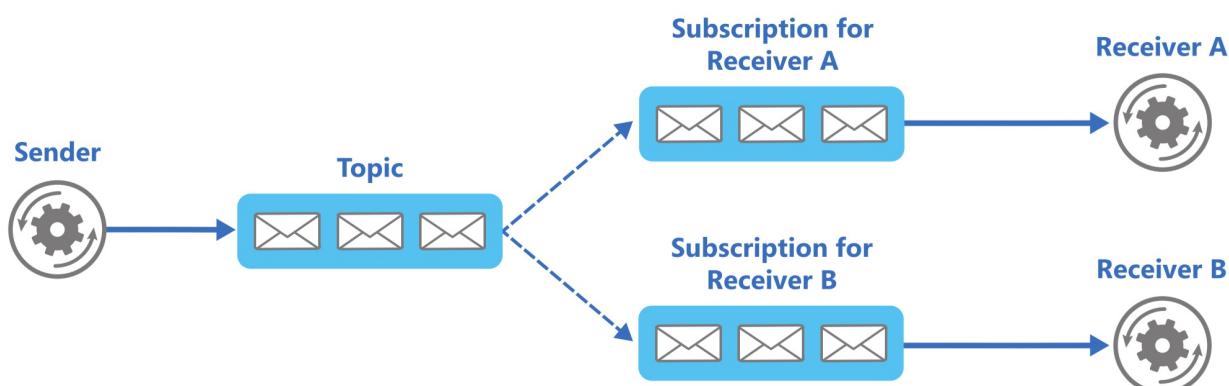


Abbildung 2: Broadcast Messaging [Cloud Design Patterns, S. 169].

Scenarien für asynchronen Nachrichtenaustausch (Auswahl)

Decoupling workloads: Das Trennen des Aufrufens einer Aktion von dessen tatsächlicher Durchführung (Beispiel: Ein Anwender ruft eine Aktion über eine Website auf, dessen Durchführung erfolgt jedoch auf einem Application-Server). (Vgl. S. 169)

Temporal decoupling: Sender und Empfänger können zu unterschiedlichen Zeiten arbeiten. Ein Empfänger kann Nachrichten auch verarbeiten, wenn der Sender nicht zur Verfügung steht. (Vgl. S. 169)

Load balancing: Es kommen verschiedene Empfänger auf mehreren Servern zum Einsatz. Die Architektur ist durch das hinzufügen weiterer Server skalierbar. (Vgl. S. 169)

Load leveling: Beim Erhalten vieler Nachrichten zur gleichen Zeit wird das System nicht direkt in die Höchstleistung getrieben. Der Queue-Mechanismus sorgt für ein langsames Steigen der Anfragen, sodass dem System die Möglichkeit geboten wird, sich den Gegebenheiten anzupassen. (Vgl. S. 170)

Cross-platform integration: Das Zusammenspiel verschiedener Software-Systeme kann über eine Queue erleichtert werden. Eine Queue kann so implementiert werden, dass die Interaktion vollkommen System- und Sprachenunabhängig ist. Es bedarf lediglich einer Definition von Schnittstellen und des Formats der Nachrichten. (Vgl. S. 170)

Überlegungen zur Implementierung von asynchronem Nachrichtenaustausch (Auswahl)

Message ordering: Die Reihenfolge der Nachrichten kann in machen Software-Systemen eine Rolle spielen. Nicht in allen Queues wird eine bestimmte Abarbeitungs-Reihenfolge garantiert. Das "Priority Queue Pattern" realisiert einen Mechanismus der das Abarbeiten bestimmter Nachrichten vor anderen umsetzt (Priorisierung). (Vgl. S. 171)

Message grouping: Typischerweise sollen Nachrichten innerhalb einer Queue unabhängig sein. In manchen Fällen lassen sich die Abhängigkeiten allerdings nicht vollkommen eliminieren. In diesem Fall sorgt "Message Grouping" für das Abarbeiten zusammengehöriger Nachrichten durch den gleichen Empfänger. (Vgl. S. 171)

Idempotency: (Idempotenz) beschreibt, dass die mehrfache Durchführung einer Operation mit den gleichen Daten immer zum gleichen Ergebnis führt. Im Bezug auf den Nachrichtenaustausch ist gemeint, dass auch wenn eine Nachricht fehlerhafterweise mehr als einmal empfangen und bearbeitet wird, die mehrfache Ausführung keine Auswirkungen auf das System hat. (Vgl. S. 171)

Verwandte Entwurfsmuster und Empfehlungen (Auswahl)

Autoscaling Guidance: Sobald die Anzahl der Nachrichten innerhalb der Queue eine bestimmte Schwelle unter- oder überschreitet, kann es sinnvoll sein einen Teil der Empfänger ab- bzw. hinzuzuschalten. (Vgl. S. 172)

Circuit Breaker Pattern: Wenn ein Empfänger oder Sender sich wiederholt nicht mit der Queue verbinden kann, ist es Sinnvoll weitere Versuche zu unterbinden, bis der Fehler gefunden respektive behoben wurde. Dieser Mechanismus wird mit dem "Circuit Breaker Pattern" realisiert. (Vgl. S. 172)

Scheduler Agent Supervisor Pattern: Der Nachrichtenaustausch ist häufig Teil eines größeren Workflows. Das Scheduler Agent Supervisor Pattern beschreibt, wie der Nachrichtenaustausch genutzt werden kann um Aktionen innerhalb von verteilten Systemen zu koordinieren, sowie andere verteilte Ressourcen sinnvoll zu nutzen. (Vgl. S. 173)

Autoscaling Guidance

Was ist Autoscaling?

Autoscaling ist ein dynamischer Prozess der Ressourcen so ver-/ bzw. einteilt, dass eine Applikation die Anforderungen an Performance und die SLAs (service level agreements) erfüllt. Häufig ist er ein automatisierter Prozess, der kontinuierlich die Performance überprüft und entscheidet, ob Ressourcen hinzugefügt oder entfernt werden sollen. Autoscaling bestimmt sämtliche Ressourcen, nicht nur Rechenressourcen.

Scaling Arten

Vertikales Skalieren umfasst das Umstrukturieren der Lösungen durch den Einsatz verschiedener Hardware. In einer Cloud stellt vertikales Skalieren bessere Ressourcen zur Verfügung und überträgt das System auf diese. Durch vertikales Skalieren wird ein System häufig zeitweise nicht erreichbar sein.

Horizontales Skalieren setzt das System auf zusätzliche Ressourcen auf, dabei kann das System ohne Unterbrechung weiterlaufen. Elemente die das System mit einbeziehen, können auf diesen zusätzlichen Ressourcen verfügbar gemacht werden und je nach Bedarf wieder runtergefahren werden. Viele Cloud Systeme nutzen diese Art des Skalierens.

Implementierung einer Autoskalierungsstrategy

Umfasst typischerweise die folgenden Prozesse und Komponenten:

- Einfangen der Schlüssel Performance und Skalierungsfaktoren wie z.B. Antwortzeiten, Warteschlangenlänge, Speichernutzung, etc
- Überwachungskomponenten
- Entscheidungslogik die anhand der überwachten Skalierungsfaktoren entscheidet ob skaliert wird oder nicht. Hierbei spielt die Zeit eine wichtige Rolle.
- Ausführungskomponenten die für Aufgaben verantwortlich sind. Sie nutzen typischerweise Tools und Skripte für:
 - Versorgung mit Ressourcen
 - Rekonfigurierung des Systems
- Testen und Validieren der Strategy

Immer häufiger werden Cloud Systeme mit einem Build-in tooling ausgestattet, um bei der Implementierung des Autoscalings zu helfen.

Gesichtspunkte für das Implementieren von Autoscaling

Autoscaling bietet keine sofortige Lösung, einfach nur Ressourcen oder mehr Instanzen eines Prozesses hinzufügen garantiert keine Performancesteigerung. Folgende Punkte sollten in Betracht gezogen werden:

- Das System sollte horizontal Skalierbar sein. Vermeiden von Instanzähnlichkeiten, keine Codelösungen, die immer in einer spezifischen Instanz eines Prozesses laufen. Anfragen derselben Quelle werden nicht immer der selben Instanz zugeteilt. Autoscaling kann zusätzliche Instanzen eines Dienstes hinzufügen!
- Langanhaltende Aufgaben sollten horizontale und vertikale Skalierung unterstützen. Ohne entsprechende Beachtung könnten Instanzen von Prozessen nicht sauber heruntergefahren werden oder sogar Daten verlieren. Idealerweise wird bei solch langanhaltenden Aufgaben die Abwicklung in kleinere Stücke aufgeteilt. Das "Pipes and Filters Pattern" zeigt ein Beispiel für die Umsetzung.
- Beinhaltet die Lösung mehrere Elemente wie Web Rollen, Arbeiter Rollen und andere Ressourcen, dann sollten diese Elemente als Einheit skaliert werden. Es ist wichtig, die Beziehung der Elemente zu verstehen und sie in Gruppen aufzuteilen.
- Um exzessives horizontales Skalieren zu vermeiden, sollten man dem Autoscaling Grenzen setzen. Autoscaling ist nicht der beste Mechanismus um plötzlichem Arbeitsaufwand entgegenzuwirken, da es Zeit braucht um Instanzen zu starten oder Ressourcen hinzuzufügen. (Throttling Pattern)
- Überwachen und Aufzeichnen von Details jedes Autoscaling Prozesses. Mithilfe dieser Informationen kann die Effektivität der Strategy gemessen werden.

Autoscaling in einer Windows Azure Lösung

- Das *Windows Azure Autoscaling* unterstützt die üblichen Skalierungsszenarien und man kann mit Hilfe des Windows Azure Management Portal Lösungen konfigurieren.
- Die *Microsoft Enterprise Library Autoscaling Application Block* ermöglicht eine Skalierungslösung basierend auf spezifischen Regeln und Performancedaten. Dieser Ansatz ist flexibler und komplexer, braucht aber Code um Performancedaten zu bekommen.
- Die *Microsoft Windwos Azure Monitioring Services Management Library* stellt den Zugang zu den *Windows Azure Monitoring Service* Operationen bereit. Sie beinhaltet eine vereinheitlichte API für das Abrufen und Konfigurieren von Metriken, Warnungen und Autoscaling Regeln der Windows Azure Dienste.

Benutzen von Windows Azure Autoscaling

Es gibt die folgenden zwei Möglichkeiten um Windows Azure für das Autoscaling zu konfigurieren:

- basierend auf Metriken wie die durchschnittliche CPU Leistung der letzten Stunde oder das Backloggen von Elementen in einer Mitteilungswarteschlange. Die Parameter des

Windows Azure Autoscaling können der Skalierung des Systems angepasst werden. Autoscaling ist kein sofortiger Prozess, es braucht Zeit um auf Metriken zu reagieren. Windows Azure setzt Regeln, die z.B. nur eine Skalierung in einem 5 Minuten Intervall zulässt.

- zeitbasierte Autoskalierung um sicherzustellen, dass Instanzen auch verfügbar sind wenn sie gebraucht werden.

Verwandte Muster und Richtlinien

Bei der Verwendung von Autoscaling sind außerdem die Muster "Throttling Pattern", "Competing Consumers Pattern" und die Richtlinien "Instrumentation and Telemetry Guidance" relevant.

Clean Code

Zusammenfassung des Buches:

Titel: Clean Code - A Handbook of Agile Software Craftsmanship

Verfasser: Robert C. Martin

Verlag: Prentice Hall

Jahr: 2009

ISBN: 978-0-13-235088-4

Zusammenfassung von: Wladimir Streck, Lukas Taake, Nils Kohlmeier, Benjamin Schmidt, Tim Jastrzembski, Christian Holzberger

2 Aussagekräftige Namen

3 Funktionen

Funktionen sind der grundlegenste Bestandteil zur Strukturierung von Sofware, sie stammen bereits aus Zeiten von Fortran und PL/1. Ihre Struktur und Qualität macht einen wichtigen Bestandteil zur Lesbarkeit von Quellcode aus. Die wichtigsten Merkmale, um gute Funktionen zu schreiben, werden in diesem Kapitel erläutert.

Anm. d. zusammenfassenden Autors: Das Buch enthält zahlreiche Codebeispiele zur Erläuterung. Diese werden aus Gründen der Übersicht nicht übernommen.

Größe

Gute Funktionen sollten **nicht besonders groß** sein. Als Richtwert wird eine Obergrenze von 20 Zeilen gegeben. Das hat zur Folge, dass sie zu kurz sind um viele verschachtelte Kontrollstrukturen zu enthalten. Große Blöcke innerhalb von Kontrollstrukturen sollten dann in separate Funktionen ausgelagert werden. Das bietet zusätzliche Dokumentation in Form eines Funktionsnamens für diesen Block.

Eine Funktion sollte **genau eine Sache tun**. Diese dafür gut und ausschließlich. Sie sollte, um ihren Zweck zu erfüllen, andere Funktionen der nächst niedrigeren Abstarktionsstufe aufrufen. Es ist ein gutes Indiz, dass eine Funktion mehr als eine Sache tut, wenn eine weitere Funktion extrahiert werden kann, deren sinnvoller Name mehr als eine Neuformulierung ihrer Implementierung ist. Ein weiteres Indiz ist die Unterteilung einer Funktion in Sektionen mittels Kommentaren.

Es sollte nicht mehr als **ein Abstraktionslevel pro Funktion** aufgerufen werden. Sie sollte nur ein Konzept auf nächst niedrigere Stufe hinunter brechen. Andernfalls kann das einen Leser verwirren, der mit den niedriglevligen Details einer Anwendung nicht vertraut ist. Essentielle Funktionen können so schnell zusammen mit Details vermischt werden.

Switch Statements sollten vermieden werden, denn sie tun per Definition immer mehr als eine Sache. In Objektorientierten Spachen können sie durch Polymorphismus fast vermieden werden. Sie müssen lediglich z.B. in einer *Abstract Factory* zur Erzeugung der entsprechenden Klassen eingesetzt werden, können aber an sonst aus dem Code ferngehalten werden.

Namen

Funktionen sollten **beschreibende Namen** tragen. Auch eine kurze Funktion sollte eher einen längeren, aber guten, als einen kurzen, wenig aussagekräftigen Namen tragen. Außerdem sind beschreibende Namen besser, als beschreibende Kommentare. Dazu sollte eine Namenskonvention verwendet, die es erlaubt, leicht Namen aus mehreren Worte zu bilden. Wichtig ist, dass die Namensgebung konsistent ist gleiche Sachverhalte immer gleich benennt. Darüber hinaus ist es nicht völlig untypisch, dass die Suche nach einem guten Namen in der Restrukturierung des Codes endet.

Parameter

Die **Anzahl der Parameter** sollte so niedrig sein, wie möglich. Bevorzugt sind gar keine Parameter, ein oder zwei sind ebenfalls unproblematisch. Drei Parameter sollten vermieden werden, wenn möglich, und mehr sollten auf keinen Fall verwendet werden. Unter Umständen ist es sinnvoll, stattdessen Instanz-Variablen zu verwenden. Return-Parameter sollten generell vermieden werden. Viele Parameter führen neben mehr kognitivem Aufwand beim Lesen einer Funktion auch und vor allem zu einem deutlich erhöhten Testaufwand. Während eine Parameterlose Funktion sehr einfach zu testen ist, liegt der Aufwand bei drei oder mehr Parametern äußerst hoch, um alle Kombinationen zu prüfen.

Beispiele für Funktionen mit einem Parameter, "monadische Form" genannt, sind etwa das Stellen einer Frage `boolean fileExists("FileName")` oder das Umformen eines Parameters `InputStream fileOpen("FileName")`, hier wird der String-Parameter in einen InputStream umgewandelt. Auch die Behandlung von Events kann gut mit einem Parameter umgesetzt werden.

Eine Art von Parametern, die unbedingt vermieden werden sollten, sind Flags. Sie zeigen an, dass eine Funktion eine Sache tut, wenn das Flag gesetzt ist und eine andere Sache, wenn es nicht gesetzt ist. Sie implizieren also eine Verletzung des Single-Responsibility-Principles.

Funktionen mit zwei Parametern werden **dyadisch** genannt. Wenn möglich sollte natürlich versucht werden, sie in eine monadische Form zu überführen. Wenn das nicht gelingt, was häufig der Fall ist, stellt das noch kein Problem dar. Wenn Funktionen allerdings drei Parameter haben, sie werden dann **Triaden** genannt, sollte sehr genau geprüft werden, ob sie nicht wenigstens in eine dyadische Form gebracht werden können.

Wenn es scheint, dass eine Funktion mehr als drei Parameter benötigt, sollten womöglich einige in eine eigene Klasse verpackt werden (z.B. X- und Y-Koordinaten als Punkt). Ein weiterer Spezialfall sind Parameter, die alle gleich behandelt werden und ggf. variabel in ihrer Anzahl sind. Solche Parameterlisten werden im allgemeinen als einzelner Parameter

betrachtet. In Java macht das bereits die Deklaration sichtbar: `String format(String format, Object... args)`, für args können eine beliebige Anzahl Objekte übergeben werden, die dann als Array verarbeitet werden.

Zur Benennung von Funktionen und Parametern bietet es sich an, Verb-Nomen Paare zu bilden. Bei `write(name)` ist, was auch immer name ist, auf den ersten Blick klar, dass es geschrieben wird. Bei Verwendung mehrerer Parameter gibt es noch die Keyword-Form, bei der die Parameter in den Funktionsnamen integriert werden:

```
assertExpectedEqualsActual(expected, actual)
```

Seiteneffekte

Seiteneffekte sollten unbedingt vermieden werden. Solche Funktionen erfüllen einen Zweck und erledigen unbemerkt noch eine andere Sache (z.B. den Zustand eines Parameters verändern, oder den internen Objektzustand). Solche Funktionen lassen versteckte, temporale Kopplungen entstehen. Das kann wiederum schwer auffindbare Fehler hervorrufen.

Ausgabeargumente sollten als eine Art von Seiteneffekten ebenfalls vermieden werden. In objektorientierten Sprachen können sie elegant durch die Verwendung von Objekt-Variablen ersetzt werden.

Command-Query-Separation

Funktionen sollen nur eine Sache machen, also entweder etwas machen, oder Informationen zu etwas zurück geben. **Command-Query-Separation (CQS)** beschreibt genau die Aufteilung in Funktionen, die etwas zurückgeben oder ausführen. Dadurch kann es nicht passieren, dass Rückgabewerte einer ausführenden Funktion fehlinterpretiert werden.

Exceptions statt Fehlercodes

Um das Auftreten von Fehlern zu signalisieren, sollten keine Fehlercodes verwendet werden. Exceptions haben den Vorteil, dass sie nicht sofort nach Ausführung der Funktion ausgewertet werden müssen, sondern dass mehrere Funktionsaufrufe in einem `try`-Block kombiniert werden können. Da `try`-Blöcke den Lesefluss stören und Fehlerbehandlung eine eigene Aktion ist, sollte sie entsprechend in eigene Funktionen ausgelagert werden. Diese Funktion sollte mit einem `try`-Block starten und mit `catch/finally` enden.

Ein weiterer Punkt gegen die Verwendung von Fehlercodes ist, dass diese häufig zentral in einer Klasse oder einem Enum gespeichert werden. Diese wird dadurch zu einem Abhängigkeitsmagneten. Beim Hinzufügen neuer Codes müssten dann alle Klassen, die Fehlercodes benutzen, neu kompiliert und deployed werden. Im Gegensatz dazu können für neue Exceptions einfach neue Klassen erstellt werden, welche bestehenden Code in keiner Weise beeinflussen.

Don't Repeat yourself

Duplikate sind möglicherweise die Wurzel allen Übels in der Softwareentwicklung. Viele Ansätze und Regeln dienen im Kern dazu, verschiedene Arten von Duplikaten zu erkennen und zu vermeiden. Kurze, gut benannte und strukturierte Funktionen sind ebenfalls eine effektive Möglichkeit dazu.

Wie schreibt man solche Funktionen?

Robert Martin schreibt, dass er zuerst meist lange, komplexe und willkürlich benannte Funktionen mit vielen Parametern und Duplikate schreibt. Dafür schreibt er ausführliche Unit-Tests und refactored anschließend so lange, bis viele kleine Funktionen entstehen, die den Beschreibungen dieses Kapitels gerecht werden. Er ist der Meinung, dass niemand solche Funktionen von vorne rein schreiben kann.

Anm. d. zusammenfassenden Autors: Eine guter Ansatz, um von Anfang an bessere Funktionen zu schreiben, ist die Anwendung von Test Driven Developement. Siehe "TDD By Example" von Kent Beck

4 Kommentare

Kommentare in Quelltexten sind meistens ein "notwendiges Übel", welche verwendet werden müssen, da Programmiersprachen nicht aussagekräftig genug sind. Kommentare würden nicht benötigt werden, wenn Programmiersprachen die Absichten der Programmierer besser ausdrücken würden. Deswegen werden Kommentare eingesetzt, um zu erklären, warum ein bestimmter Ansatz gewählt wurde.

Das Einsetzen von Kommentaren ist in der Regel schlecht, da es bedeutet, dass eine Absicht nicht durch den Quelltext ausgedrückt werden kann und der Code nicht aussagekräftig genug ist. Wenn man dabei ist, einen Kommentar zu schreiben, sollte man sich überlegen, ob man den Quelltext nicht so umschreiben kann, dass er aussagekräftiger ist und somit kein Kommentar benötigt wird. Häufige Probleme mit Kommentaren sind zum Beispiel Kommentare, die irreführend sind oder Falschinformationen liefern. Des Weiteren werden viele Kommentare nicht angepasst, wenn der Code umgebaut wird.

Gute Kommentare

Es gibt einige Arten von Kommentaren, die gut sind und deren richtiger Einsatz meist erwünscht ist. Trotzdem sollten Kommentare nur sparsam eingesetzt werden. So sind zum Beispiel rechtliche Kommentare erwünscht, die Copyright-Informationen angeben und beschreiben, unter welcher Lizenz der Quelltext steht. Diese Kommentare sind notwendig, sollte aber so kurz wie möglich gehalten werden. Es ist besser, auf ein externes Dokument zu verweisen, welches Informationen zu der verwendeten Quelltext-Lizenz enthält, als den gesamten Lizenz-Text als Kommentar einzufügen.

Weiterhin sind informative Kommentare erwünscht, die zum Beispiel angeben, welches Muster ein regulärer Ausdruck als gültig erkennt. Neben dieser Art von Kommentaren zählen Kommentare, die beschreiben, warum eine bestimmte Lösung gewählt wurde, zu den guten Kommentaren. Sie erklären, warum ausgerechnet die Lösung, die implementiert wurde, gewählt wurde.

Ein großes Problem sind nichtaussagende Rückgabewerte, die meist Integerwerte sind. Ein gutes Beispiel sind die Rückgabewerte der `compareTo()`-Methode. Sie liefert je nach Ergebnis des Vergleichs -1, 0 oder 1 zurück. Die beste Lösung für dieses Problem wäre eigentlich, dass der Code so geändert wird, dass die Rückgabewerte aussagekräftig sind. Dies lässt sich aber nicht immer durchführen, da man nicht jeden Quelltext ändern kann (die Standard-Bibliothek zum Beispiel). In diesem Fall ist es dann angebracht, Kommentare beim

Aufruf der Methoden zu verwenden, die die Bedeutung des Rückgabewertes beschreiben. Ein Risiko besteht darin, dass die Kommentare falsch sind, wenn der Code geändert wurde, die Kommentare aber nicht angepasst wurden.

Des Weiteren sind Kommentare erwünscht, die vor Gefahren warnen. Zum Beispiel, wenn eine bestimmte Methode nicht threadsafe ist.

TODO-Kommentare sind meist nützlich, um sich schnell Aufgaben für bestimmten Code-Teilen zu notieren. Auch wenn Aufgaben an andere abgegeben werden sollen, sind TODO-Kommentare nützlich. Zu beachten ist, dass sie wieder schnell aus dem Quelltext entfernt werden.

Zusätzlich sind Kommentare, die die Wichtigkeit von Code hervorheben, angemessen, wenn der Code überflüssig erscheint, um Probleme zu verhindern. Andere Entwickler könnten ansonsten Codeteile überarbeiten und wichtigen, aber überflüssig erscheinenden Code entfernen.

Weitere wichtige Kommentare sind JavaDocs für öffentliche APIs. Ohne gute Dokumentation sind neue APIs nur schwer zu verwenden.

Schlechte Kommentare

Die meisten Kommentare sind schlecht. Viele Kommentare sind meist ein Indikator für schlechten Code. So sollten zum Beispiel redundante Kommentare, also Kommentare, die selbsterklärenden Code erklären vermieden werden. Außerdem werden manchmal Kommentare geschrieben, die irreführend und nicht präzise genug sind. Sie können den Betrachter verwirren und letztendlich muss er sich doch den Quelltext ansehen, um zu verstehen, was dort passiert. Eine weitere schlechte Art von Kommentaren sind Kommentare, die wegen Coding-Richtlinien geschrieben werden und eigentlich nicht nötig wären. Sie blähen den Quelltext nur unnötig auf und hindern Betrachter daran, schnell durch den Quelltext durchzusteigen.

Ferner sollten Kommentare mit Änderungshinweisen vermieden werden. Dies sind Kommentare, die auflisten, wann wer welche Änderung an der Datei durchgeführt hat. Heute gibt es dafür Versionierungssysteme, die diese Aufgabe übernehmen. Genauso sind Kommentare überflüssig, die die Autoren von Codeblöcken angeben. Auch diese Aufgabe übernehmen Versionierungssysteme. Außerdem sollten auch überflüssige Kommentare vermieden werden. Kommentare für Getter- und Setter-Methoden werden zum Beispiel nicht benötigt. Sie sind redundant, da der Name der Methode schon aussagt, welche Art von Werten die Methoden zurückliefern oder entgegennehmen.

Auch sollten Kommentare nicht verwendet werden, wenn sie durch die Benutzung von Variablen oder Funktionen überflüssig wären. So sollten Codeblöcke in Methoden mit aussagekräftigen Namen ausgelagert werden, wenn sie Kommentare benötigen, die beschreiben, was die Codeblöcke machen.

Positionsmarkierungen, also Kommentare, die Gruppen von Funktionen und Methoden markieren, sollten nur selten eingesetzt werden. Werden Positionsmarkierungen zu häufig eingesetzt, werden sie ignoriert und blähen damit den Quelltext nur unnötig auf.

Ein Indikator für zu große Codeblöcke sind Kommentare, die angeben, zu welcher öffnenden Klammer eine schließende Klammer gehört. Sind diese Art von Kommentaren notwendig, sind die Codeblöcke zu groß und müssen zum Beispiel durch Auslagerung von Codeteilen in Methoden verkleinert werden.

Quelltext, der auskommentiert ist, sollte entfernt werden. Codeleichen machen den Quelltext unübersichtlich. Versionierungssysteme helfen dabei, Code, der entfernt wurde, wiederzubeschaffen.

Kommentare, die HTML-Formatierungen beinhalten, sollten außerdem nicht verwendet werden. Auch sie blähen den Quelltext unnötig auf und machen die eigentlichen Kommentare im Quelltext schwer lesbar. Die Programme, die die Kommentare aus den Quelltexten extrahieren, sollten die Formatierungen hinzufügen.

Auch Kommentare, die Informationen zum gesamten System enthalten oder Teile beschreiben, die nichts mit dem direkt anliegenden Codeblock zu tun haben, sollten vermieden werden. Es besteht die Gefahr, dass der Kommentar nicht angepasst wird, wenn etwas an anderer Stelle geändert wird. Wenn ein Kommentar geschrieben werden muss, sollte darauf geachtet werden, dass er nur den direkt anliegenden Codeblock beschreibt. Außerdem sollten Kommentare immer kleingehalten werden und nicht einen standardisierten Ablauf erklären. Diese Informationen können im Internet nachgelesen werden. Man sollte hierbei lieber auf die entsprechenden Dokumente dazu verweisen.

Außerdem sollten JavaDoc-Kommentare in nichtöffentlichen Quelltext vermieden werden. Sie liefern keine nützlichen Informationen und das Warten der JavaDoc-Kommentare ist umständlich.

5 Formatierung

Ein guter Entwickler sollte nicht nur funktionierenden Code entwickeln, sondern auch guten lesbaren Code, da ein guter lesbarer Code die Wartung und die Erweiterbarkeit einer Software deutlich verbessert. In einem Team müssen den Entwicklern die Formatierungsregeln bekannt sein, damit jeder Entwickler sich schnell im Code zu Recht finden kann. Diese Formatierungsregeln müssen mit dem Team abgesprochen werden, damit alle die gleichen Formatierungsregeln verwenden. Wir werden ein paar Standartformatierungsregeln uns anschauen, die man gut verwenden kann. Als erstes schauen wir uns die Dateigröße an und sehen, dass große Projekte oft aus vielen großen Dateien bestehen. Die meisten Dateien haben im Schnitt 200 Zeilen. Diese Größe ist ein guter Richtwert für eigene Projekte.

Um den Code gut lesbar zumachen, sollte man Leerzeilen einbauen. Die Leerzeilen haben den Vorteil, dass die Augen immer auf eine neue Zeile gerichtet werden und der Code dadurch auch strukturiert wird. Der Code sieht flüssiger aus, als ohne Leerzeilen. Man trennt die Package Deklaration, die Import-Anweisungen und die Funktionen in einer Klasse mit Leerzeilen. Das ist vertikale Offenheit. Wenn vertikale Offenheit gilt, dann kann man die Bestandteile des Codes mehr im Zusammenhang setzen. Das heißt vertikale Dichte. Zusammenhängender Code wird untereinander ohne Leerzeile voneinander getrennt. Also zum Beispiel gibt es zwei Variablen, diese stehen direkt untereinander, weil die beiden Variablen zusammengehören. Ein weiterer Schritt für guten lesbaren Code ist der vertikale Abstand. Der Abstand zu zusammenhängenden Code sollte möglichst klein sein. Also sollte man nicht Variablen-deklarationen irgendwo sinnlos im Code positionieren, sondern in der Nähe wo man diese verwendet. Zum Beispiel in Funktionen sollten lokale Variablen am Anfang stehen und in Schleifenanweisungen sollte man die Schleifenvariable deklarieren, da der Abstand zu der Verwendung der Variable sehr gering ist. Bei langen Funktionen können die Variablen auch vor der Schleife und vor der Funktion deklariert werden. Aber Instanzvariablen sollten immer am Beginn einer Klasse stehen. Sie werden von fast allen Methoden verwendet und sollten deswegen am Anfang der Klasse stehen. Auch abhängige Funktionen sollten untereinander stehen. Wenn zum Beispiel eine Funktion eine andere Funktion aufruft, dann sollte die aufgerufene Funktion unter der anderen stehen. Dieses gilt auch für Codeteile die sich sehr ähnlich, denn diese weisen eine konzeptionelle Affinität auf. Diese Codeteile sollten einem geringen vertikalen Abstand zu einander haben. Zum Beispiel wenn Funktionen einen gleichen Methodenkopf haben, dann aber andere Parameter übergeben bekommen als die andere Funktion. Es ist wichtig, dass der Code einer vertikalen Anordnung gleicht. Der Code sollte sich von oben nach unten lesen lassen, also sollten die Funktionen, welche als erstes aufgerufen werden, auch als erstes im Code folgen. Man findet sich dadurch leichter im Code zurecht.

Die Horizontale Formatierung befasst sich mit der Zeilenbreite im Code. Die Zeilen sollten kurz sein. Am besten ist es, wenn man eine Zeilenbreite von maximal 120 Zeichen nimmt, da der Benutzer sonst sehr lange nach rechts scrollen muss. Das Scrollen stört den Lesefluss des Benutzers und kostet einen Entwickler Zeit und Nerven. Mit dem Whitespace kann der Entwickler Zusammenhänge im Code gut verdeutlichen oder auch keine Zusammenhänge im Code zeigen. Zum Beispiel bei der Zuweisung der Variablen, dort wird die Zuweisung mit einem Gleichheitszeichen und einem Leerzeichen zwischen der Variable und dem Wert getrennt. Vor dieser Trennung wird die Variablenzuweisung noch eingerückt nach innen in der Funktion. Man kann mit diesem Schritt gut getrennten Code zeigen. Aber auch umgekehrt geht das, wenn zum Beispiel der Funktionsname und die Funktionsklammer zusammen schreibt ohne diese mit einem Leerzeichen zu trennen. Man kann aber auch so die Parameter im Funktionskopf voneinander trennen, da jeder Parameter für sich steht. Eine andere Besonderheit ist, dass man mit Whitespaces die Reihenfolge von Codeaufrufe gut darstellen kann. Eine Gleichung ist so gut lesbar. Auch Variablendeklarationen sollten nicht eine allzu große Liste darstellen. Eine Alternative wäre die Variablendeklarationen auf mehrere Klassen aufzuteilen. Es ist sehr wichtig den Code einzurücken, da dadurch der Code nicht nur übersichtlicher sondern auch die Verarbeitung der Codestücke untereinander ersichtlich ist. Wir wissen genau, dass zum Beispiel der innere Codeblock als erstes verarbeitet wird und danach der äußere Codeblock. Beide Codeblöcke sind mit geschweiften Klammern voneinander getrennt. Also rückt man als erstes Funktionen ein, dann den Code der diese Funktion implementiert. Der Code besteht aus Bedingungen, Schleifen und Variablendeklarationen. Diese werden auch wieder untereinander eingerückt, auch wenn der Code nur aus einer Zeile besteht.

In einem Team gelten die Formatierungsregeln des Teams, nicht seine eigenen Formatierungsregeln, da jeder Entwickler gerne seine eigenen Regeln benutzen will. Das Team muss eine gemeinsame Basis finden.

6 Objekte und Datenstrukturen

Es ist für einen Entwickler sehr wichtig, dass die Implementierung von außen geheim gehalten wird. Der Entwickler sollte niemals Variablen mit Public definieren, da jeder auf diese Variablen zugreifen könnten und diese dann manipulieren könnte. Es ist viel besser, wenn die Variablen nur mit Get- und Set-Methoden ausgeführt werden. Die Set-Methoden sollten aber nicht die einzelnen Variablen ändern, sondern sie sollten in einem Zusammenhang stehen. Zum Beispiel keine Set-Methoden für setX() oder setY(), sondern viel besser setRechteck() oder setKreis(), da man von außen nicht weiß, welche Variable jetzt verwendet wird. Diese Methoden sollten nach außen durch ein abstraktes Interface dargestellt werden, der Benutzer kann so die Methoden verwenden. Die Implementierung bleibt vor dem Benutzer geheim. Die Methodennamen sollten möglichst abstrakt sein, da man so nicht auf die Daten kommt. Wir wollen den Benutzer so wenig Details geben wie nötig. Das Ziel ist eine hohe Datenabstraktion zu erreichen, die vorigen Tipps helfen einen Entwickler dies zu erreichen.

Man muss beachten das Objekte und Datenstrukturen etwas Verschiedenes sind. Objekte verstecken die Daten und stellen Funktionen bereit, die auf diese Daten arbeiten. Datenstrukturen zeigen ihre Daten und sie stellen keine Funktionen bereit. Diese Kenntnis ist für die Programmierung sehr wichtig, da wir damit entscheiden können, ob wir prozedurale Programmierung oder objektorientierte Programmierung anwenden. Beide Programmierstile haben einige Vorteile für die Datenabstraktion. Die prozedurale Programmierung ist gut, wenn man zu der benutzten Datenstruktur neue Funktionen hinzufügen will. Die objektorientierte Programmierung ist gut, wenn man zu bestehenden Funktionen neue Klassen hinzufügen will. Man kann mit prozeduraler Programmierung Problemstellung in der objektorientierten Programmierung sehr gut lösen. Wiederum kann man mit objektorientierte Programmierung sehr gut Problemstellungen in der prozeduraler Programmierung sehr einfach lösen.

Das Gesetz von Demeter besagt, dass ein Modul, welches ein Objekt verändert, nichts wissen muss wie das Objekt aufgebaut ist. Eine Methode a einer Klasse b sollte am besten nur folgende Methoden benutzen:

- Aus der Klasse b
- Ein erstelltes Objekt von der Methode a
- Ein übergebenes Objekt an der Methode a als Argument
- Ein Objekt das in einer Instanzvariable von der Klasse b gespeichert ist

Die Methoden sollten keine Methoden von fremden Objekten aufrufen, nur der befreundeten Objekte. Es sollten auch lange Aufrufketten vermieden werden also zum Beispiel `getKreis().getRadius().getTyp()`, da diese Aufrufketten keinen schönen Stil haben. Man sollte eine Aufrufkette in mehrere Einzelschritte aufteilen.

Es ist sehr schlecht, wenn Sie hybride Strukturen aus Objekten und Datenstrukturen erstellen, da dadurch unsauberer Code erstellt wird und der Benutzer einen Einblick in die Implementierung bekommt. Und man kann in den Code sehr schwer neue Funktionen hinzufügen. Sie sollten diese möglichst vermeiden.

Um Strukturen sehr gut zu verstecken, nehmen Sie ein Objekt a und übergeben diesem das Objekt b. So kennt Objekt a nichts von Objekt b kann dieses Objekt aber verarbeiten. So zum Beispiel `a.createFile(b)` , da Objekt a jetzt eine Datei erzeugt und dafür Objekt b bekommt.

Datentransfer-Objekte sind Klassen, welche keine Funktionen haben aber öffentliche Variablen besitzen. Die Klassen sind Datenstrukturen. Java Beans nutzt diese Datentransfer-Objekte sehr oft, um rohe Daten in Objekte zu kapseln. Es gibt noch die Active Records, diese sind auch Datentransfer-Objekte. Der Unterschied zu dem normalen Datentransfer-Objekte ist, dass diese Objekte noch zusätzliche Funktionen besitzen. Mit diesen Funktionen kann man durch die Datenstruktur navigieren. Viele Entwickler benutzen diese Datenstruktur als Objekte und fügen Funktionen mit passender Geschäftslogik mit ein. Durch diesen Schritt wird aus der Datenstruktur ein hybrider und das ist für den Entwickler sehr schlecht, in Bezug auf das Verstecken der Implementierung nach außen. Besser ist es, wenn man die Active Records als Datenstruktur nur nutzt und die Geschäftslogik in passende Objekte der Active Records einfügt, da die Daten dort versteckt sind und von außen nicht erreichbar sind.

7 Fehler-Handling

Unser Ziel ist es sauberen Code zuschreiben, aber auch gleichzeitig die Anwendung vor Abstürzungen zu schützen. Genau dafür ist das Fehler-Handling zuständig. Wir werden dafür Ausnahmen benutzen, statt nur das Ergebnis des Fehlers zurück zugeben und dieses Ergebnis mit mehreren If-Abfragen zu prüfen. Das Prüfen mit mehreren If-Abfragen führt zu unnötigen Fehlern und zusätzlichen unnötigen Code, besser sind Ausnahmen. Die Geschäftslogik ist dadurch besser vom Fehler-Handling getrennt und der Code ist dadurch viel sauberer.

Try-Catch-Blöcke definieren für sich einen eigenen geschlossenen Bereich im Code. Ein Entwickler sollte immer mit Try-Catch-Anweisungen arbeiten, da beim Auftreten eines Fehlers im Try-Block der Fehler im Catch-Block gefangen wird. Als erstes sollte ein Entwickler immer Tests schreiben, dabei sollte der Entwickler dort schon ausnahmen berücksichtigen. Denn Tests mit Ausnahmen helfen den Entwickler schon möglichst früh Fehler abzufangen. Der Entwickler baut so Stück für Stück seinen Try-Block auf und kann gleichzeitig beim finden eines Fehlers den passenden Catch-Block hinzufügen. So baut der Entwickler das Fehler-Handling ganz genau auf und trennt dieses von der Geschäftslogik.

Sie sollten als Entwickler keine Unchecked Exceptions bei Methoden werfen, da Sie bei jeder weiteren Methode die throw-Klausel hinzufügen müssten, wenn die Methode von einer anderen Methode aufgerufen wird. Sie als Entwickler müssen die den Fehler solange an einer höheren Methode weiterleiten bis ein passender Catch-Block für diesen Fehler ausgeführt wird. Gerade bei großen Systemen kann der Aufwand dafür Erheblich sein.

Wichtig ist das Sie als Entwickler immer die Fehlermeldungen sehr gut beschreiben, damit Sie als Entwickler immer die Quelle und den Ort des Fehlers wiederfinden, um diesen Fehler dann zu beheben. Sie sollten in den Fehlermeldungen folgende benötigte Informationen nennen den Stack-Trace, den Namen und den Typ der gescheiterten Operation. Mit diesen Informationen können sie sehr schnell den Fehler finden und diesen dann beheben.

Sie können auch Exception-Klassen definieren. Wenn Sie mehrere zusammengehörende Catch-Blöcke haben, dann können Sie diese zu einer Exception-Klasse zusammenfassen und das Objekt der Exception-Klasse ausführen. Sie übergeben dann alle Parameter dem Objekt. Das Objekt ruft die Konstruktor-Methode auf und in der Konstruktor-Methode werden die Parameter gesetzt, da sich dort die Geschäftslogik aus dem Try-Block befindet. Selbstverständlich ist dieser Code wieder in einem Try-Block drin und die zusammenhängenden Catch-Blöcke kommen nach dem Try-block. Jetzt muss man nur noch das Objekt in dem ursprünglichen Try-Block einfügen und noch eine Catch-anweisung, falls das Objekt nicht generiert wird. Durch eine Exception-Klasse können wir den Code

besser zusammenfassen und der Code ist besser austauschbar. Die Exception-Klasse kann gut für fremde APIs genutzt werden, da man diese mit anderen APIs austauschen kann, wenn man eine neue API verwenden will.

Man sollte als Entwickler niemals zu viele Null-Objekte zurückgeben, da diese eventuell nicht alle abgefangen werden. Wir müssten diese als Entwickler dann auch abfangen, besser ist es wenn man ein Objekt zurückgibt, welches einen Spezialfall definiert oder man gibt eine Ausnahme zurück. Der Entwickler muss so nicht mehr so viele Null-Abfragen definieren oder man gibt eine leere List als Objekt zurück, durch dieses Vorgehen kann es nicht zu einer NullPointerException kommen und damit stürzt das Programm nicht ab.

Ein Entwickler sollte niemals eine Null als Parameter einer Methode übergeben, denn das kann zu einer NullPointerException führen. Der Entwickler kann stattdessen ein Objekt übergeben, welches diesen Spezialfall abfängt.

8 Grenzen

Unternehmen produzieren selten die komplette Software eines Systems. Meist arbeiten sie mit anderen Teams zusammen, welche ihre eigene Software schreiben, oder sie beziehen fertige Software von Drittanbietern (sowohl lizenziert als auch Open Source). Damit diese auch sauber in das eigene System eingebunden werden kann, sollte man das folgende Kapitel berücksichtigen.

Fremdcode nutzen

In der Regel kann die angebotene API von Drittanbietern wesentlich mehr als der Nutzer für sein spezielles Problem braucht. Die Drittanbieter wollen nämlich viele Leute für ihre API begeistern. Der Nutzer hingegen erwartet Funktionen von der API, die speziell für sein Problem geeignet sind. Dieser Interessenskonflikt kann die Grenzen unseres Systems unnötig verkomplizieren.

Die Grenzen erforschen

Das Nutzen von Drittanbieter-Code ermöglicht das Gestalten von mehr Funktionalitäten in weniger Zeit. Wie und wo soll man jedoch jenen Code verwenden? Wie benutzt man ihn? Um das Verständnis für solchen Code zu verbessern, liegt es im eigenen Interesse, diesen auszutesten. Dafür bieten sich sogenannte **Learning Tests** an.

Learning Tests bestehen darin, dass man den Fremdcode weitestgehend auf die benötigten Funktionalitäten testet, wie sie später im eigenen Code verwendet werden. Das ermöglicht ein präzises Lernen der API in einer isolierten Testumgebung. Zudem eignen sich diese Tests zur Evaluation für spätere Updates des Drittanbieter-Codes, da diese direkt ungewollten Veränderungen bzgl. der genutzten Funktionen aufzeigen können.

Die Tests sollten eine saubere Grenzeinhaltung wie im richtigen Quellcode gewährleisten. Ohne diese Grenztests könnte man an ältere Versionen von Drittanbieter-Codes gebunden werden, was nicht unbedingt von Vorteil ist und zumeist mit weiteren Umstrukturierungsmaßnahmen im eigenen Quellcode endet.

Code nutzen, der (noch) nicht existiert

Es kann vorkommen, dass in der Softwareentwicklung Funktionen benötigt werden, die zwar beschlossen, jedoch dessen Schnittstellen noch nicht weitestgehend definiert worden sind. Damit dies die Entwicklung nicht unnötig aufhält, kann man die Schnittstellen vorerst so definieren, wie man sie sich später im Quellcode vorstellt.

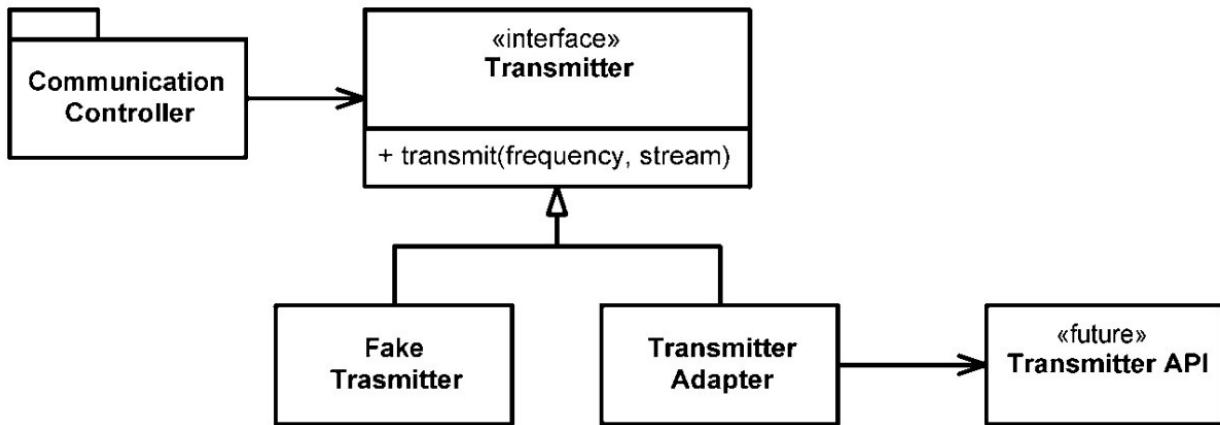


Abbildung 1 (Quelle: Clean Code)"

Abbildung 1 zeigt einen Fall, wo ein Transmitter für die weitere Entwicklung benötigt wird. Die Transmitter API dazu ist jedoch noch nicht fertiggestellt, sodass die Funktionen weder bekannt sind noch genutzt werden können. Der Einsatz vom Adapterpattern erlaubt es trotzdem, Tests für die fehlende Funktionalität zu schreiben und somit eine saubere Grenzeinhaltung zu definieren. Sobald die API fertig wird, zeigen die Tests sogar, ob sie, wie erwartet, arbeitet.

Saubere Abgrenzung

Es kann vorkommen, dass sich die fremde API im Laufe der Zeit verändert. Wenn das der Fall ist, ist es von Vorteil, die Schnittstellen zum eigenen Quellcode so sauber wie möglich gestaltet zu haben. Das spart nämlich Zeit und Arbeit, die man dann in das Überarbeiten dieser Schnittstellen investieren muss.

Der eigene Code an den Schnittstellen muss klar definiert und eine klare Abgrenzung zu Fremdcode haben. Dabei helfen Tests, welche die Funktionserwartungen definieren. Es ist immer besser auf etwas angewiesen zu sein, was man kontrollieren kann, als auf etwas, was man nicht kontrollieren kann, da es ansonsten einen selbst kontrolliert.

9 Unit-Tests

Unit-Tests finden heutzutage vor allem im Test Driven Development und agilen Softwareentwicklungsprozessen sehr viel Verwendung, was vor rund 20 Jahren (1997) noch ganz anders aussah: Viele konnten mit dem Begriff "Unit-Test" nichts anfangen, da es prinzipiell nur Wegwerfcode war, welcher nur bestätigt hat, dass ihr Programm "läuft". Zwar werden viele automatisierte Unit-Tests geschrieben, jedoch vergessen die meisten den wahren Sinn dieser Tests.

Die 3 Gesetze des TDD

Test Driven Development setzt voraus, dass vor dem Verfassen des eigentlichen Quellcodes die dazugehörigen Tests geschrieben werden. Darüber hinaus gibt es diese drei Gesetze, welche die wesentliche Struktur der Test definieren:

1. **Gesetz:** Schreibe zuerst den zum Quellcode passenden Test, bevor du den Quellcode selbst verfasst.
2. **Gesetz:** Schreibe nur so viele Unit-Tests wie nötig, damit ein Test fehlschlagen kann. Ein nicht kompilierbarer Test ist auch ein Fehlgeschlagener Test.
3. **Gesetz:** Schreibe nur so viel Code wie nötig, um den aktuell fehlgeschlagenen Test zu bestehen.

Tests sauber halten

Viele Programmierer tendieren dazu, "Quick and Dirty" zu programmieren. Die resultierenden Konsequenzen werden dabei mit der Zeit immer größer: Je schlampiger die Tests geschrieben werden und je größer der Quellcode wird, desto schwieriger kann er später geändert werden. Im schlimmsten Fall kann sich das Problem wie ein Rattenschwanz sowohl durch den Quellcode als auch durch die dazu geschriebenen Tests ziehen, sodass man im Endeffekt nun die doppelte Arbeitszeit in die Wartung der Tests und des Quellcodes investieren muss. Das kann zur Folge haben, dass die ganze Testsuite verworfen wird, was dann im Laufe der Zeit zu mehr Inkonsistenzen im Quellcode führt. Letztlich beginnt der Quellcode zu verfaulen. Die Tests halten den Quellcode dementsprechend flexibel, wartbar und wiederverwertbar.

Saubere Tests

Lesbarkeit ist der Schlüssel zu sauberen Tests. Die Tests sind lesbar, sofern diese klar und simpel formuliert sind. Das **BUILD-OPERATE-CHECK**-Pattern bietet eine simple und klare Struktur zum Aufbauen von Tests:

1. **BUILD**: In der ersten Phase werden Daten zum Testen vorbereitet.
2. **OPERATE**: In der zweiten Phase werden die Daten von der zu testenden Funktion genutzt.
3. **CHECK**: In der dritten Phase wird überprüft, ob die Funktion die Daten wie erwartet verarbeitet hat.

Zudem sollte man in Erwägung ziehen nur ein Assert und ein einzelnes Konzept pro Test zu verwenden. Somit bleibt der Test übersichtlich und ist für den Leser besser nachzuvollziehen.

Domain-spezifische Testsprache

Es gibt Domain-spezifische Sprachen, welche Funktionen und Tools zum Testen von Quellcode bieten. Mithilfe dieser Sprachen ist das Erstellen und Lesen von Tests wesentlich einfacher.

F.I.R.S.T.

Saubere Tests folgen auch folgenden fünf Regeln, welche unter dem Akronym FIRST bekannt sind.

- **Fast**(Schnell): Tests müssen schnell sein, damit sie frequenziell getestet werden können. Sind sie zu langsam, wird zu weniger Teststarts tendiert, was dazu führen kann, dass bestehende Probleme nicht schnell genug gefunden werden. Das wiederum führt dazu, dass man hinterher den Code aufräumen muss, was zusätzliche Zeit und Arbeit kostet.
- **Independent**(Unabhängig): Tests müssen unabhängig voneinander sein. Man muss alle separat starten können. Wenn Tests voneinander abhängig, verkompliziert sich die Fehleranalyse, da nicht direkt bekannt ist, welche der getesteten Funktion fehlerhaft ist.
- **Repeatable**(Wiederholbar): Tests sollten in jeder Umgebung wiederholbar sein.
- **Self-Validating**(Selbstvalidierung): Man kann Tests entweder bestehen oder bei diesen durchfallen. Entsprechend sollen Tests nur einen `boolean` zurückgeben können.
- **Timeley**(Zeitig): Die Tests sollten direkt vor dem eigentlichen Quellcode entwickelt werden. Wenn der Quellcode davor entwickelt wird, könnte die Testentwicklung zu schwer erscheinen und wird letzten Endes vielleicht nicht umgesetzt.

10 Klassen

Im 10. Kapitel wird beschrieben in welcher Art und Weise mit Klassen umgegangen werden soll.

Eine Klasse soll in folgender Reihenfolge die Definitionen ihrer Bestandteile enthalten:

1. Liste der Variablen
 - i. public static Konstanten
 - ii. privat static Variablen
 - iii. private Variablen
 - iv. public Variablen - Es sollte so gut wie nie ein Grund vorliegen diese einzusetzen

Dazu werden 3 Prinzipien beschrieben

- Klassen sollen klein gehalten werden - Das "Single Responsibility Principle"
- Kohäsion - Sichbarkeit von Details
- Gruppierung nach Änderungshäufigkeit - "Organizing for Change"

Klassen sollen klein gehalten werden

Während man bei Funktionen die Größe anhand der Zeilen von Code bestimmt ermittelt man die "Größe" einer Klasse anhand ihrer Zuständigkeiten. Eine Klasse sollte nach Möglichkeit genau eine Zuständigkeit haben.

Werden innerhalb einer Klasse Methodennamen wie "Manger*", "Prozess", "Super benutzt" ist ein gutes Zeichen dafür, dass die Klasse zu viele Zuständigkeiten hat.

Die Zuständigkeit einer Funktion sollte mit 25 Wörtern beschrieben werden können. Sollte man in dieser Beschreibung die Wörter "Wenn", "Und", "Oder" und "Aber" benutzen ist es ein gutes Zeichen dafür, dass die Klasse zu viele Zuständigkeiten hat.

Single Responsibility Principle (SRP)

Weiterhin kann man überlegen welche Gründe für eine Änderung an der Klasse sprechen (Neue Version, Neue Währung, Anderer Server,...), wenn es mehr als einen Grund für eine Änderung gibt ist die Klasse vermutlich nicht klein genug.

Das SRP legt fest, dass es für eine Klasse genau einen Grund geben darf sie zu verändern. Das führt dazu, dass eine Software die sonst aus eingen großen Klassen bestehen würde aus vielen kleinen Klassen besteht. Diese können dann wie ein Werkzeugkasten eingesetzt

werden.

Kohäsion

Klassen sollten eine wenige Instanz-Variablen besitzen. Jede Methode der Klasse manipuliert eine oder mehrere dieser Variablen. Man spricht von geringer Kohäsion wenn eine Methode eine oder wenige Instanz-Variablen verändert. Verändert alle Methode alle Variablen einer Klasse spricht man von einer Klasse mit maximaler Kohäsion.

Maximale Kohäsion ist anzustreben da dies verdeutlicht, dass die Klasse eine logische Einheit bildet.

Um von einer Klasse mit geringer Kohäsion zu mehreren Klassen mit hoher Kohäsion zu kommen kann man sich dem Refactoring bedienen.

Organizing for Change

Durch sich ändernde Anforderungen wird auch Code im Laufe der Zeit geändert. Jede dieser Änderungen birgt die Gefahr neue Fehler ins System einzubauen. Die Funktion des Systems ist dann nicht mehr oder nur eingeschränkt gegeben. Durch die Organisation von Code in Klassen kann das Risiko, neue Fehler einzubauen, verringert werden.

Durch den Einsatz von SRP und dem Ziel eine möglichst hohe Kohäsion zu erreichen werden Änderungen isoliert. Diese Änderungen können dann gezielt getestet werden.

11 Systeme

Die Erstellung von Software ist nicht nur auf das schreiben von Code beschränkt. Die Organisation des Codes ist von der Anwendung getrennt. Einzelne Bestandteile des Codes werden in einem System benutzt um bestimmte Ziele zu erreichen. Es ist nicht erwünscht, dass jeder Teil des Codes so wenig wie möglich über die Umgebung in der er eingesetzt wird "weiß".

Abhängigkeiten

Die Organisation der Funktionseinheiten sollte ohne Rücksicht auf die konkrete Umsetzung, ohne zu viel "Level of Detail" erreicht werden. z.B. ist es nicht gut wenn eine Klasse Ihre Abhängigkeiten (Datenbankverbindung, etc.) selber erzeugt.

Separation of Main

Eine Möglichkeit um Abhängigkeiten zu isolieren ist es, alle Komponenten innerhalb der Main Methode zu initialisieren und dann den jeweiligen abhängigen Klassen als Konstruktor-Argument mitzugeben.

Hierbei verlässt man sich darauf, dass die Main-Methode keinen Fehler bei der Initialisierung macht und die Abhängigkeiten über die Laufzeit des Programms gültig bleiben.

Factories

Um den Zeitpunkt der Erstellung weiterhin der jeweiligen Klasse zu überlassen, können Factories eingesetzt werden. Factories stellen Methoden bereit die bestimmte Abhängigkeiten zum Anfragezeitpunkt erzeugen.

Dabei bleibt der konkrete Vorgang der Erzeugung vor dem Aufrufer versteckt.

Dependency Injection (DI)

Das Dependency Injection Pattern geht mit dem Inversion of Control (IoC) Pattern einher. Die benötigten Ressourcen werden außerhalb der Klassen erstellt und durch einen Mechanismus in die Klasse injiziert. Die Klasse findet die benötigte Abhängigkeit in einer Ihrer Instanzvariablen vor, ohne die Abhängigkeit selber zu initialisieren. Diese

Vorgehensweise gibt jegliche Kontrolle an den DI-Mechanismus ab, daher spricht man von Inversion of Control. Die Anwendung steuert alle Logik die benötigt ist um Abhängigkeiten bereitzustellen.

Wenn Systeme wachsen

Die beschriebene Art Abhängigkeiten zu injizieren funktionieren gut wenn es sich um Domain-Spezifische-Objekte handelt. Alle Objekte sind hier am erreichen eines (Geschäfts-)Ziels beteiligt. Es gibt allerdings auch Funktionalität die über die "Domain" hinweg eingesetzt werden muss (z.B. Persistenz). Diese Anforderung führt dazu, dass bestimmte Teile des Codes immer wieder verwendet werden. Diese Art der Anforderung wird "Cross-Cutting Concern" genannt.

Um Cross-Cutting Concerns abzubilden sollte man sich einer form der Aspektorientierten Programmierung (AOP) bedien.

Proxies

Das Proxy-Pattern bietet beschränkte Möglichkeit bestimmte Aspekte eines Systems zu Kapseln. Die Persistenz eines POJO (Plain Old Java Object) kann durch einen MySQLProxy erreicht werden. Diese Proxy kümmert sich lediglich darum das Objekt in einer Datenbank abzulegen und reicht alle sonstigen Anfragen direkt an das Objekt weiter.

Der Aspekt der Persistenz wurde dann in das Proxy-Object verlagert.

Proxies müssen explizit eingesetzt werden und können nicht zentral gesteuert werden. Sie machen es schwer die Qualität des Codes herzustellen oder zu erhalten.

Aspect Oriented Programming

Durch den Einsatz von AOP Frameworks kann ein System deklarativ erweitert werden. Die Cross-Cutting Concerns können zentral gesteuert werden und an bestimmte Objekte gebunden werden (z.B. bestimmte Objekte werden immer in die Datenbank geschrieben). Diese Frameworks gibt es in verschiedener Qualität.

Weitere Methoden

Im folgenden werden weitere Wege beschrieben die man beim erstellen komplexer Systeme beachten sollte.

Optimize Decision Making

Entscheidungen sollten zu dem Zeitpunkt getroffen werden an dem sie für das System relevant werden. Frühzeitig getroffene Entscheidungen führen zu einem suboptimalen System, da das Problem noch nicht vollständig erkannt ist.

Use Standard when they add Demonstrable Value

Standards sollten dann eingesetzt werden wenn sie dem System einen echten nutzen hinzufügen. Viel Technologie wird bei Ihrem erscheinen gehyped und bietet wenig nutzen. Die Archtitektur macher Standards entwickelt sich eher von echten Anwendungsfällen weg.

Domain Specific Languages

Programmersprachen bieten viele Features. Sie bieten viele verscheidene Anwendungsmöglichkeiten und Frameworks. Allerdings bietet diese Flexibilität auch Nachteile. Das erstellen einer Sprache die nur für den jeweiligen, konkreten, Anwendungsfall geeignet ist bietet Vorteile bei der Zusammenarbeit mit fachgebietsspezifischen Experten. Diese müssen sich nicht in eine große Programmiersprache einarbeiten sondern haben eine kleine Sprache in der spezifische Befehle für genau eine Domäne hinterlegt sind.

12 Emergenz

Kents Becks regeln des "Simple Design"s

Nach Kent Beck ist ei Desing einfach ("simple") wenn es den folgenden Regeln folgt:

1. Es besteht alle (geschriebenen) Tests
2. Es enthält keinen doppelten Code
3. Es drückt die Absicht des Programmierers aus
4. Es hat so wenig wie möglich Klassen und Methoden

Die Regeln sind ihrer Wichtigkeit nach aufsteigend Nummeriert.

Regel 1 - Es besteht alle Tests

Ein System das nicht Testbar ist kann nicht verfiziert werden. Ein System das nicht verifiziert werden kann sollte nich eingesetzt werden.

Ein System zu erstellen das testbar ist bringt Anforderungen an die Architektur und den Quelltext mit sich. Um die einzelnen Teile des Systems testbar zu machen müssen die Anforderungen aus den vorhergehenden Kapiteln befolgt werden. Software Testbar zu machen führt zu niedriger Koppelung sowie hohe Kohäsion.

Regel 2-4 - Refactoring

Die erste Regel ermöglicht es festzustellen ob vorherige Funktionen durch spätere Änderungen verändert werden. Die Regeln 2-4 können nun durch Refactoring erreicht werden. Eine nicht-saubere Code-Base kann Schritt für Schritt in eine saubere überführt werden.

Dabei sind vorallem folgende Regeln einzuhalten:

- Kein doppelter Code: Doppelter Code macht es schwer die funktionfähigkeit eines System aufrecht zu erhalten.
- Ausdruck der Absicht: Die Absicht des Programmiers sollte im Code, auch ohne Kommentare, ersichtlich sein. Code wird öfter gelesen als geschrieben, hierüber sollte man sich beim Schreiben von Code klar sein.
- so wenig wie möglich Klasse und Methoden: Die richtige Anzahl an Klassen und Methoden zu finden ist schwer, gerade wenn das System dem SRP folgen soll. Man sollte jedoch versuchen die Anforderungen mit so wenig wie möglich Klassen zu

erreichen.

Soft Skills für Softwareentwickler

Zusammenfassung des Buches:

Titel: Soft Skills für Softwareentwickler

Verfasser: Uwe Vigenschow, Björn Schneider, Ines Meyrose

Verlag: dpunkt.verlag

Jahr: 2014

ISBN: 978-3-86490-190-4, 978-3-86491-529-1 (eBook)

Zusammenfassung von: Jonas Wiese, Daniel Beneker, Fabian Lorenz

Teil I Projektarchitektur und Kommunikationsschnittstellen

1 Software- und Projektstruktur

1.1 Komplexität von Projektstrukturen

Bei der Softwareentwicklung können die Projekt und Kommunikationsstrukturen schnell komplex werden. Dabei können sich die Strukturen der Kommunikation und Software ähneln. Allerdings ist die Software oftmals besser strukturiert. Anders als bei Software verläuft die Kommunikation zwischen Menschen nicht verlustfrei. Wenn Menschen Kommunizieren geht immer ein Teil der Information verloren. Der Verlust wird noch höhere, wenn die Kommunikationspartner nicht in ihrer Muttersprache reden. Noch weiter verschlimmert wird das Problem durch Vorurteile und „Schubladen Denken“. Der Effekt verschärft sich besonders, wenn Teammitglieder mit und ohne technischen Hintergrund zusammenarbeiten. Aber die Zusammenarbeit mit diesen Teammitgliedern ist wichtig, weil auch sie und ihre Fähigkeiten für das Projekt wichtig sind. Es muss auch beachtet werden, dass die nicht Entwickler, z.B. Manager, abhängig von den Entwicklern sind. „Sie sind also abhängig von Menschen die sie nicht verstehen.“

1.2 Bedeutung für IT-Projekte

Softwareprojekte sind primär Kommunikationsprojekte. Mit technische Probleme befassen sie sich nur sekundär. Die vier zentralen Prinzipien agiler Softwareentwicklung sind:

- Mut: Darauf vertrauen Probleme von morgen auch morgen zu lösen. Probleme von heute, heute ansprechen
- Kommunikation: Für ein persönliches Verhältnis sorgen und Kommunikationsprobleme sofort lösen
- Feedback: Reviews, Akzeptanztests und die Entwicklung im Team zeigen an, ob man auf dem richtigen Weg ist.
- Einfachheit: Die einfachste Lösung ist immer noch Komplex genug. Auch bei drei ähnlichen Problemen ist es sinnvoll drei verschiedene Lösungen zu entwickeln. Erst beim vierten Mal ist ein generischer Ansatz einfacher.

2 Projektpolitik? Projektumfeldanalyse!

Da bei Softwareprojekte viel Geld im Spiel ist, sind Machtspiele bestimmter Personen unvermeidlich. Auch als Softwareentwickler wird man in diese Spiele automatisch eingebunden. Die Komplexität übersteigt dabei häufig die von Schach, da es mehr Spieler und Interessen gibt. Um trotzdem einen Überblick zu behalten, kann die Projektumfeldanalyse angewendet werden. Mit ihr ist es möglich die Interessen und Bedürfnisse, aber auch die Motivation und Ziele aller Beteiligten zu erfassen. Auch können die Beziehungen der Beteiligten besser verstanden werden. Ziele der Analyse sind zum einen das frühestmögliches Erfassen von Einflussfaktoren, Problemfeldern, Potenzialen, sowie dessen Beurteilung bezüglich des Projekterfolges. Außerdem ist die Optimierung der Kommunikation anhand der Umfeld Beziehungen ein Ziel. Projektmanagement ist ein ganzheitlicher Prozess für den es kein „Kochbuch“ gibt. Die reine Budget- und Ressourcenbetrachtung birgt allerdings große Gefahren. Ein methodisches Grundgerüst können allerdings diese Schritte bilden: Identifikation des Projektumfelds (z.B. Personen mit Hierarchien), Gliederung nach organisatorisch-sozialen Umfeldgrößen und die Ableitung von Strategien und Maßnahmen.

2.1 Was sind Stakeholder?

Stakeholder sind Interessenhalter. Sie haben entweder direkten Einfluss auf das Projektergebnis oder sind von den Projektzielen abhängig. Stakeholder können zum Beispiel Anwender, Fachabteilungen, Kunden, Gesetzgeber oder Käufer sein.

2.2 Stakeholder Elicitation: Wer hat Interessen?

Bei der Stakeholder Elicitation wird versucht die Interessen der einzelnen Stakeholder ans Licht zu bringen. Hierbei wird zuerst eine möglichst vollständige Liste aller Stakeholder und ihrer Interessen erstellt. Anschließend können die Stakeholder gruppiert werden. Die Gruppierung erfolgt anhand von 3 Kriterien:

- „Anforderungsverantwortliche“ (sind befugt Anforderungen festzulegen)
- „Fachexperten“ (Experten eines Fachgebiets)
- „Systembetroffene“ (spätere Anwender) Ein Stakeholder kann zu einer, mehrerer oder keiner dieser Kategorien gehören.

Eine zweite Gruppierung kann nach Umfeldgruppen vorgenommen werden. Hierbei sind vier Kategorien relevant:

- „Promoter und Sponsoren“ (förderen den Innovationsprozess aktiv)
- „Unterstützer und Verändere“ (geben inhaltlichen Rückhalt im Unternehmen)
- „Unentschlossene und Meinungswechsler“ (kein direkter Macht Einfluss und wechseln mit äußeren Einflüssen ihre Meinung)
- „Gegner und Veränderungsbarrieren“ (Personen mit offenem oder verstecktem

Widerstand).

Beide Gruppierungen sind voneinander unabhängig und haben jeweils eigene Vorteile. Um die Stakeholder Analyse zu visualisieren wird die Stakeholder Map genutzt. Hierbei werden die gebildeten Gruppen mit 3 verschiedenen Pfeilen (befreundet, verfeindet oder loyal) mit einander verknüpft.

3 Projektmarketing

Ein Grundproblem der IT ist die Wahrnehmung der IT durch nicht ITler. Sie sehen die IT nur als Mittel zum Zweck und als selbstverständlich an.

3.1 Wie funktioniert Marketing

Projektmarketing hilft beim Erreichen der Projektziele. Für ein erfolgreiches Projektmarketing müssen die Ziele klar definiert sein. Ein Ziel erfüllt folgende Kriterien:

- Operativ einsetzbar
- konkret ausdrückbar
- realisierbar, erreichbar
- nachvollziehbar
- schriftlich fixiert Passend zu den Zielen kann eine Marketingstrategie entwickelt werden. Sie ordnet sich angemessen in das Marketing Dreieck ein (Kunde, Wettbewerb, Unternehmen). Das Marketing spricht die Gefühle der Zielgruppe an und wird deshalb als „Emotionsmarketing“ betrieben.

3.2 Projektbegleitendes Marketing

Projekte dauern oft länger als ein paar Wochen, daher ist es wichtig schon während des Projekt über den Erfolg dessen zu reden. Das schafft Vertrauen bei den Stakeholdern. Die Grundformen lautet daher „Tue Gutes und rede darüber!“. Mit Hilfe dieser Formel wird Misserfolg nicht schöneredet, aber gute Leistung wird besser wahrnehmbar.

3.3 Begeisterungsqualität

Das Kano-Modell (1984 vorgestellt) ist ein Qualitätsmodell aus Kundensicht. In diesem finden sich klassische Qualitätsaspekte (Zuverlässigkeit, Benutzbarkeit, Übertragbarkeit, Änderbarkeit, Effizienz, Funktionalität) wieder. Sie werden jedoch aus Kundensicht beurteilt. Es werden drei Arten von Anforderungen unterschieden:

- Basisanforderungen: Muss das Produkt haben um bestehen zu können
- Eindimensionale Anforderungen: Sie sind direkt mit der Kundenzufriedenheit

verbunden. Je besser diese Anforderungen erfüllt sind, desto zufriedener sind die Kunden

- Attraktive Anforderungen: Sie haben den größten Einfluss auf die Kundenzufriedenheit. Das Fehlen dieser Anforderungen kann nicht zur Unzufriedenheit führen.

Ein Beispiel für Attraktive Anforderungen können kleine Anpassungen sein, die dem Kunden aber positiv Auffallen (anpassen der Tabulatorreihenfolgen an den neuen Arbeitsfluss, oder bearbeiten der voreingestellten Belegungen von Eingabemasken). Das Kano Modell hilft die Zusammenhänge zwischen Anforderungen und Kundenzufriedenheit besser zu erkennen. Wenn die Software also an einigen Stellen Begeisterung weckt, können andere Probleme leichter gelöst werden.

3.4 Events und Präsentationen

„Projektmarketing heißt aktive Selbstdarstellung“. Dazu ist es nötig regelmäßig den Stand der Entwicklung zu präsentieren. Auch das Feiern eines erreichten Meilensteins ist hierbei wichtig. Schon ein kurzes Event (15-30 Minuten) reicht aus, um einen positiven Eindruck zu hinterlassen. Bei der Präsentation der Ergebnisse ist es wichtig technische Details nicht zu erläutern, da „Nicht-Entwickler“ diese nicht nachvollziehen können. Es ist wichtiger, sichtbare Ergebnisse zu präsentieren und auf Nachfrage technische Details erläutern zu können. Bei der Entwicklung mit Scrum fällt dem am Ende einer Iteration durchgeföhrte Reviewmeeting eine wichtige Bedeutung zu. Es sind alle Stakeholder anwesend, deshalb können diese Vertrauen in das Projekt aufbauen und Feedback geben. Eine gute Vorbereitung und straffe Organisation durch den Scrum-Master ist wichtig.

Teil II Mit Fragetechniken zu besseren Informationen

4 Grundlegende Fragetechniken

4.1 Informationsfragen

ITler stellen meist Fragen um Probleme zu Analysieren. Dazu gibt es offene und geschlossene Fragen. Geschlossene Fragen können mit Ja oder Nein beantwortet werden. Offene Fragen verlangen eine strukturierte Antwort, die über Ja oder Nein hinausgeht. Um die Intention hinter einer Softwareanforderung zu verstehen ist die „Warum“ Frage gut geeignet. Meistens reicht aber eine direkte Frage nach der Anforderung nicht aus, sodass weiter gefragt werden muss. Es ist daher nötig mehrere „Warum“ Fragen hintereinander zu stellen. Das genaue Nachfragen mit „Warum“ kann bis zu fünf Mal erfolgen. Danach sind meist keine Sinnvollen Antworten mehr zu erwarten. Eine „Warum“ Frage kann aber auch dazu führen, dass sich der Befragte angegriffen fühlt. Daher kann es Sinnvoll sein, die Frage zu umschreiben und das direkte „Warum“ zu vermeiden. Dadurch kann eine Verteidigungshaltung des Befragten vermieden werden.

4.2 Mit Fragen auf den Punkt kommen

Um die Thematik einzuschränken können geschlossene Fragen oder „Alternativfragen“ (Welche der vorgestellten Alternativen erscheint Ihrer Meinung nach sinnvoll) genutzt werden. Sie lassen nur wenige Antwortmöglichkeiten zu und treiben damit die Ergebnisfindung voran. Mit einer geschlossenen Frage kann ein „Commitment“ erreicht werden (Sind Sie mit der Lösung zufrieden). Dadurch kann eine Diskussion vermieden werden. Mit Fragen kann ein Gespräch also Gelenkt werden, allerdings ist vor dem Gespräch nicht immer klar wohin. Durch Rückfragen können Zusammenhänge und Details festgestellt werden. Hier wird zwischen „konstruktiven“ und „destruktiven“ Rückfragen unterschieden. Mit konstruktiven Rückfragen kann eine „Lösungsebene“ erreicht werden (z.B. „Was brauchen Sie, um das Problem zu lösen“). Mit einer destruktiven Rückfrage verbleibt man weiter auf der „Problemebene“ (z.B. „Wie ist das Problem Entstanden“). Destruktive Fragen sind in die Vergangenheit gerichtet, wohin gegen Positive Rückfragen eher in die Zukunft gerichtet sind. Es ist auch möglich mit Fragen zu motivieren oder zu provozieren. Beide Mittel können in der richtigen Situation das Gespräch weiter

voranbringen. Provozierenden Fragen (z.B. „Ist das schon alles was Ihnen dazu einfällt“) rufen aber nicht immer eine positive Reaktion hervor. Sie müssen wohl überlegt geäußert werden.

5 Die Sechs-Stufen-Fragetechnik

Wie bereits beschrieben kommt es beim Kommunizieren immer zu einem Verlust von Informationen. Um trotzdem an alle Informationen zukommen kann die Sechs-Stufen-Fragetechnik helfen. Sie setzt sich aus diesen Schritten zusammen:

- „Prozesswörter überprüfen“: Identifikation von Verben - W Fragen stellen („wer/was/wann melden“)
 - „Komparative und Superlative überprüfen“: Bezugspunkte bei Vergleichen bestimmen. Falls möglich Messmethoden nutzen um Überprüfbarkeit zu schaffen
 - „Universalquantoren überprüfen“: Universalquantoren (jeder, keiner, alle, immer) finden und hinterfragen bzw. Ausnahmen finden
 - „Bedingungen überprüfen“: Sind alle Varianten aufgeführt, gibt es noch andere Verzweigungsbedingungen?
 - „Konstanten und konfigurierbare Werte überprüfen“: Feste sowie konfigurierbare Größen finden und richtig benennen.
 - „Abkürzungen und Fachbegriffe im Glossar definieren“: Abkürzungen, Akronyme und Fachbegriffe aufzählen und erklären

Die Schritte bilden einen Gesprächsleitfaden, der dabei helfen soll, sowohl Grundlagen, als auch Details herausfinden zu können.

5.3 Fragetechniken in Reviews anwenden

Reviews dienen dazu, eine Vorstufe oder ein Zwischenergebnis, aber keine fertige Software zu prüfen. Sie können von hohem Nutzen sein, da sie:

- Ungenauigkeiten in den Anforderungsdokumenten identifizieren
- Einen regen Wissensaustausch zwischen den beteiligten Personen fördern
- Fehler frühzeitig gefunden werden können

Allerdings müssen Reviews gut vorbereitet sein, da sie sonst keinen Nutzen haben und Zeit verschwenden. Um das Review voran zu bringen ist es sinnvoll erst die positiven Aspekte herauszustellen. Die Kritischen Punkte können mit Hilfe von Offenen Fragen angesprochen werden. Dabei ist es wichtig direkte „Du“ Aussagen zu vermeiden („Du hast vergessen...“). Genauso sollten direkte Bewertungen zurückgehalten werden („das klappt so nicht“).

5.4 Anregende Fragen

Kreative Fragen, die die Perspektive wechseln, können helfen festgefahrenen Gespräche zu vermeiden. Ein solcher Perspektivwechsel kann zum Beispiel durch eine Kontrollfrage ausgelöst werden („Was würde Ihr Chef denn antworten?“). Sie sind besonders nützlich, wenn der Gesprächspartner auf eine Frage keine Antwort findet. Wenn der Wechsel der Perspektive nicht hilft, können „Prozessfragen“ genutzt werden. Eine Prozessfrage ist z.B. „Was brauchen Sie um die Frage beantworten zu können?“.

6 Feedback und aktives Zuhören

6.1 Warum überhaupt Feedback geben?

Feedback kann Verhältnisänderungen bewirken und damit die Gruppendynamik stärken. Allerdings muss es dazu richtig eingesetzt werden, da es sonst starke negative Folgen haben kann. Ob das Feedback Aussicht auf Erfolg hat, hängt von zwei Fakten ab. Zum einen Wertschätzende Beziehung vorliegen, zum anderen müssen die Feedbackregeln eingehalten werden. So erfolgt das Feedback in einer „Sandwich-Technik“, indem in negativer Aspekt zwischen zwei positiven verpackt wird. Zu beachten ist, dass Feedback nur dann eingesetzt werden darf, wenn es auf einen Aspekt abzielt, der auch durch den Feedback-Empfänger beeinflussbar ist.

6.2 So funktioniert es: Feedback-Regeln

Insgesamt lassen sich 7 Feedback-Regeln für den Sender definieren:

- Blickkontakt
- Fragen ob Feedback erwünscht ist
- Konkrete Beobachtungen artikulieren: Ich Botschaften sowie Aussagen zum Verhalten, nicht zur Person direkt
- Sandwich- Technik: 2x positiv, 1x negativ
- Empfindung als „Ich Botschaften“ benennen
- Gegebenenfalls Wünsche anfügen
- Nur auf veränderbare Verhaltensweisen eingehen

Für den Empfänger gibt es 6 Regeln:

- Bereitschaft für Feedback prüfen
- Still zuhören, nur Verständnisfragen stellen
- Sender Ausreden lassen, nicht rechtfertigen
- Über das Feedback nachdenken

- Für das Feedback danken
- Selbst entscheiden, was man mit dem Feedback macht – berücksichtigen oder ignorieren

6.3 Aktiv zuhören: Verluste minimieren

Der Zuhörer fasst alle paar Sätze das Gehörte zusammen und wiederholt es mit seinen eigenen Worten. So kann der Sender korrigierend eingreifen und Missverständnisse vermeiden. Diese Technik kann sich besonders bei Feedback oder bei Softwareanalyse mit einem Fachbereichsmitarbeiter lohnen.

6.4 In kritischen Situationen auf die Meta-Ebene

Falls trotzdem eine Situation entsteht, in der Informationen verloren gegangen sind und es deshalb zu einem Missverständnis kam, hilft es auf die Meta-Ebene zu wechseln. Das bedeutet, dass nicht mehr über konkrete Problem, sondern über die Art der Kommunikation gesprochen wird. Beispiel: „Das nervt mich total, dass du so zögerlich bist und keine Entscheidungen triffst“ in der Antwort ist es möglich auf die Meta-Ebene zu wechseln. So wäre eine mögliche Antwort „Augenblick mal, du verletzt gerade die Feedback-Regeln: Nur Ich-Botschaften sind erlaubt“. Diese Antwort zieht zum einen eine Entschuldigung des Feedback-Senders nach sich, zum anderen aber auch eine Konkretisierung des Problems („Entschuldige, ich meine, durch die ganzen ausstehenden Entscheidungen und offenen Punkten fühle ich mich ganz verunsichert!“).

4 IT-Kommunikationstypen

Entwickler-Kommunikationstypen

Der No-Future-Entwickler

Dieser Typus hat meistens ein großes Motivationsproblem. Er hat keine Lust auf irgendwas und träumt von einem Job, der seine Arbeit anerkennt. Er ist oft ironisch und zynisch und verbreiter negativer Nachrichten. Zu seinen Stärken zählen das Erledigen von wiederkehrenden Aufgaben. Mit Gleichgesinnten baut er oft tiefe Freundschaften auf. Einer seiner Schwächen neben der fehlenden Motivation ist, dass er die Verantwortung oft bei jemand anderem sucht und sich schwer in Teams integrieren lässt. Wer mit diesem Typus zusammenarbeitet, sollte nicht auf seine Ablenkungen eingeben und gleichzeitig versuchen ihn möglichst wenig zu drängen. Auch die Wertschätzung seiner Arbeit hilft bei der Zusammenarbeit. Die Führung eines solchen Typus sollte kooperativ ablaufen. Das wichtigste ist, nicht demotivierend zu wirken!

AAAA - der allwissende, allgegenwärtige, arrogante Architekt

Der Architekt arbeitet oft an Themen die nur für ihn relevant sind bzw. die andere nicht verstehen oder die viele einfach nicht interessieren. Er diskutiert gerne überall mit und wird deswegen schnell als arrogant abgestempelt. Zu seinen Stärken zählt sein breites technisches Wissen. Er ist stets motiviert und kann auch bei großen komplexen Sachverhalten den Überblick behalten. Außerdem ist er in der Lage sich schnell in neuste technische Entwicklungen einzulesen und diese auch zu integrieren. Obwohl er ein breites technisches Wissen hat, sieht er oft nicht das große Ganze, sondern verliert sich in technisches Details. Weitere Schwächen sind seine mangelnde soziale Kompetenz und das schlechte Ansehen vor Kollegen. Bei der Zusammenarbeit sollte auf eine offene Kommunikation geachtet werden. Bedenken sollten ausgesprochen werden und bei Verständigungsproblemen sollte nachgehakt werden. Die Führung sollte versuchen das Ansehen des Architekten vor seine Kollegen zu verbessern. Bei der Kommunikation sollte man sich ausreichend Zeit nehmen und die Ideen würdigen. Eine weitere Möglichkeit können auch Seminare sein, die die zwischenmenschliche Seite des Typus stärkt.

XXPlor - der eXtreme eXtreme Programmer

Dieser Typus ist stark ergebnisorientiert. Er programmiert direkt drauf los, verliert aber nach spätestens 80% der Arbeit die Motivation. Er legt anfangs eine ungeheure Arbeitsgeschwindigkeit an den Tag, bringt am Ende aber kaum etwas zu Ende. Seine größte Stärke ist, dass er in Krisenzeiten schnell und effektiv helfen kann. Allerdings ist auch zu beachten, dass er oft auch die Ursache ist. Denn seine größte Schwäche ist, dass er nichts komplett fertig bekommt. Aufgaben wie Programmtests müssen dann oft von Kollegen erledigt werden. Außerdem ist er kein Experte in Kommunikation sowie in Konfliktmanagement. Die Zusammenarbeit mit diesem Typus ist nicht einfach. Arbeiten, die er Kollegen überlässt, sollten nicht einfach übernommen werden. Stattdessen sollte er auf den Umfang seiner Arbeit hingewiesen werden. Als Führungsfigur sollte man diesen Typus offen auf seine Stärken und Schwächen ansprechen. Zudem kann es helfen, eine vollständige Lösung zu verlangen.

Der Hacker

Der Hacker lässt sich als zurückgezogener Technikfreak beschreiben. Er mag keine Hierarchien und missachtet schnell Vorschriften. Technisch hingegen ist er sehr interessiert und verfügt über sehr tiefgehendes Wissen. Dieses Wissen teilt er allerdings auch nur mit denen, denen er absolut vertraut. Im Allgemeinen sucht dieser Typus Anerkennung. Zu seinen Stärken zählen seine Zielstrebigkeit und seine Ausdauer. Dies kann besonders hilfreich sein in Krisen. Seine Schwächen sind seine eingeschränkte Teamfähigkeit und die Möglichkeit Vorschriften zu akzeptieren. Eventuell ergibt es auch Sinn, dass dieser Typus in den Betriebsrat oder die Forschung wechselt oder zum Sicherheitsexperten benannt wird. In diesen Positionen kann sich der Hacker unter Umständen besser ausleben.

Mr. 120%

Mr. 120% macht keine halben Sachen. Er ist Perfektionist durch und durch. Spezifikationen müssen bis ins kleinste ausgearbeitet worden sein, ansonsten wird dieser Typus nachfragen. Die Vollständigkeit ist auch seine größte Stärke, besonders in den letzten Phasen eines Projektes, wo andere die Motivation verlieren oder Aspekte übersehen. Meistens ist dieser Typus ruhig und hilfsbereit. Er kann jedoch schlecht mit Zeitdruck umgehen und hat Probleme damit den Aufwand abzuschätzen, da er immer Angst hat, etwas zu übersehen und seine Aufgabe nicht vollständig zu erledigen. Bei der Zusammenarbeit sollte darauf geachtet werden, dass Aufgaben klar abgegrenzt werden und eine genaue Verteilung vorgenommen wird. Die Führung sollte versuchen keine Fragen zu stellen, die bei diesem Typus Unsicherheit auslösen. Dazu gehören alle Fragen, auf die er keine absolut sichere Antwort geben kann, wie beispielsweise die Frage nach dem Zeitaufwand. Ansonsten sollte klargemacht werden, dass keine genaue Antwort benötigt wird.

Kommunikationstypen in den Fachbereichen

Der bessere Verkäufer

Dieser Kommunikationstyp will überzeugen, seine Meinung verkaufen. Dies tut er, indem er viel erzählt und oft in Monologe verfällt. Die Wünsche des Kunden kommen dabei oft zu kurz, da dieser Typus seine eigene Meinung durchsetzt. Seine Stärken sind die Rhetorik und sein breites Wissen, oft auch über andere Fachgebiete hinweg. Seine größte Schwäche ist sein gering ausgeprägtes Einfühlungsvermögen. Er merkt nicht, wann andere ihm nicht mehr zuhören und er mit seinem Monolog übertreibt. Ein weiteres Problem ist, dass er sich nicht so gerne in fachliche Tiefe begibt und Anforderungen an die Entwicklungsabteilung entsprechend ungenau abliefer. Dazu kommt, dass das was der Kunde eigentlich wollte, nicht enthalten ist. Wenn mit diesem Typus zusammen gearbeitet wird, ist es wichtig, den eigenen Standpunkt zu vermitteln und sich nicht durch Monologe tot reden zu lassen. Als Führungsperson sollte auf direktes Feedback geachtet werden. Zusätzlich sollten in Besprechungen stillere Kollegen dazu motiviert werden, ebenfalls ihre Meinung zu äußern. Eine weitere Möglichkeit ist das Anbieten von Kommunikationsseminaren.

Der zurückgezogene Spezialist

Der zurückgezogene Spezialist ist immer überlastet, denn er steht vor einem großen Berg an Aufgaben, den er nicht beherrschen kann. Er findet immer neue Aufgaben und schließt alte daher nicht komplett ab. Die größten Stärken dieses Typus sind die Vollständigkeit und die Ausdauer. Der Drang nach Vollständigkeit ist jedoch auch seine Schwäche, denn er findet immer neue Baustellen und schließt seine Aufgaben daher nie komplett ab. Zusätzlich hat er leichtes Kommunikationsdefizit. Wenn mit diesem Typus zusammen gearbeitet wird, sollten Deadlines definiert werden. Zudem sollte klar gesagt werden, was bis dahin enthalten ist. Außerdem ist es von Vorteil zwischendurch den aktuellen Stand zu erfragen, bzw. den Kontakt aufrecht zu erhalten. Als Führungsperson sollte man immer auf Ergebnisse bestehen und Probleme offen ansprechen. Zur Not kann beispielsweise mit dem Projektplan Druck ausgeübt werden.

Der Konzepteklopfer

Dieser Typus ist außerordentlich selbstbewusst und versucht sich immer durchzusetzen, auch bei Kunden. In seinem Fachgebiet ist er ein Profi und hat entsprechende Dokumente wie Konzepte, Spezifikationen, etc... ausgearbeitet. Diese stellt er auch gerne der breiten Masse vor. Seine Stärken bestehen darin, dass er besonders gut im Überzeugen ist und darin auch viel Ausdauer an den Tag legt. Durch sein tiefgreifendes Wissen ist er auch in der Lage seine Arbeit kritisch zu hinterfragen und fremde Konzepte zu überarbeiten. Hierbei ist

er sehr gründlich. Seine Schwäche ist sein mangelndes Einfühlungsvermögen. Seinen Gesprächspartner bezieht der daher selten mit ein. Er drängt ebenfalls nach Vollständigkeit und hat deshalb Probleme seine Aufgaben komplett abzuschließen. Oft endet er in viel zu komplexen Strukturen. Kollegen sollten stets darauf achten, dass ihre Lösung mit einbezogen wird. Stehen zwei Meinungen gegeneinander, sollte immer auf der Sachebene diskutiert werden. Als Führungsposition besteht die Möglichkeit an Abstimmungen teilzunehmen und dafür zu sorgen, dass es zu konstruktiven Diskussionen kommt. Generell ist viel Feingefühl gefordert.

Der Visionär

Als Visionär ist die Gegenwart bereits Vergangenheit. Er lebt in der Zukunft. Für ihn sind aktuelle Probleme bereits gelöst, deshalb löst er zukünftige Probleme, obwohl diese Lösungen noch garnicht benötigt werden. Das zukunftsorientierte Denken ist seine größte Stärke. Fragen wie "Ist ein bestimmtes System zukunftssicher?" sind bei ihm genau richtig. Er ist in der Lage in der Gegenwart Muster zu erkennen und zusammen mit seiner Erfahrung Prognosen für die Zukunft zu stellen. Seine Schwäche ist, die Gegenwart zu vernachlässigen. Oft schließt er Aufgaben in der Gegenwart nur zum Teil ab. Bei der Zusammenarbeit sollten seine Stärken genutzt werden. Andernfalls sollten sie deutlich machen, dass sie die Antwort benötigen um das Projekt weiter voranzutreiben oder den Vorgesetzten einschalten. Als Führungsperson sollten sie auf Konsequenzen für das Projekt aufmerksam machen.

Projektleiter-Kommunikationstypen

Der freundliche Kollege

Dieser Kommunikationstyp trägt den Namen "Der freundliche Kollege", da er dies früher tatsächlich war. Er hatte gute Arbeit geleistet und wurde zum Projektleiter befördert. Das funktionierte anfangs auch ganz gut. Doch dieser Typus erhielt nie eine passende Weiterbildung zum Projektleiter. Er ist daher schnell überfordert, wenn es darum geht Entscheidungen zu treffen und komplexe Sachverhalte zu erfassen. Seine Kollegen kennen ihn gut und versuchen ihn zu unterstützen. Auf Dauer mangelt es jedoch an klarer Führung und jeder Mitarbeiter macht es er für richtig hält. Der "freundliche Kollege" wiederum erkennt, das er kaum Einfluss auf das Projekt hat, was ihn nach und nach demotiviert. Letztendlich versagt er oft als Projektleiter und wird vom Upper-Management wieder degradiert. Dieser Typus hat jedoch auch gewisse Stärken. Als ehemaliger Entwickler kann er bestens im Team agieren. Er versucht immer alle Meinungen zu berücksichtigen und versucht auch die stilleren Kollegen zu motivieren. Er hat ein tiefes fachliches Wissen und hat sich mit seiner bisherigen Arbeit im Unternehmen bereits bewiesen. Seine Schwächen

liegen vor Allem in der Kommunikation und der Eigeninitiative sich über die Aufgaben eines Projektleiters vorzubilden. Als Mitarbeiter ist es wichtig, diesem Typus klar zu machen, dass er Entscheidungen treffen muss. Trifft er keine Entscheidung, kann zur Not eine eigene Lösung umgesetzt werden und ihm mitgeteilt werden. Die Führungsperson dieses Typus sollte sehr genau auf den Fortschritt des Projektes achten. Es empfiehlt sich, anfangs Meilensteine zu vereinbaren. Werden diese Meilensteine nicht eingehalten, sollten sie sich möglichst früh informieren lassen. Außerdem sollte gezielt auf Problemen in der Kommunikation zwischen diesem Typus und seinen Mitarbeitern geachtet werden. Auch hier kann es hilfreich sein, Schulungen zur Gruppendynamik oder zum Konfliktmanagement anzubieten.

Der Choleriker

Dem Choleriker ist nichts zu kompliziert. Er kann immer ein Problem in kleine Schritte aufteilen und ist daher auch besonders beliebt beim Upper-Management. Dies sind auch seine größten Stärken. Diese Schritte dürfen jedoch nie in Frage gestellt werden, denn wenn eine gewisse Schwelle überschritten ist, kommt es bei dem Choleriker zu einem Wutausbruch. Dadurch verlieren oft die Mitarbeiter das Vertrauen zu dieser Person und informieren ihn nicht mehr ausreichend. Der Choleriker wiederum ist dadurch auch nicht umfassend informiert, was dem Upper-Management irgendwann auffällt. Ein weiterer Nachteil dieses Typus ist neben den Wutausbrüchen, die eingeschränkte Sicht auf zukünftige Ereignisse. Er kann jederzeit die nächsten Schritte diktieren, hat jedoch keinen Blick für das große Ganze. Außerdem entscheidet er oft voreilig. Bei der Arbeit mit diesem Typus, sollte versucht werden, ihn zum nachdenken zu bringen. In einem Gespräch könne beispielsweise gezielt Risiken angesprochen werden. Dies kann helfen, dass der Choleriker nicht voreilig entscheidet. Als Führungsperson sollte darauf geachtet werden, dass dieser Typus nicht alles für seine Mitarbeiter entscheidet.

Der formale Prozessler

Dieser Typ eines Projektleiters liebt die vollständige Sicherheit. Er versucht für alles Prozesse bis ins kleinste Detail zu entwickeln. Das Testen einer Software ohne eine ganz spezifische Strategie ist beispielsweise für ihn nicht denkbar. Gefühle haben für ihn neben den Prozessen keinen Platz. Als formaler Prozessler schafft er ein sicheres Projektumfeld. Einzelne Projektphasen werden gründlich abgearbeitet und im Projektverlauf gibt es kaum Probleme. Im Umgang mit seinen Mitarbeitern agiert dieser Typus ruhig und gelassen. Zudem schafft er es, äußere Aspekte, wie Geldmangel oder Zeitmangel und den dadurch entstehenden Druck von den Mitarbeitern fernzuhalten. Das streben nach vollständiger Sicherheit ist die größte Schwäche dieses Typus, denn sie ist in der Realität nicht erreichbar. Durch die vielen formalen Prozesse ist sein Team meistens weniger effizient, als andere. Bei

der Zusammenarbeit sollte immer hinterfragt werden, ob mit einem bestimmten Prozess eine Aufgabe wirklich schneller und effizienter umgesetzt werden kann. Es ist wichtig Bedenken offen zu äußern. Als Führungsperson sollten eventuelle Schwächen bei der Effizienz des Teams offen angesprochen werden. Außerdem kann es hilfreich sein, diesem Typus zu zeigen, wann eine bestimmte Abgeschlossen ist.

5 Konfliktmanagement

Konflikte analysieren

Konfliktbeschreibung

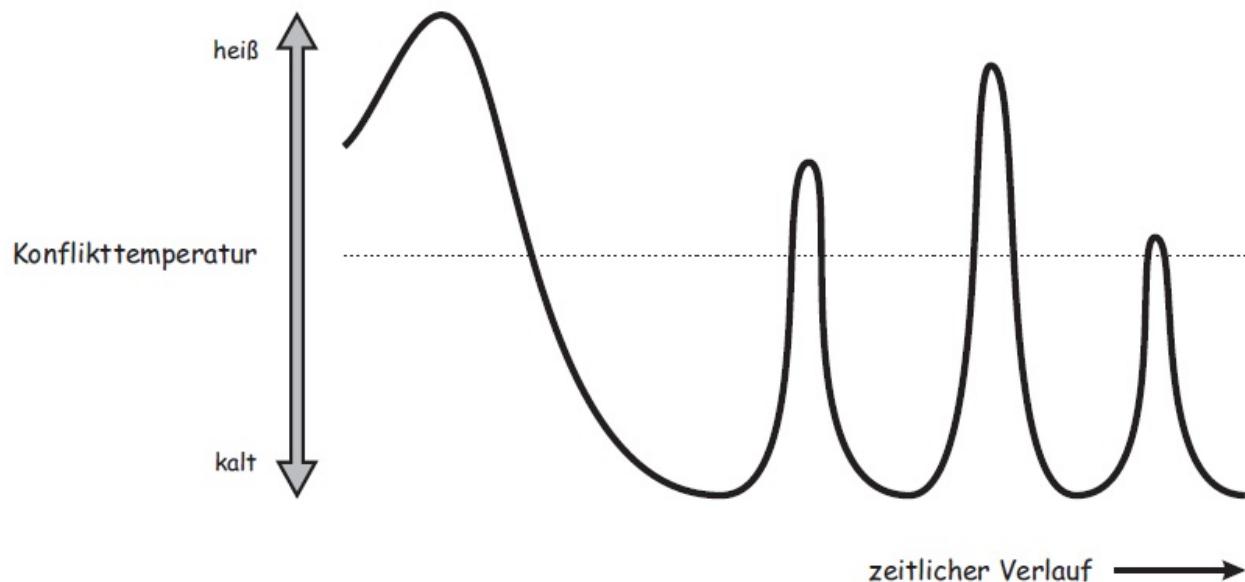
Ein Konflikt im IT-Bereich lässt sich als spezielle soziale Beziehung definieren, die folgende Eigenschaften besitzt:

- Es gibt entgegengesetzte, widersprüchliche Ziele, Interessen oder Handlungsweisen
- Es sind zwei oder mehr voneinander abhängige Personen daran beteiligt
- Die Beteiligten erkennen zum Zeitpunkt des Konflikts keine sofortige Lösung

Innerhalb von IT-Projekten kommt es sehr häufig zu Konflikten. Einschränkende Rahmenbedingungen, eintretende Risiken, Zeitdruck und gegebenenfalls unterschiedliche Zielvorstellungen von Stakeholdern führen zu Konflikten im Team, sowie mit bzw. zwischen den Stakeholdern. Auch die Auswirkungen eines Projekts sind Auslöser von Konflikten, da beispielsweise die Arbeitsbedingungen der Anwender verändert werden, oder im Extremfall durch neue Software sogar Arbeitsplätze wegfallen. Da für den Erfolg eines Projekts die Zusammenarbeit mit den Anwendern extrem wichtig ist, sind solche Konfliktpotentiale äußerst risikoreich. Konflikte helfen aber auch dabei, Unterschiede offenzulegen. Ein konstruktiver Umgang mit Konflikten ist daher essenziell für den Projekterfolg.

Heiße und kalte Konflikte

Konflikte lassen sich in zwei Formen einteilen: heiße und kalte Konflikte. Heiße Konflikte sind laut, die Konfliktparteien sind (über-)motiviert und suchen die direkte Konfrontation untereinander. Kalte Konflikte sind dagegen unterschwellig, es herrscht Frustration und Enttäuschung und der direkte Kontakt wird von den Konfliktparteien gemieden.



Die Temperatur eines Konflikts verändert sich im Laufe der Zeit. Von außen sichtbar ist er dabei nur oberhalb der gestrichelten Linie, wenn er auflodert.

Konflikte im kalten Zustand sind schwer zu erkennen und zu lösen. Es empfiehlt sich daher zu warten bis der Konflikt wieder in eine heiße Phase übergeht, eventuell ihn sogar selbst anzuhühen. Im heißen Zustand ist der Konflikt dann wieder zugänglicher, man kann ihn angehen und letztendlich lösen.

Konfliktarten

Sachkonflikt

Unterschiedliche Meinungen treffen im Rahmen einer sachlich geführten Diskussion aufeinander.

- Ursache: Wegen fehlender Informationen, unterschiedlicher Kenntnisse und Erfahrungen der beteiligten Personen oder durch Missverständnisse werden unterschiedliche Wege beim Lösen eines Problems gesehen.
- Beispiel: Designentscheidungen können häufig wegen fehlender Informationen nur auf Basis der eigenen Erfahrung getroffen werden. Dabei kann es beispielsweise bei der Auswahl eines geeigneten Patterns zu einem Sachkonflikt kommen.
- Lösungsweg: Das Informationsdefizit muss abgebaut werden. Es müssen Informationen gesammelt werden und alle Beteiligten auf den gleichen Stand gebracht werden. Dazu werden offene Fragen gestellt und aktives Zuhören genutzt. Häufig ist auch das prototypische Ausprobieren ein erfolgreicher Lösungsweg im IT-Bereich. Somit können konkrete Informationen schnell beschafft werden und alle Szenarien, die auf ungenauen Informationen beruhen, können eliminiert werden.

Beziehungskonflikt

Die Beziehung zwischen zwei Personen ist gestört und führt zu einer einseitigen oder gegenseitigen Abwertung.

- Ursache: Vorurteile, Ängste, mangelnder Respekt und nicht ausreichende Wertschätzung.
- Beispiel: Das teilweise immer noch existierende Vorurteil gegenüber Frauen und Technik oder Vorurteile aufgrund von Altersunterschieden.
- Lösungsweg: Aufbau von gegenseitigem Respekt, Wertschätzung und Verständnis.

Interessenkonflikt

Dem Verhalten der Konfliktpartner liegen unterschiedliche Interessen zugrunde, die nicht direkt geäußert werden, sondern hinter Positionen versteckt sind.

- Ursache: Es bestehen unterschiedliche Interessen, Diese werden jedoch nicht offen geäußert, sondern vordergründige Positionen werden ausgetauscht.
- Beispiel: Bei Gehaltsverhandlungen werden immer nur Forderung und Gegenposition genannt und wiederholt: "Ich will 300€ mehr im Monat!" "Ich kann Ihnen aber nur 50€ bieten!".
- Lösungsweg: Über das Erkennen und Verstehen der tatsächlichen Interessen wird versucht ein Lösungsweg zu finden. Eine offene und vertrauensvolle Umgebung kann dabei hilfreich sein, diese Interessen zu erkennen. Im Beispiel der Gehaltsverhandlung muss erkannt werden, warum der Arbeitnehmer mehr Gehalt möchte, beispielsweise vielleicht weil er mehr Geld für seinen Lebensunterhalt braucht, oder weil seine Kollegen mehr verdienen und er sich ungerecht behandelt fühlt. Beim Arbeitgeber können begrenzte Mittel eine Rolle spielen oder vielleicht hängt seine Prämie daran, möglichst niedrige Gehaltsabschlüsse zu erzielen. Sind die tatsächlichen Interessen identifiziert, kann ein Interessenausgleich verhandelt werden.

Wertekonflikt

Als Wert wird bezeichnet, was eine einzelne Person als wichtig oder lohnend einzuschätzen gelernt hat. Dies können Lebensprinzipien sein, oder Ziele die man erreichen möchte. Ein Wert ist also recht unabhängig von konkreten Situationen oder anderen Personen. Auch ändern sich die Werte einer Person oder einer Gesellschaft mit der Zeit.

- Ursache: Unterschiedliche, dauerhafte Wertvorstellungen treffen aufeinander
- Beispiel: Zwei Entwickler teilen sich einen Arbeitsplatz, z.B. weil sie beide halbtags arbeiten. Der eine möchte den Schreibtisch stets frei und aufgeräumt haben, der andere möchte seine Arbeitsmittel verteilt und stets zugreifbar auf der Schreibtischfläche

verstreut haben.

- Lösungsweg: Die gegenseitigen Wertvorstellungen müssen von der Beteiligten offengelegt und anerkannt werden. Ziel ist dabei die Unterschiedlichkeit zu erkennen und zu akzeptieren. Da Wertvorstellungen nicht kurzfristig verändert werden können, ist es nur möglich Regeln aufzustellen, die von den Konfliktparteien akzeptiert und eingehalten werden müssen, sodass eine Zusammenarbeit weiterhin erfolgen kann. Im Beispiel könnten sich die beiden Personen darauf einigen, dass jeder einen bestimmten Teil des Schreibtisches bekommt. Ist dies nicht dauerhaft möglich, können nur noch andere Wege gefunden werden, in denen dieser Wertkonflikt nicht mehr auftritt.

Rollenkonflikt

Rollenkonflikte entstehen durch unterschiedliche Anforderungen, die von verschiedenen Seiten an einer Rolle gestellt werden. Sie sind damit unabhängig von der tatsächlichen Person, die diese Rolle einnimmt.

- Ursache: Eine Person hat eine Rolle inne, an die unterschiedliche Erwartungen gestellt werden.
- Beispiel: Mittlere Führungskräfte bekommen Anforderungen von der Geschäftsführung, sowie von den ihr unterstellten Mitarbeitern. Beide haben dabei meist unterschiedliche Erwartungen.
- Lösungsweg: Rollenkonflikte können meist nicht vollständig aufgelöst werden. Stattdessen sollten die unterschiedlichen Erwartungen offengelegt und transparent priorisiert werden. Somit werden die Entscheidungen zumindest nachvollziehbar. Auch kann es oft helfen die Kompetenzen der beteiligten Rollen zu verteilen bzw. zu optimieren.

Ein weiteres bekanntes Beispiel für Rollenkonflikte in der IT sind dabei die Konflikte zwischen Entwicklern und der (externen) Qualitätssicherung. Beide verfolgen zwar dasselbe Ziel, die Software zu verbessern, allerdings kann beispielsweise durch schlechte Kommunikation von Fehlern die Situation schnell verschlechtert werden. Entwickler können sich stark mit ihren geschriebenen Quellcode identifizieren und sich durch Kritik am Quellcode persönlich angegriffen fühlen.

Systemkonflikt

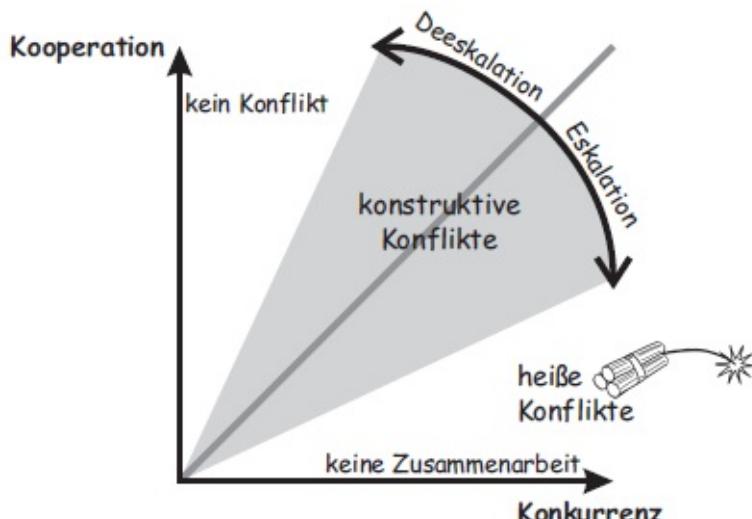
Bei einem Systemkonflikt wirken Konflikte von außerhalb der eigentlichen Gruppe, also aus dem umgebenden System auf die Gruppe ein. Ein Systemkonflikt kann daher nicht innerhalb der eigentlich betrachteten Gruppe aufgelöst werden.

- Ursache: Aufgrund äußerer Engpässe oder anderer, widersprüchlicher Rahmenbedingungen kommt es zu Konfliktsituationen innerhalb einer Gruppe.

- Beispiel: In der Entwicklungsabteilung haben zwei Mitarbeiter gekündigt, das derzeit laufende Projekt ist jedoch von äußerster Wichtigkeit für die Existenz des Unternehmens. Aufgrund einer Unternehmensweiten Umstrukturierung herrscht jedoch ein genereller Einstellungsstopp für neue Mitarbeiter und Mittel für die Beauftragung von externen Mitarbeiter steht auch nicht zur Verfügung.
- Lösungsweg: Die Situation kann im Rahmen des Projektmanagements nur selten aufgelöst werden, da die vorhandenen Reserven meist nicht ausreichend für solche Extremsituationen sind. Es kann daher nur Klarheit über die Zusammenhänge geschaffen werden und der Konflikt an die Stelle delegiert werden, die ihn auflösen kann. Für die Akzeptanz der übergeordneten Instanz ist es wichtig, im Rahmen der Eskalation auch mehrere, alternative und konkrete Lösungsszenarien zu benennen.

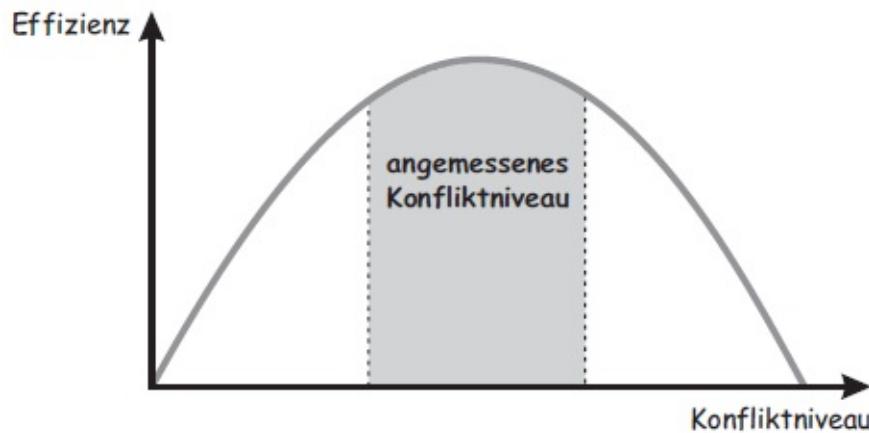
Konfliktdynamik

Konflikte können selbst in den bestmöglich funktionierenden Gruppen entstehen. Dies muss jedoch nicht negatives sein, denn Konflikte können dabei helfen die Entwicklung weiterzubringen und kreative Lösungen hervorzubringen. Dafür müssen sie jedoch konstruktiv sein. Zu viel Kooperation und damit ein nicht vorhandenes Konfliktpotential führt zu einfallslosen Lösungen. Zu viel Konkurrenz führt dazu, dass jeder nur noch für sich selbst arbeitet und somit nie das bestmögliche Potential der Gruppe ausgeschöpft werden kann.



Konstruktive Konflikte entstehen aus dem Wechselspiel von Kooperation und Konkurrenz der beteiligten Personen

Es muss also ein angemessenes, konstruktives Konflikt niveau erreicht werden, um die bestmöglichen Lösungen zu finden. Dafür ist es notwendig, dass die Zusammenarbeit mit gegenseitiger Wertschätzung und einem gemeinsamen Zielkonsens stattfindet.

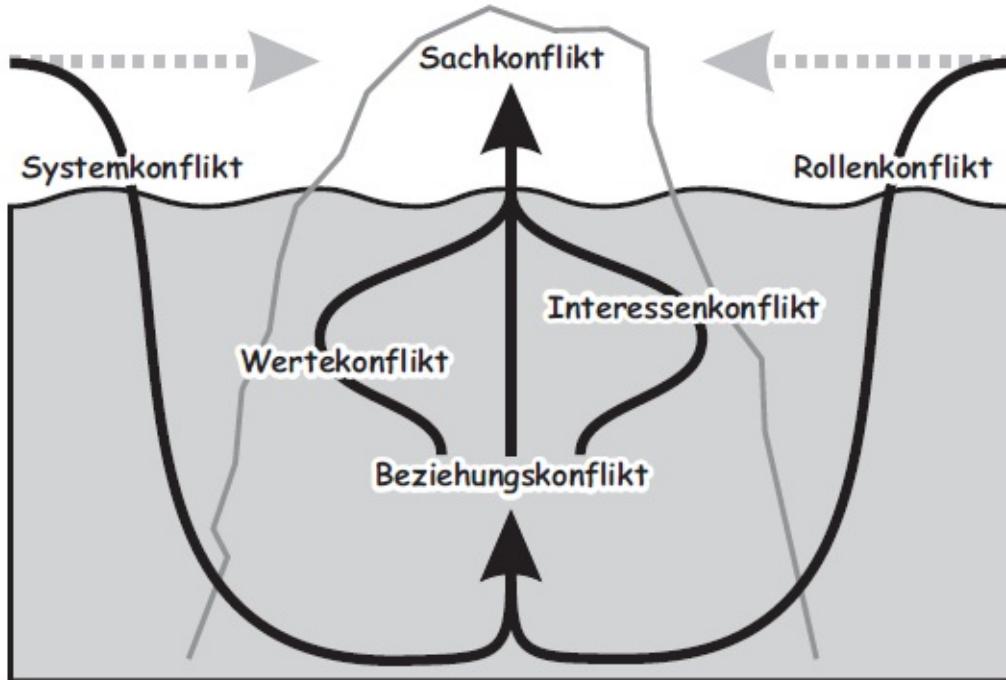


Konflikte sind notwendig, um uns zu hinterfragen und Lösungen zu optimieren. Die Kunst ist es, dafür ein angemessenes Konflikt niveau zu finden

Grundsätzliche Konfliktlösungsstrategie

Ein realer Konflikt besteht meist aus einer Vermengung von verschiedenen Konfliktarten. Jede einzelne Art kann dabei entweder gerade heiß oder erkaltet sein. Hinter einem Sachkonflikt kann beispielsweise auch ein Beziehungskonflikt stehen, der vor längerer Zeit seinen Anfang nahm, jedoch immer noch unterschwellig eine Rolle bei dem derzeitigen Konflikt spielt. Wird nun zuerst der Sachkonflikt gelöst, die gegenseitige Wertschätzung jedoch noch immer nicht vorhanden ist, weil der Beziehungskonflikt nicht gelöst wurde, so wird sehr schnell ein neuer Konflikt auf der Sachebene entstehen, angefacht durch den Beziehungskonflikt. Wird jedoch zuerst der Beziehungskonflikt gelöst, kann auf Basis gegenseitigem Verständnisses und Wertschätzung der Sachkonflikt deutlich besser gelöst werden. Ein sinnvolles Vorgehen bei der Lösung von Konflikten bietet die folgende Abbildung.

Konflikt spielt. Wird nun zuerst der Sachkonflikt gelöst, die gegenseitige Wertschätzung jedoch noch immer nicht vorhanden ist, weil der Beziehungskonflikt nicht gelöst wurde, so wird sehr schnell ein neuer Konflikt auf der Sachebene entstehen, angefacht durch den Beziehungskonflikt. Wird jedoch zuerst der Beziehungskonflikt gelöst, kann auf Basis gegenseitigem Verständnisses und Wertschätzung der Sachkonflikt deutlich besser gelöst werden. Ein sinnvolles Vorgehen bei der Lösung von Konflikten bietet die folgende Abbildung.

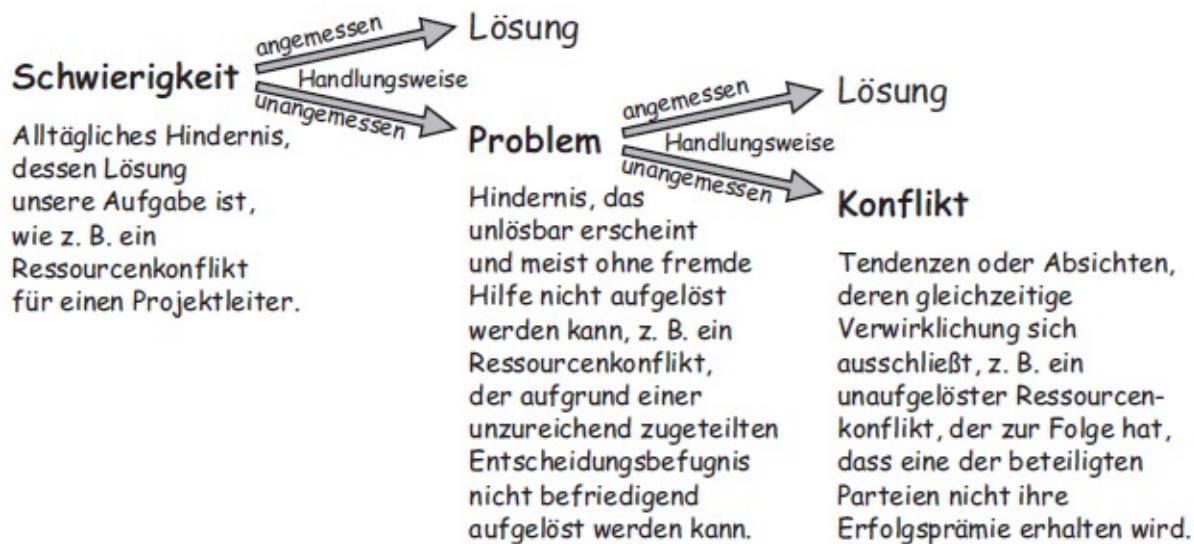


Eine sinnvolle Reihenfolge, in der die einzelnen Konfliktanteile nach den Konfliktarten angegangen werden.

Die verschiedenen Konflikte werden dabei als Eisbergmodell dargestellt. Der Sachkonflikt ist dabei meist nur die Spitze des Eisbergs, darunter liegen aber deutlich mehr, meist nicht sichtbare andere Konflikte. Statt also den Sachkonflikt direkt anzugehen, sollten zuerst die zugrunde liegenden Konflikte gelöst werden.

Konfliktmuster rechtzeitig erkennen

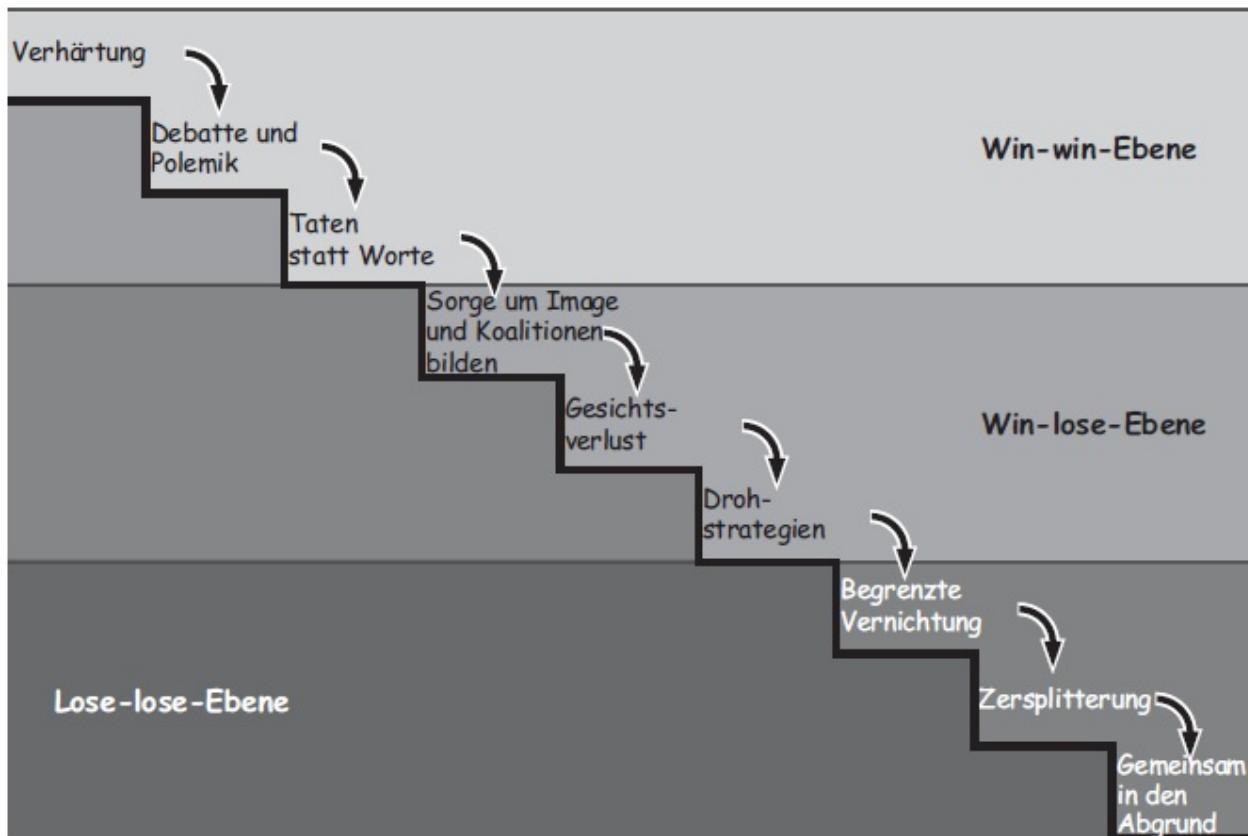
Für ein erfolgreiches Konfliktmanagement ist es notwendig aufkommende Konflikte frühzeitig wahrzunehmen. Nicht jedes auftretende Hindernis in einem Projekt wird sich direkt zu einem Konflikt entwickeln. Zu Beginn sind es meist nur Schwierigkeiten, alltägliche Hindernisse, die meist die betroffene Person selbst lösen kann. Wird angemessen auf die Schwierigkeit reagiert und eine Lösung gefunden kann kein Konflikt mehr darauf entstehen. Eine unangemessene Behandlung der Schwierigkeit führt jedoch zu einem Problem. Ein Problem ist dabei ein Hindernis, das für eine einzelne Person meist unlösbar erscheint und nur mit Hilfe von anderen überwunden werden kann. Wird auf ein Problem wieder



Aus alltäglichen Schwierigkeiten werden durch unangemessene Lösungsversuche Probleme. Werden diese Probleme nicht gelöst, entstehen daraus Konflikte.

Konflikteskalationsstufen

Die Entwicklung eines Konflikts kann auch anhand eines neunstufigen Phasenmodells dargestellt werden. Jeweils drei Phasen sind dabei zu einer Ebene zusammengefasst. In der ersten Ebene, der Win-Win Ebene, kann dabei noch jeder Beteiligte am Konflikt einen Gewinn bei der Lösung des Konflikts gewinnen. In der Win-lose Ebene gibt es jedoch mindestens einen Verlierer, beispielsweise weil jemand anderes seine Meinung durchgesetzt hat, oder es wird ein Kompromiss gefunden. Dann hat zwar keiner einen vollständigen Gewinn gemacht, aber auch nicht komplett verloren. Die Lose-lose Ebene hat zur Eigenschaft, dass alle Beteiligten nur noch Verlieren können. Hier wird entweder der eigene Verlust in Kauf genommen, solange die andere Partei noch größere Verluste hat, oder er wird mit allen Mitteln versucht die andere Partei vom Gewinnen abzuhalten.



Konflikte eskalieren häufig nach dem neunstufigen Phasenmodell der Eskalation von Friedrich Glasl. Jeweils drei Stufen bilden eine Ebene, die die Möglichkeiten der beteiligten Parteien beschreibt.

1. Stufe: Verhärtung

Alltägliche Schwierigkeiten lassen Meinungsverschiedenheiten aufeinanderprallen. Da Meinungsverschiedenheiten recht alltäglich sind, werden sie häufig auch nicht als Beginn eines Konflikts wahrgenommen.

2. Stufe: Debatte und Polemik

Konfliktpartner überlegen sich Strategien, um andere von ihren Argumenten zu überzeugen und unter Druck zu setzen.

3. Stufe: Taten statt Worte

Konfliktpartner erhöhen durch Taten den Druck auf die anderen um ihre jeweilige Meinung durchzusetzen. Gespräche werden abgebrochen, es findet keine direkte Kommunikation mehr statt.

4. Stufe: Sorge ums Image und Koalitionsbildung

Konfliktparteien versuchen Unterstützer für ihre Sache zu finden, Gegner werden denunziert. Konflikt verschärft sich so stark, dass es nicht mehr um die eigentliche Sache geht, sondern nur noch darum zu gewinnen.

5. Stufe: Gesichtsverlust

Mit Gesichtsverlust ist der Verlust der moralischen Glaubwürdigkeit der Gegner gemeint. Sie sollen in ihrer Identität vernichtet werden, unter anderem durch Unterstellungen und politischen Machtspielen. Verlust des kompletten gegenseitigen Vertrauensverhältnisses.

6. Stufe: Drohstrategien

Konfliktparteien versuchen die Situation durch Drohungen zu kontrollieren. Zwar können auch in den vorherigen Phasen schon Drohungen ausgesprochen werden, in dieser Stufe sind sie jedoch sehr ernst gemeint und auch umgesetzt.

7. Stufe: Begrenzte Vernichtung

Konfliktparteien versuchen sich mit allen Tricks gegenseitig zu schaden. Der Gegner wird dabei nicht mehr als Mensch wahrgenommen und solange sein Schaden größer ist, wird auch ein eigener kleiner Schaden als Gewinn angesehen.

8. Stufe: Zersplitterung

Konfliktparteien versuchen sich mit Vernichtungsstrategien gegenseitig zu zerstören.

9. Stufe: Gemeinsam in den Abgrund

Eigene Vernichtung wird einkalkuliert, um den Gegner zu besiegen.

Kommunikationsmuster in Konflikten

Häufig werden in Konflikten destruktive Kommunikationsweisen genutzt. Dabei wird dies nicht bewusst gemacht, sondern man fällt aufgrund von Stress unbewusst in bestimmte Kommunikationsmuster zurück.

Um diese Muster zu verdeutlichen wird die Dimension eines Konflikts auf drei Aspekte reduziert.

Selbst: Die eigene Person, meine Bedürfnisse, Interessen, Anliegen, sowie mein Selbstwertgefühl

Andere: Die anderen Personen im direkten Umfeld und deren Bedürfnisse, Interessen und Anliegen

Kontext: Die konkrete Sache um die es geht und alle direkt damit im Zusammenhang stehenden fachlichen Aspekte.

Es können nun vier Kommunikationsmuster anhand daraus abgeleitet werden, wie eine Person in ihrer Kommunikation diese Aspekte gewichtet. Je nach Muster können dabei manche Sachen gut funktionieren, andere wiederum nicht. Es kann daher für die Konfliktbewältigung von Vorteil sein manche Muster durchzusetzen, oder zu verlassen.

Beschwichtiger: Fokussierung auf Kontext und Andere, Vernachlässigung des Selbst.

- **Motto:** Nur wenn ich für andere Sorge, geht es mir gut
- **Charakteristika:** Macht sich viele Gedanken und ist anspruchslos für sich selbst
- **Typische Syntax:** häufiger Gebrauch von Einschränkungen und Konjunktiven
- **Was funktioniert gut?** Harmonie schaffen, vermitteln im Team, Sympathie erzeugen, anpassungsfähig, hohes Risikobewusstsein, zuverlässig sein und sehr genau hinschauen (allerdings dafür langsamer)
- **Was funktioniert nicht?** Etwas durchsetzen, standfest sein, streiten (gibt schnell nach)

und aktive Führung

Rationalisierer: Fokussierung auf Kontext, Vernachlässigung von Selbst und der Anderen.

- **Motto:** Nur wenn wir ganz sachlich und logisch vorgehen, kommen wir zu einem guten Ergebnis
- **Charakteristika:** Fakten dominieren, zeigt keine Emotionen, detailverliebt
- **Typische Syntax:** Substantivierung, tilgen von Verben, Formulierungen mit "man"
- **Was funktioniert gut?** Forschen, Ziele und Maßnahmen finden, Wissen sammeln, Einzelarbeit, Regeln und Grenzen einhalten, Auswirkungen betrachten und Neutralität im Sinne eines Schiedsrichters
- **Was funktioniert nicht?** Unkonventionelle Wege gehen, Kompromisse erarbeiten, Konflikte und die dahinter stehenden Bedürfnisse erkennen

Ankläger: Fokussierung auf Kontext und Selbst, Vernachlässigung der Anderen.

- **Motto:** Nur wenn ich kämpfe, bekomme ich etwas
- **Charakteristika:** lebhaft, laut, schnell, aggressiv
- **Typische Syntax:** Formulierungen mit "andere" oder Universalquantoren wie "immer" oder "jeder"
- **Was funktioniert gut?** Etwas vorantreiben, effizient arbeiten, etwas optimieren, Klarheit schaffen, Sicherheit geben, schnell Ergebnisse liefern und Führung im Sinne von Orientierung
- **Was funktioniert nicht?** Kompromisse erarbeiten, andere einbinden, zuhören, genau arbeiten, Umwelt wahrnehmen, Fingerspitzengefühl und Zusammenarbeit

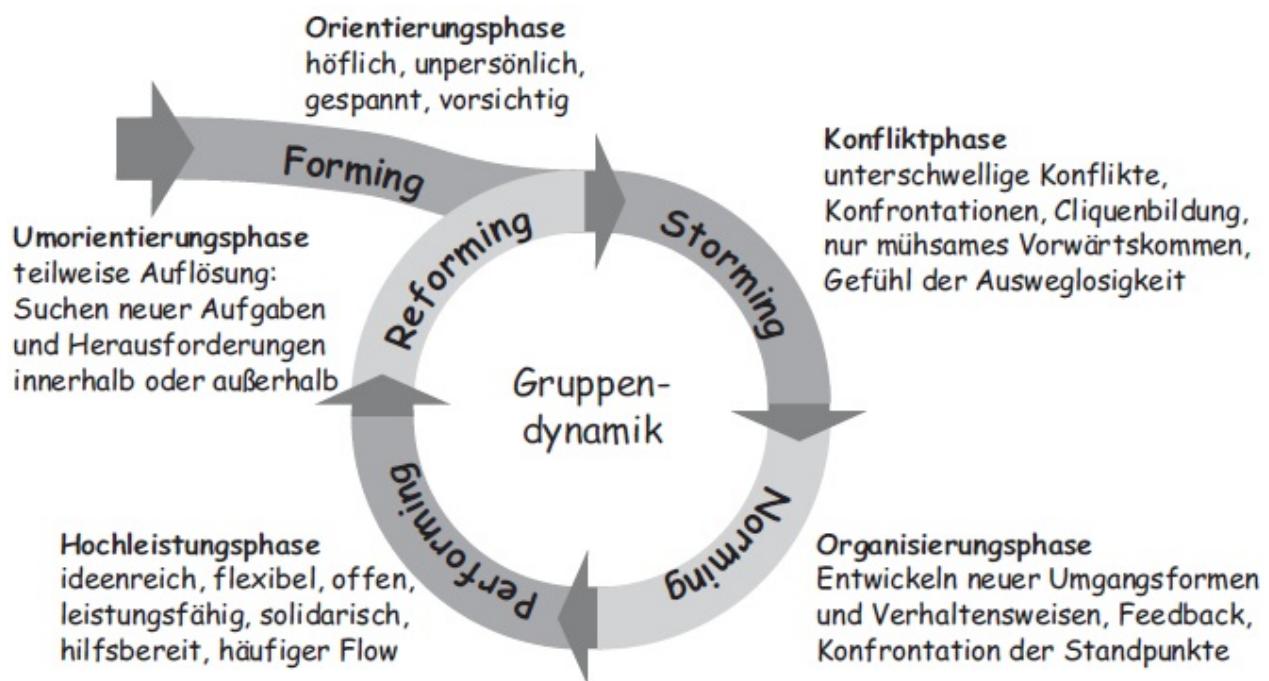
Ablenker: Fokussierung auf nichts, Vernachlässigung des Kontext, des Selbst und der Anderen

- **Motto:** Nichts ist wirklich wichtig
- **Charakteristika:** unstrukturiert bis chaotisch, unverbindlich, kreativ, immer gute Stimmung
- **Typische Syntax:** rascher Wechsel der Wortwahl, eher generelle und zukunftsorientierte Formulierungen, Worte ohne Beziehung zum Gesprächspartner und dadurch kaum konkret greifbar
- **Was funktioniert gut?** Neue Ideen einbringen, Begeisterung, Vision entwickeln und Schwung einbringen, Motivation sowie Innovation
- **Was funktioniert nicht?** Konflikte austragen, Verbindlichkeit schaffen und Dinge zu Ende bringen

Gruppendynamik

Konflikte zwischen Menschen können nur entstehen, wenn Menschen aufeinander treffen, oder in irgendeiner Beziehung zueinander stehen. Es wird also stets eine Gruppe von Menschen betrachtet. Jede Gruppe hat dabei eine eigene Dynamik und es kann innerhalb einer Gruppe zu Konflikten kommen. Des Weiteren besteht auch eine Beziehung zu anderen Gruppen und auch in dieser Beziehung kann es zu Konflikten kommen.

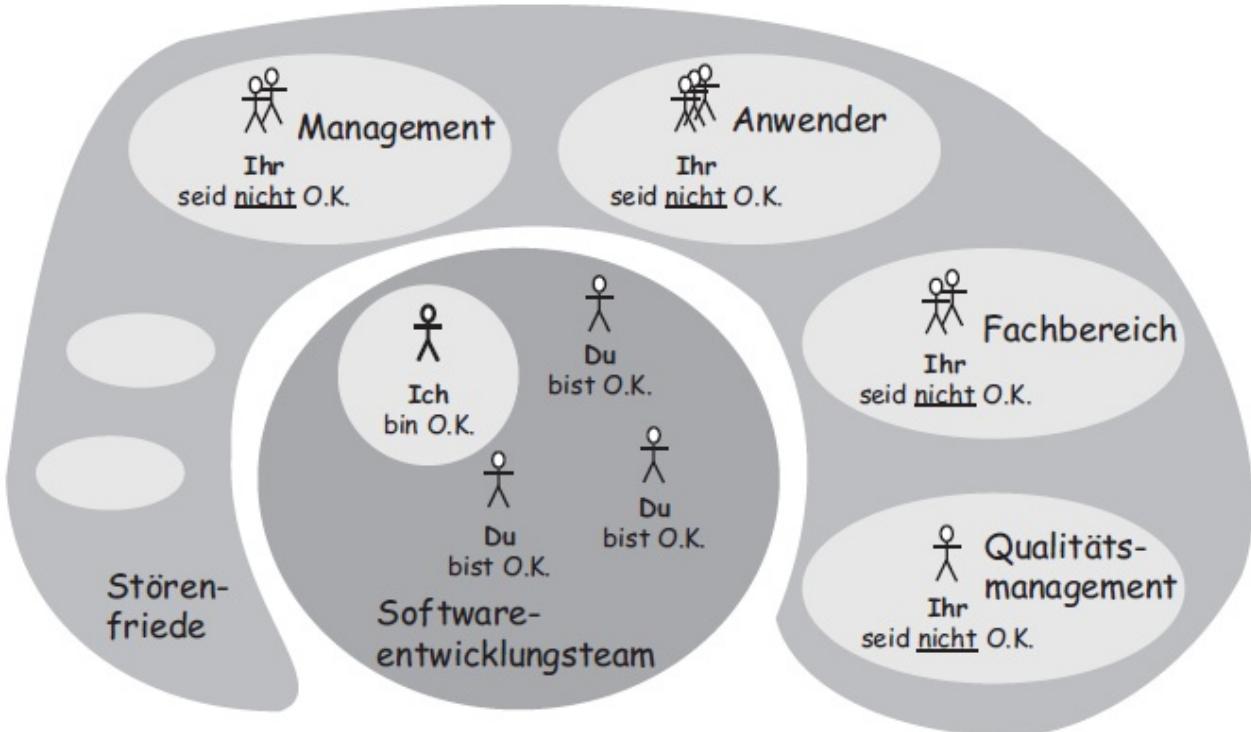
Die Struktur einer Gruppe ist ständigen Veränderungen unterworfen und sie entwickelt sich somit weiter. Diese Entwicklung kann dabei in mehreren Phasen erfolgen, dargestellt in einem Phasenmodell.



Jede Gruppe hat einen Lebenszyklus und obliegt damit einer inneren Dynamik. Damit verbunden ist eine Konfliktphase in der Findung einer Gruppe (Storming)

Zu Beginn des Lebenszyklus einer Gruppe steht dabei die Orientierungsphase, hier herrscht noch Unsicherheit und ein distanziertes Verhalten unter den Gruppenmitgliedern. In der anschließenden Konfliktphase beginnt der Machtkampf der Gruppenmitglieder untereinander. Jeder versucht seine gewünschte Rolle in der Gruppe einzunehmen, es kommt zu Konflikten und die Gruppe selbst kommt nur mühsam voran. Diese Phase ist die Härteprobe einer jeden Gruppe, ist sie überstanden geht es in die Orientierungsphase. Hier entwickeln sich die Umgangsformen und Verhaltensweisen, die Gruppenmitglieder haben ihren Platz gefunden und der Konkurrenzkampf weicht einer intensiven Zusammenarbeit. Die nächste Phase ist dann die Hochleistungsphase, die Gruppe ist eingespielt, kann optimal arbeiten und ihr Potential voll zur Geltung bringen. Nach dieser Phase erfolgt die Phase der Umorientierung. Einzelne Personen verlassen die Gruppe, andere kommen hinzu und es werden neue Aufgaben gesucht. Dabei kommt es wieder zu Spannungen und es tritt erneut die Konfliktphase ein.

Neben Konflikten in der Gruppe selbst, kann es auch zu Konflikten mit anderen Gruppen kommen. Als Entwickler ist man beispielsweise häufig den anderen Mitgliedern des Entwicklungsteams positiv gegenüber eingestellt. Ausstehenden Gruppen, wie beispielsweise der Qualitätssicherung, ist man jedoch negativ gegenüber eingestellt. Sie werden als Störenfriede gesehen, die die eigene Gruppe negativ beeinflussen. Diese Haltung gegenüber anderen Gruppen ist jedoch langfristig schädlich und von Konflikten geprägt.

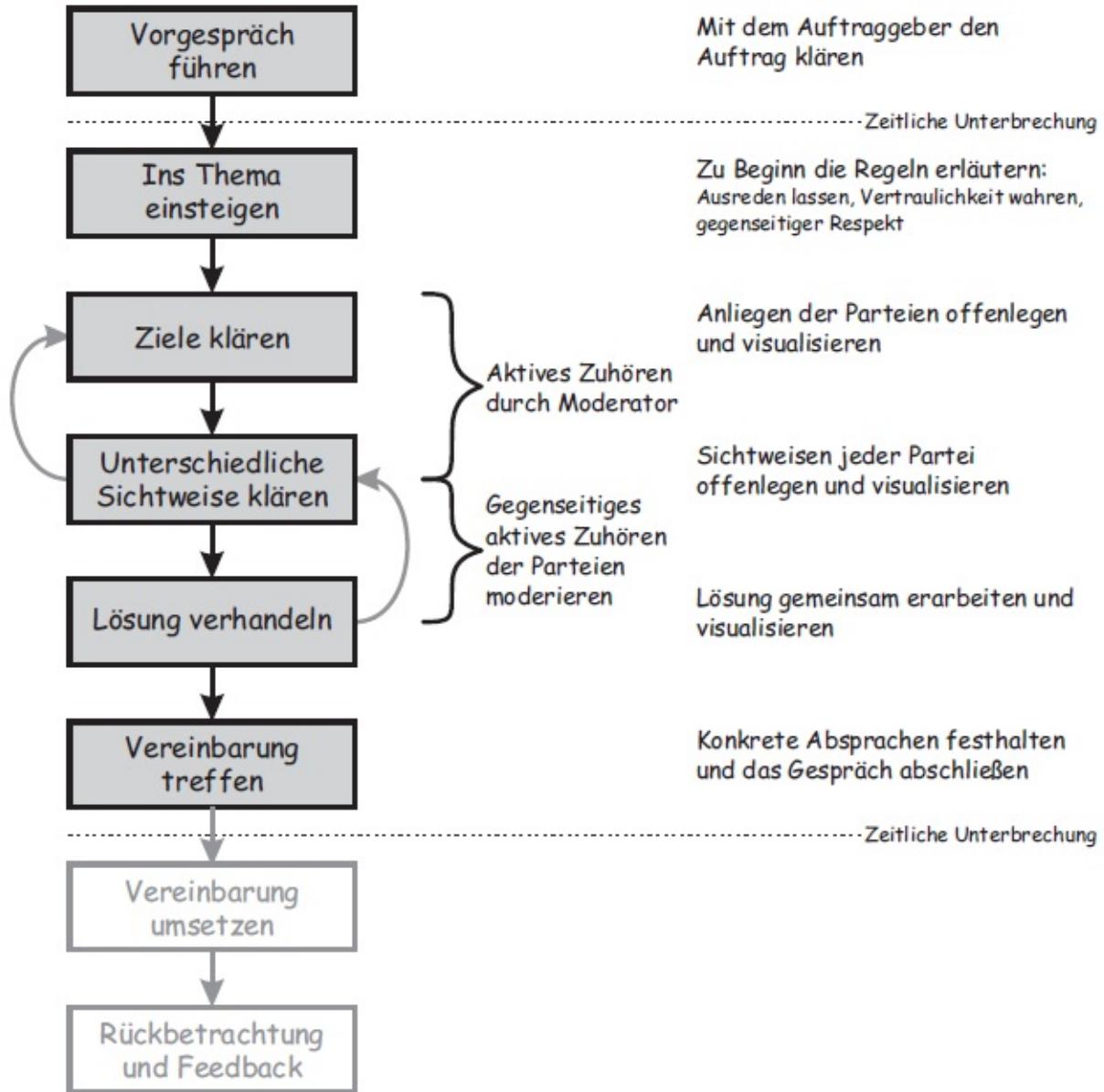


Die Nicht-O.K.-Einstellung zu Gruppen außerhalb der eigenen führt leicht zu Konflikten mit diesen Menschen.

Konflikte managen

Natürlich ist es nicht nur notwendig Konflikte rechtzeitig zu erkennen, sondern auch angemessen mit ihnen umzugehen. Häufig hilft dabei ein Kritikgespräch. Bei einem Kritikgespräch setzen sich zwei Personen zusammen und stellen ihre jeweils unterschiedlichen Sichtweisen dar. Dabei wird stark auf aktives Zuhören gesetzt, sobald eine Person ihre Sichtweise dargestellt hat, gibt die andere wieder, was sie verstanden haben. Dadurch wird ein gemeinsames Verständnis aufgebaut, auf dessen Grundlage nun konkrete Handlungen vereinbart werden können. Wie bei vielen anderen Konfliktaspekten, ist auch hier die gegenseitige Wertschätzung vonnöten: Kritik soll nicht dazu dienen die Person zu verletzen, sondern ihr zu helfen. Eine andere Art Konflikte zu managen besteht in der Konfliktmoderation. Dabei gibt es einen allparteilichen Moderator, der keiner Seite einen Vorzug gibt und allen Lösungsmöglichkeiten offen gegenübersteht. Dieser leitet die

Diskussion der Konfliktparteien und legt dabei den Fokus auf den Inhalt der Auseinandersetzung und dem Empfinden und Sichtweisen der Beteiligten. Er achtet darauf, dass ein aktives Zuhören stattfindet sowie darauf, dass sachliche Aussagen und positive Ergebnisse visualisiert werden. Der Ablauf einer Moderation kann dabei in acht Phasen erfolgen, dargestellt in der folgenden Abbildung.



Struktur einer Konfliktbearbeitung mit Moderation.

Hilft auch die Konfliktmoderation nicht mehr, so gibt es die Möglichkeit der Konfliktmediation. Dabei gibt es einen neutralen außer parteilichen Moderator, der den beteiligten Parteien dabei hilft, selbstständig eine Lösung zu finden. Die Konfliktparteien müssen dabei freiwillig an der Mediation teilnehmen, bereit sein fair und offen zu kommunizieren und für alle Lösungsmöglichkeiten offen sein. Der Ablauf einer Mediation ist dabei ähnlich wie bei der Konfliktmoderation, sie ist jedoch etwas förmlicher. Außerdem besitzt sie eine gesetzliche Grundlage, Mediatoren dürfen eine Mediation nur durchführen, wenn sie die gesetzlichen Voraussetzungen erfüllen.

Erfolgreich Verhandlungen führen

Die Fähigkeit, erfolgreich Verhandlungen zu führen, ist für Softwareentwickler von Vorteil, beispielsweise um wichtige Entscheidungen bei der Projektgestaltung mitzubestimmen.

Meist trifft man bei Verhandlungen auf zwei Verhandlungsstrategien, zum einen dem harten Verhandeln, bei dem der Sieg und somit der eigene Vorteil im Mittelpunkt steht, zum anderen das weiche Verhandeln, wo versucht wird eine möglichst konfliktfreie Übereinkunft zu finden.

	Hartes Verhandeln	Weiches Verhandeln
Ziel:	Sieg	Übereinkunft
Haltung:	Hart zu Menschen und Positionen, Verhandlungspartner sind Gegner, Misstrauen	Weich zu Menschen und Positionen, Verhandlungspartner sind Freunde, Vertrauen
Strategie:	Drohen, beharren auf Positionen, Zugeständnisse der anderen Seite sind der Preis für die Übereinkunft, Druck ausüben	Angebote machen, ändern von Positionen, einseitige Zugeständnisse, bestehen auf Übereinkunft, Druck nachgeben
Linie:	Verdeckt	Offen

Hartes und weiches Verhandeln in der Gegenüberstellung bzgl. Ziel, innerer Haltung, Strategie und Verhandlungslinie.

Beide Varianten haben jedoch Nachteile, harte Verhandlungen führen langfristig zu schlechten Geschäftsbeziehungen, weiche Verhandlungen führen dazu, dass die eigenen Ziele und Wünsche nicht beachtet werden.

Um erfolgreiche Verhandlungen zu führen, ist es daher hilfreich sich am Harvard Konzept zu orientieren. Hier wird eine dritte Möglichkeit des Verhandelns benutzt, das Sachorientierte Verhandeln.

	Sachorientiertes Verhandeln
Ziel:	Dauerhafte, vernünftige Übereinkunft
Haltung:	Teilnehmer sind Problemlöser bzw. Partner, offen und weich zu den Menschen, hart bei den eigenen Interessen
Strategie:	Menschen und Probleme getrennt behandeln, beiderseitige Interessen herausarbeiten, Optionen finden, Bewertungskriterien entwickeln
Linie:	Vermeiden, um flexibel einen Interessenausgleich erreichen zu können

Kennzeichen sachorientierten Verhandelns bzgl. Ziel, innerer Haltung, Strategie und Verhandlungslinie.

Hier wird beim Verhandeln zwischen den Sach- und Beziehungsebene unterschieden. Geht es um die Sache werden harte Argumente benutzt, gleichzeitig drücken wir jedoch in der Beziehungsebene unsere gegenseitige Wertschätzung aus. So können Lösungen gefunden werden, die von allen beteiligten Parteien vernünftig akzeptiert werden.

Soft Skills für IT-Berater

Zusammenfassung des Buches:

Titel: Soft Skills für IT-Berater

Verfasser: Uwe Vigenschow, Björn Schneider

Verlag: dpunkt.verlag

Jahr: 2012

ISBN: 978-3-89864-780-9, 978-3-86491-203-0 (eBook)

Zusammenfassung von: Tolga Aydemir, Justin Jagieniak, Niklas Harting, Malte Berg, Oliver Nagel, Jonathan Jansen, Sven Schirmer

1 Beratung in der IT

Die Beratung in der IT kann durch externe und interne Berater in einem Unternehmen erfolgen. Dabei kann die Beratung auch abteilungsübergreifend stattfinden. Da der Begriff des "Beraters" sehr häufig genutzt wird, kann es weiter differenziert werden in methodische und fachlich-fokussierte Berater. Der methodische Berater bindet sich nicht langfristig an das Kundenunternehmen und verlässt es nach der Herbeiführung einer bestimmten (vom Kunden gewünschten) Veränderung. Eine weitere Differenzierung der Beratungstätigkeiten kann durch die folgenden Begriffe erfolgen, die im Anschluss genauer definiert werden: Technologieberatung, Strategieieberatung, Organisationsberatung.

- Technologieberatung:

Die Technologieberatung ist eine fachlich-fokussierte Beratungstätigkeit, wobei es auch eine rein methodische Beratung sein kann. Dazu darf der Berater allerdings keine langfristige Entwicklungsarbeit übernehmen und muss das Unternehmen nach dem vom Kunden gewünschten Wissentransfer verlassen. Die Aufgaben eines Technologie-Beraters sind die Auswahl, Einführung und Entwicklung mit einer neuen Technologie. Dies kann die Auswahl eines für den Kunden neuen Entwicklungsumfelds oder Softwareprodukts sein und betrifft in den meisten Fällen auch die Implementierung der Kundensoftware bzw. das verändern von Produkten. Häufig kommt dabei ein ganzes Team von Beratern zum Einsatz mit einer eigenen hierarchischen Struktur. Die meisten dieser Berater arbeiten dann vor Ort beim Kunden als Softwareentwickler oder in einer ähnlichen Rolle.

- Strategieieberatung:

Die Strategieieberatung ist eine methodische Beratungstätigkeit und umfasst die Unterstützung des Kunden bei strategischen Entscheidungen mit langfristigen Folgen. Dies kann auch die Einführung einer neuen Technologie oder eines Softwareprodukts betreffen. Der Fokus liegt jedoch auf der fachlichen Beratung, die auf der Erfahrung des Beraters mit der gewünschten Lösung und ihrer Einführung beruht. Vom Kunden wird dabei bewusst ein Veränderungsprozess angestoßen. Der Berater unterstützt bei der Entscheidungsfindung, Konzeption und Umsetzung des Veränderungsprojekts.

- Organisationsberatung:

Hier steht die Veränderung einer Organisation im Vordergrund, um die Arbeitsprozesse zu optimieren. Auslöser sind häufig veränderte Marktbedingungen oder erweiterte bzw. verschobene Aufgabenfelder einer Organisation. Technologische Betrachtungen spielen dabei meist nur als Rahmenbedingung eine Rolle.

Die methodische Beratung ist auch für kleine Beratungsunternehmen und Freelancer geeignet, die keine langfristige Bindung beim Kunden haben wollen oder können. Die Verantwortung bei der methodischen Beratung liegt beim Kunden und nicht beim Berater.

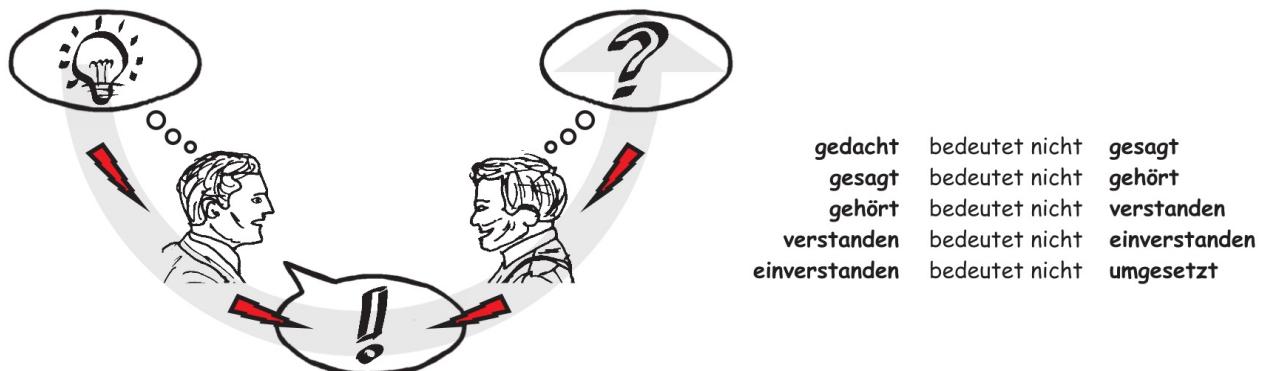
2 Kommunizieren und verstehen

Um den Prozess der Beratung effizient zu gestalten, gibt es Methoden und Hilfestellungen im Bereich der Kommunikation, die in diesem Kapitel vorgestellt werden. Dabei gibt es eine grobe Aufteilung in: Fragen stellen und miteinander reden; Auf Einwände angemessen reagieren; Inhalt, Form und Beziehung

Fragen stellen und miteinander reden

Die Beratung eines Unternehmens hat auch zur Folge, dass der Berater auf die Wünsche und Probleme der einzelnen Mitarbeiter eingehen muss. Dies gilt umso mehr in einer methodischen Beratung. Die Einarbeitung in diesen Abteilungen mit den jeweiligen Mitarbeitern läuft im Idealfall über ein gemeinsames Gespräch ab.

Dabei werden jedoch Probleme auftreten:



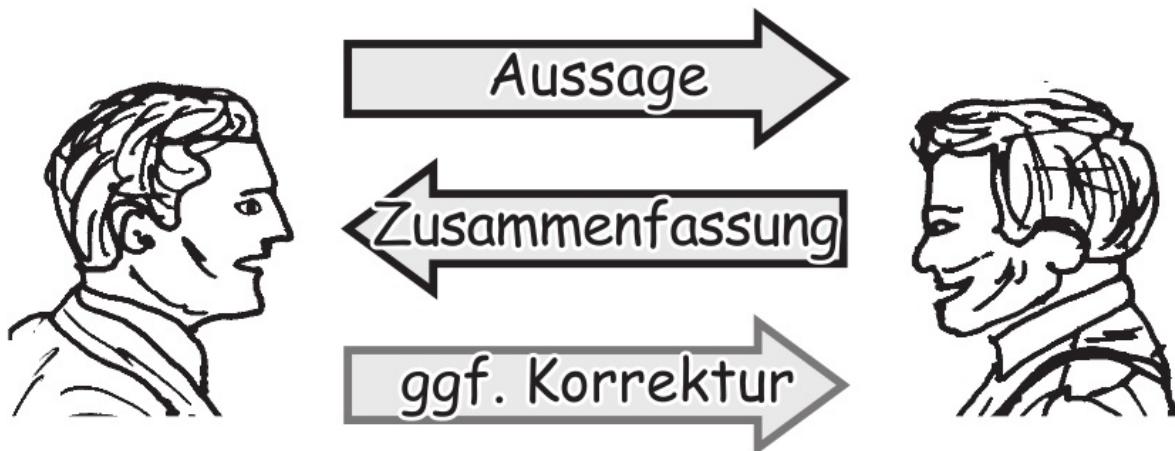
(Abbildung: Mögliche Probleme in der Kommunikation)

Wenn das Gespräch in einer Fremdsprache für einen der beteiligten Personen abläuft, dann nehmen diese Kommunikationsprobleme zu.

Der Berater muss ein gemeinsames Bild in den Gedanken der beteiligten Personen erschaffen, damit die Kommunikationsprobleme minimiert werden.

Aktives Zuhören

Ein Werkzeug, um diese Kommunikationsprobleme zu minimieren ist das aktive Zuhören. Person A erzählt etwas. Person B fasst das erzählte zusammen und gibt dadurch Person A die Möglichkeit das Bild in den Gedanken von Person B zu prüfen und ggf. zu korrigieren. Außerdem wird dadurch das Gesprächstempo verlangsamt, sodass auch Fremdsprachler die Details des Gesprächs mitnehmen können.



(Abbildung: Aktives Zuhören)

Mit Fragen ein Gespräch führen

Während der Beratungsgespräche ist die Aufgabe des Beraters das Gespräch zu führen, um so auf Problemstellungen aufmerksam zu machen. Die sog. offenen Fragen (auch W-Fragen genannt) können dabei helfen. Die geschlossenen Fragen (mit ja/nein zu beantworten) sind eher dazu geeignet, um Zustimmung oder Ablehnung innerhalb der Gesprächsgruppe abzufragen.

Wichtig werden die Fragen, wenn die Mitglieder der Gesprächsgruppe sich gegenseitig blockieren oder Probleme auftreten. Dann ist es die Aufgabe des Beraters mit den richtigen Fragen aus dieser Blockade zu manövrieren.

Sechs-Stufen-Fragetechnik

Die Sechs-Stufen-Fragetechnik kann dabei helfen Informationen zu erlangen, die noch gar nicht erwähnt wurden, jedoch wichtig für die Analyse des Problems sind. Dadurch entsteht ein Gesprächsleitfaden und unterstützt den Berater dabei auch an Informationen zu gelangen, die für den Gesprächspartner (interne Mitarbeiter) so selbstverständlich sind, dass diese gar nicht erwähnt werden.

- Prozesswörter überprüfen:

Identifizieren Sie Verben und substantivierte Verben wie z. B. melden, Auslastung, erfassen. Stellen Sie zu jedem dieser Prozesswörter die W-Fragen: Wer? Was? Wann? Wie? Wo? Warum?

- Komparative und Superlative überprüfen:

Bestimmen Sie die Bezugspunkte: Worauf bezieht sich der Vergleich oder die Steigerung? Bringen Sie die Messmethode in Erfahrung, über die die geforderten Eigenschaften nachgeprüft werden können.

- Universalquantoren überprüfen:
Identifizieren Sie die Universalquantoren wie alle, keiner, immer, nie, jeder, stets usw. Hinterfragen Sie darüber die Ausnahmen sowie die impliziten Annahmen.
- Bedingungen überprüfen:
Gibt es noch andere Varianten? Sind alle Möglichkeiten vollständig aufgezählt? Sind alle Entscheidungs- und Verzweigungsbedingungen genannt?
- Konstanten und konfigurierbare Werte überprüfen:
Identifizieren Sie feste sowie konfigurierbare Größen und Werte und geben Sie ihnen sprechende Namen wie z. B. Rechnungsschwellwert oder Volljährigkeit. Tragen Sie die Namen mit ihren aktuellen Werten in ein tabellarisches Glossar an zentraler Stelle z. B. im Intranet ein.
- Abkürzungen und Fachbegriffe im Glossar definieren:
Identifizieren Sie alle Abkürzungen, Akronyme und Fachbegriffe und definieren Sie diese in einem Glossar. Prüfen Sie dabei auf Widersprüche oder unterschiedliche Sichten in den beteiligten Fachbereichen. Der Begriff Kunde könnte z. B. im Marketing oder Produktmanagement unterschiedliche Aspekte haben.

Auf Einwände angemessen reagieren

Wenn Einwände seitens des Gesprächspartner geäußert werden, dann muss der Berater die jeweilige Person zunächst analysieren, um angemessen zu reagieren. Eine falsche Reaktion, kann zu nachhaltigen Kommunikationsproblemen führen und die Tätigkeit des Berates negativ beeinflussen. Um die Gesprächspartner analysieren zu können, gibt es eine grobe Einteilung der Persönlichkeiten in vier Quadranten und wie diese zu erkennen sind:



(Abbildung: Das Vier-Quadranten-Modell)

Zum Beispiel können Aussagen wie "Wozu das alles?" dem Warum-Quadranten zugeordnet werden. Diese Person hat eine skeptische Haltung und tendiert zu philosophischen Fragen. Das kann allerdings ein Zeichen für den Rückzug dieser Person aus dem Gespräch sein. Die Erkennungsmerkmale und typischen Aussagen für diese Quadranten sind in den folgenden Abbildungen dargestellt:

Warum	Was	Wie	Wohin noch
Skeptische Haltung	häufiges Rückfragen	kompromissbereit	visuelle, schnelle Sprache
Philosophische Fragen	prüfende Fragen	schnelle Entscheidung	in die Zukunft gerichtete Fragen
Provokation, Unterstellung	keine emotionale Übertreibung	situativ aufmerksam	unverbindlich, unabhängig von anderen
Rückzug oder Aggression	auditive Neigung	schnell, kurz	ignorant

(Abbildung: Erkennungsmerkmale der vier Grundtypen)

Warum	Was	Wie	Wohin noch
Wozu das alles?	Das ist ein Problem!	Das ist doch ganz einfach!	Dann könnte man ja auch ...
Das ergibt für mich keinen Sinn!	Das ist eine gute Frage!	Wozu lange reden?	Ich sehe da für uns noch ...
Das ist doch alter Wein in neuen Schläuchen!	Können Sie mal sagen ... Erklären Sie mir genauer ...	Ich mache das mal eben!	Das schränkt uns zu stark ein!

(Abbildung: Typische Aussagen oder Fragen der vier Grundtypen)

Nach der Einordnung eines Einwands in eines der vier Quadranten, kann nun angemessen darauf reagiert werden. Auch dazu gibt es Empfehlungen für den Berater:

- Warum
 - Einwand wertschätzen
 - Persönlichen Nutzen, Sinn bzw. Zweck herausstellen
 - Das dem Einwand zugrunde liegende Problem herausarbeiten und dessen Lösung als zentralen Erfolgsfaktor für eine angestrebte Veränderung erkennen und würdigen
- Was
 - Informationen und Details geben
 - Weiterführende Quellen und Literaturhinweise nennen
 - Konkrete Fakten und Detailtiefe bieten

- Etwas demonstrieren
- Wie
 - Konkrete Handlungsanweisungen
 - Muster und Beispiele
 - Ergebnistypen und Checklisten
 - Selbst ausprobieren lassen
- Wohin noch
 - Möglichkeiten, Optionen und Zusammenhänge herausstellen
 - Lösungen anbieten
 - Stufenpläne oder schrittweise Lösungskonzepte erarbeiten, um die Freiheitsgrade für zukünftige Entscheidungen lange offen halten zu können

Die Einteilung einer Präsentation diesen vier Quadranten entsprechend, kann den Zuhörern dabei helfen, die Präsentation besser zu verstehen. Dazu gibt es mit dem runden Pfeil in der Abbildung "Das Vier-Quadranten-Modell" eine Empfehlung für die Reihenfolge der in einer Präsentation zu beantwortenden Fragen.

3 Workshops gezielt einsetzen

Ein Workshop ist ideal, um in kleinen fokussierten Gruppen Ideen zu entwickeln, zu präsentieren oder zu implementieren.

Klein und fokussiert, damit sich die Gruppenmitglieder nicht gegenseitig blockieren.

Allerdings müssen Workshops geübt werden, damit diese effizient abgehalten werden können (auch in Notfällen).

Der Workshop-Leiter (Berater) hat die Aufgaben der Vorbereitung, Durchführung und Nachbereitung des Workshops.

Die Vorbereitung umfasst zum Beispiel die genaue Benennung des Themas, um überhaupt möglichst geeignete Workshop-Teilnehmer zu finden. Die Ziele und Aufgaben des Workshops müssen klar definiert sein, sodass auch der Workshop-Leiter selbst die Möglichkeit hat, die Diskussionen zielorientiert zu lenken.

Während der Durchführung kann auch eine gezielte Gruppierung der Teilnehmer helfen das Thema zu bearbeiten. Zum Beispiel können abteilungsübergreifende Gruppen förderlich für das Ziel sein oder hinderlich. Diese Entscheidungen muss der Workshop-Leiter treffen.

Der Workshop-Leiter muss nicht unbedingt mehr Fachwissen als die Teilnehmer haben.

Die Nachbereitung des Workshops kann die Ergebnisse besser verständlich machen und Erkenntnisse für den nächsten Workshop können gewonnen werden.

4 Workshops leiten Teil 1

Workshop-Leiter müssen nicht zwingend eine gewisse Position im Unternehmen oder in der Abteilung haben. Sie müssen auch nicht mehr Fachwissen als die Teilnehmer haben, denn der Moderator ist für den Ablauf des Workshops zuständig und nicht für den Inhalt. Die Teilnehmer müssen den Inhalt gemeinsam erarbeiten. Allerdings ist ein Berater mit den entsprechenden Soft Skills und den Werkzeugen, um Kommunikationsprobleme zu lösen, dafür prädestiniert ein Workshop zu leiten. Denn der Workshop-Leiter hat die Aufgabe ein Workshop zu moderieren und damit die diversen Diskussionen in die richtige Richtung zu steuern und hitzige Diskussionen zu mäßigen und auf eine sachliche Ebene zurückzuführen.

Ein Werkzeug des Moderators kann dabei das Flipchart sein. Der Moderator sorgt dafür, dass alle wichtigen Erkenntnisse und Probleme transparent für alle Beteiligten notiert wird. Aber auch sowas wie Blickkontakt und Auftreten sind für den Moderator wichtig - ähnlich wie bei einer einfachen Präsentation vor einer Gruppe.

Das aktive Zuhören (siehe Kapitel: Kommunizieren und verstehen) ist sehr wichtig als Moderator. Der Moderator muss auf das Befinden aller Teilnehmer reagieren können und erkennen wann die Diskussion zu schnell wird. Die kurzen Zusammenfassungen und deren Korrektur helfen allen Teilnehmern ein gemeinsames Bild in den Gedanken zu erzeugen.

4.2 Visualisierung - die optische Rhetorik

Damit das Gehirn besser Inhalte verarbeiten kann ist es wichtig durch Visualisierung Inhalte zu transportieren, da sie durchs Auge wesentlich stärker als über Worte aufgenommen werden. Durch UML (Unified Modeling Language) lassen sich Modelle zu Abläufen und Strukturen modellieren.

4.2.1 Visuelles Kommunizieren

Um Ideen und Konzepte bei einem Fotoprotokoll zu finden, kann man Whiteboards, Flipcharts und Metaplanwände einsetzen. Dabei sind folgende Regeln zu befolgen:

- Stifte sollt man mit schrägen Kanten als Spitze benutzen. Schreiben mit der ganzen Kante, wobei die Spitze zum Daumen zeigt. Dies erhöht die Lesbarkeit.
- Blockschrift mit vergleichsweise großen Kleinbuchstaben verwenden.
- Eng schreiben
- Mit Groß- und Kleinschreibung schreiben
- Beim Einsatz von Metaplankarten maximal 3 Zeilen Text pro Karte.

4.2.2 Gestaltung

Informationen zu Flipcharts werden wie folgt strukturiert:

- Einstieg: Veranstaltung eröffnen, Kontakt herstellen und Organisatorisches klären.
- Themenorientierung: Situation und Ziel der Veranstaltung klären, Aufgaben und Rahmen abstecken.
- Themenbearbeitung: Inhalte besprechen, Lösungen entwickeln und deren Akzeptanz sicherstellen.
- Handlungsorientierung: Vereinbarungen treffen, Aktivitäten planen und offene Punkte notieren.
- Abschluss: Ergebnisse zusammenfassen, Feedback zu Ablauf einholen und Veranstaltung beenden.

Um im späteren Fotoprotokoll ein Flipchart schnell zu finden, könnte man auf folgende Regeln und Ideen setzen:

- **Stiftfarben:** Schwarzer oder Dunkelblauer Text und Überschriften. Anmerkungen oder Uhrzeiten von Zeitplanungen in Grün. Rot nur bei Auszeichnungen, wie das Unterstreichen einer Überschrift oder das Herausstellen eines wichtigen Aspekts.
- **Überschrift:** Linksbündige Überschriften bei Charts mit einer farbigen Unterstreichung.
- **Datum:** Datum rechts oben platzieren. Dient der Strukturierung bei mehrtägigen

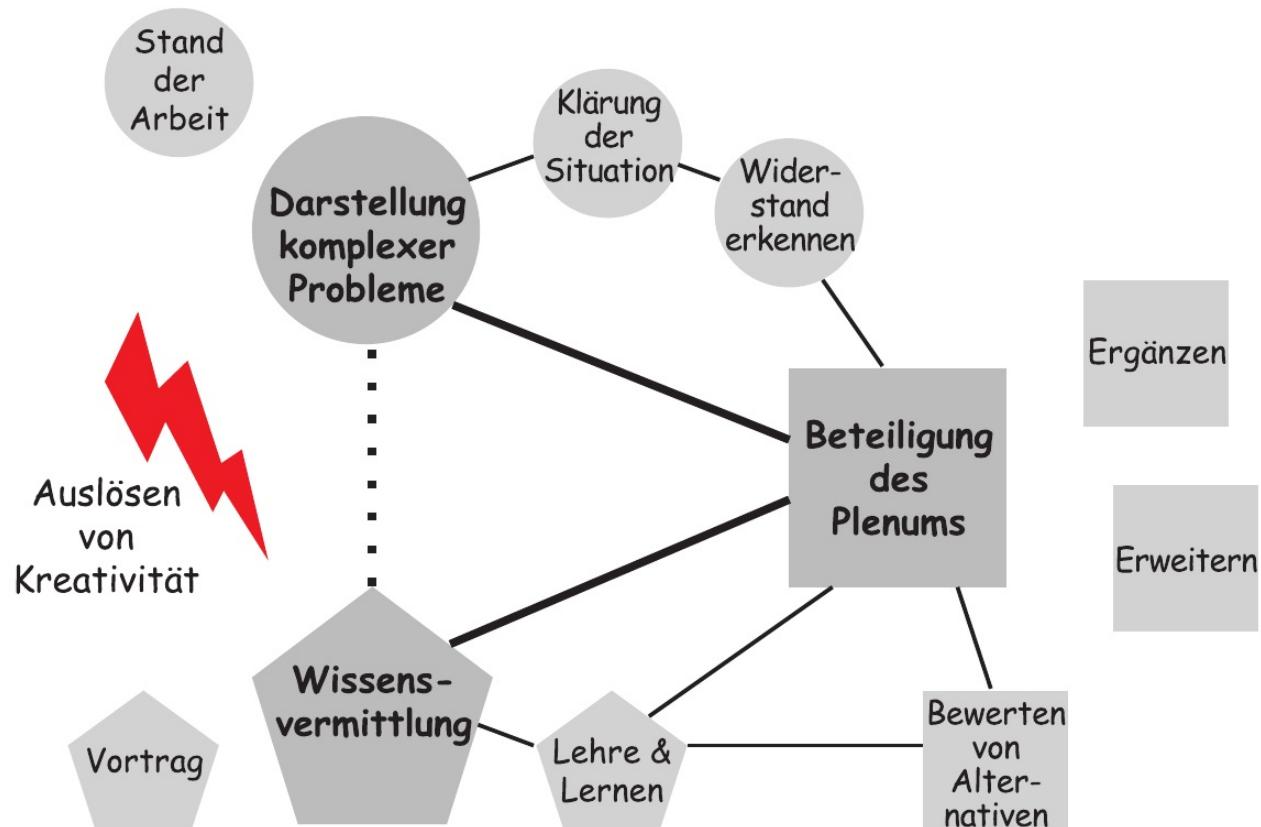
Workshops.

- **Nummerierung:** Für die spätere Rekonstruktion der Entstehungsreihenfolge.
- **Spiegelstriche:** Neue Punkte durch ein Spielgestrich eventuell in Grün oder Rot auszeichnen. Erhöht die Übersichtlichkeit.
- **Protokollkennzeichen:** Beim Abfotografieren eines Flipcharts sollt es sofort zum Beispiel durch einen roten Punkt unten rechts gekennzeichnet werden. Bei Einsatz von mehreren Fotografen oder Kamerassen kann die Kennzeichnung ein Kürzel sein. Beim nachträglichen Verändern kann das Protokollkennzeichen durchgestrichen werden.

Befolgt man alle genannten Regeln hat man eine Basis für erfolgreiches Arbeiten in einem Workshop.

4.2.3 Visuelle Rhetorik - mit Karten strukturieren

Durch visuelle Rhetorik lassen sich Inhalte und Aussagen mit Gefühlen verbinden und dies ist hilfreich bei der Moderation einer Gruppe oder einer Gruppenarbeit.



Man setzt dabei auf folgende vier Grundelemente beim Arbeiten auf einer Metaplanwand, einem Flipchart oder Whiteboard: die Liste, die Tabelle, der Baum und das Netz.

Eine Struktur lässt sich durch Figuren erschaffen, die aus den Karten gebildet werden. Um eine Betonung zu erzielen oder zu vermeiden lassen sich gleichfarbige Karten verwenden. Einzelne Aspekte lassen sich durch das Hervorheben mit einem Stift oder dem

Neuschreiben auf eine andersfarbige Karte betonen. Eine Aneinanderreihung von Karten kann gruppiert werden. Dadurch entsteht ein Rhythmus. Auch Dynamik oder ein Istzustand lässt sich visualisieren. Damit kann man offene Punkte oder zeitliche Aspekte aufzeigen.

5 Grundtechniken für Workshops

5.1 Priorisierungstechniken in Workshop einsetzen

Zunächst einmal ist eine aussagekräftige Priorisierung für den inhaltlichen Umfang eines Workshops notwendig. Das heißt man muss Rahmenbedingungen vorgeben, welche Themen zu behandeln sind, und klären, wie man alle Teilnehmer in die Priorisierung mit einbindet. Somit beschäftigt sich das Kapitel mit der Priorisierung in einer Gruppenarbeit.

5.1.1 Ziele der Priorisierung über eine Leitfrage festlegen

Eine Priorisierung bezieht sich immer auf einen Rahmen und ein Ziel. Es muss geklärt werden, was für die Priorisierung als sinnvoll erscheint. Ein Rahmen kann abstrakt oder allgemein durch eine Aufgabe oder ein wirtschaftliches Ziel vorgegeben sein. Mit einer Leitfrage, die sich aus den konkreten Zielen des Workshops ergibt, kann der Rahmen präzisiert werden. Man kann dies auch als Paraphrasierung umformulieren. Also eine offene Frage, die eine Denkrichtung vorgibt, aber die Lösungssuche nicht einschränkt. Es ist aber auch ein Bewertungskriterium, das zielführende Ideen von ablenkenden unterscheidet. Eine Leitfrage bezieht sich dabei auf den Zeitraum und die Wirkung, die innerhalb des Zeitraums messbar erfolgen soll. Im Folgenden kommen die Priorisierungstechniken.

5.1.2 Muddiest Point

Jeder Workshopteilnehmer kann den schlammigsten Aspekt (muddiest Point) oder das größte Problem auf eine Karte schreiben. Durch das Einsammeln der Karten entsteht eine Übersicht über die schwierigsten Aspekte der Aufgabenstellung. Diese besitzt eine hohe Aussagekraft und man kann in kurzer Zeit wirkungsvolle Ergebnisse erzielen. Der Nachteil liegt dabei an der mangelnden Breite der Ergebnisse und einer subjektiven Priorisierung. Es werden häufig Aspekte erwähnt, die als schwierig erscheinen, aber den Gesamtprozess kaum belasten. Für Softwareentwickler ist es dennoch sinnvoll, da es Informationen auf mögliche Barrieren enthält, die die Einführung von neuen Abläufen oder neuer Software zu einem späteren Zeitpunkt verhindern oder die Akzeptanz verringern.

5.1.3 Punkte kleben

Bei einer längeren Liste an Themen hilft das Punktekleben. Jeder Teilnehmer kriegt eine bestimmte Anzahl an Klebepunkten und darf diese frei an der Liste auf einem Flipchart zuordnen. Mit der Summe der Klebepunkte wird die Priorität gebildet. Mit Hilfe folgender Formel kann man grob die Anzahl der Klebepunkte bestimmen:

$$\frac{\text{AnzahlThemen}}{\text{AnzahlPersonen}} + 1 = kp_{proPerson}$$

Bei mehreren Aspekten

sollte man das Ergebnis lieber abrunden. Beim Provozieren eines stark polarisierten Ergebnisses kann es in Extremfällen sinnvoll sein nur einen Klebepunkt pro Person zu verteilen. Dabei ist es erlaubt die Klebepunkte beliebig zu verteilen. Man darf auch mehrere Klebepunkte einem Aspekt zuordnen. Nur das Teilen eines Klebepunktes bleibt dabei verboten. Taktisch entscheiden sich manche Teilnehmer erst am Ende ihre Punkte zu kleben. Um diesen Trend zu verhindern, kann die Anzahl der Punkte reduziert werden oder mit Hilfe eines Extrablattes die Wahl aller geheim gehalten werden, so dass ein Moderator am Ende die Extrazettel einsammelt und die Punkte auf den Flipchart klebt. Wenn Themen ausgeschlossen werden, werden sie als solche gekennzeichnet. Diese Vorabplanung hilft, wenn für jene Themen zusätzliche Personen anwesend sein sollen.

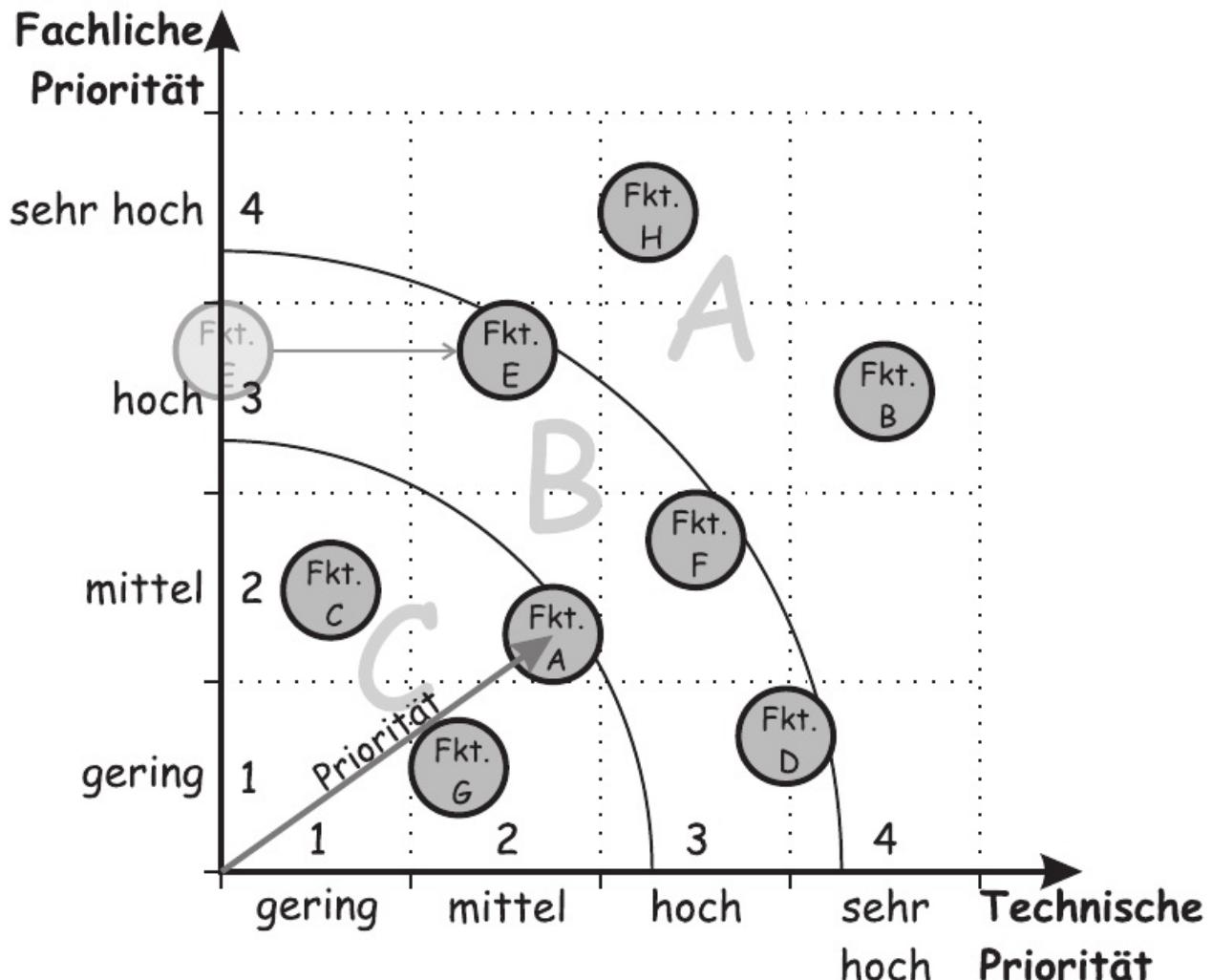
5.1.4 Prioritätsliste

Vielen fällt es schwer sich für die Wichtigkeit eines Themas zu entscheiden und halten lieber alles für gleich wichtig bzw. unwichtig. Um dies zu vermeiden, gibt es eine Technik mit der eine einzelne Person oder Gruppe ohne Bedenken priorisieren kann. Man schreibt dazu jedes Thema auf eine Karte und bringt die Karten auf einen Stapel in einen beliebigen Ausgangszustand. Als nächstes werden zwei Karten gezogen. Erscheint die obere Karte wichtiger als die andere wird sie in der ursprünglichen Reihenfolge wieder eingesortiert, andernfalls werden die Karten getauscht. Dies geschieht solange bis sich länger keine Veränderung mehr im Stapel ergeben hat. Dieser Stapel stellt dann das Ergebnis und damit die relative Priorisierung in der Liste dar. Bei starker vorheriger Vorprägung kann mit Hilfe eines Kollegen der Stapel sortiert werden, so dass man selbst die Karten nicht sehen kann und von ihm nur bei der Auswahl der Themen gefragt wird. Ebenfalls kann dies in Gruppenarbeit geschehen, so dass man reihum den Stapel sortiert und diskutiert bis ein gemeinsamer Konsens gefunden wurde. Sollte kein gemeinsamer Konsens gefunden werden, kann man mit Hilfe eines Prioritätskoordinatensystems arbeiten.

5.1.5 Prioritätskoordinatensystem

Bei zwei unterschiedlichen Ansichten in einer Priorität kann ein Prioritätskoordinatensystem als grafisches Verfahren helfen. Dabei erstellen zwei Parteien ihre Prioritätsliste. Anschließend wird eine der beiden Prioritätslisten auf ein vorbereitetes Koordinatensystem auf die y-Achse übertragen. Danach werden mit Hilfe der x-Achse mit der zweiten

Prioritätsliste die Karten verschoben.



Die gemeinsame Priorität ergibt sich dabei aus dem resultierenden Vector. Wenn Zahlen auf den Achsen hinterlegt sind, kann man nach den Satz des Pythagoras den Vektorwert mit folgender Formel berechnen:

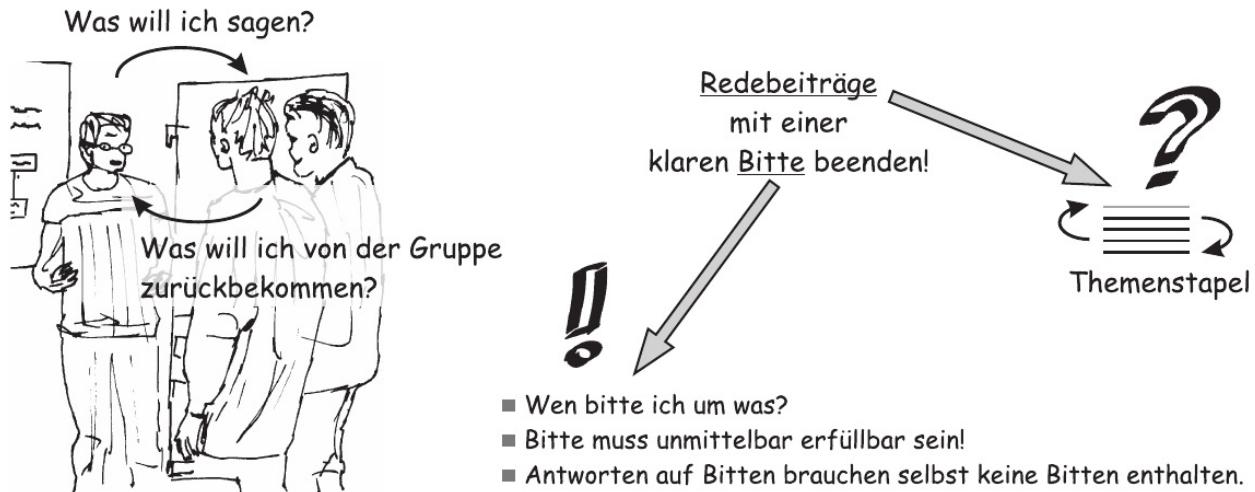
$$Prio_{ges} = \sqrt{Prio_{fachl}^2 + Prio_{techn}^2}$$

5.2 Effiziente Diskussionen

5.2.1 Um den Fokus bitten

In vielen Diskussionen geht der auf Ergebnis orientierte Fokus verloren, indem zwischen einzelnen Themen hin- und hergesprungen wird. Die klassische Lösung wäre es einen Moderator einzusetzen, um Struktur und Klarheit zu erlangen, ist aber nicht immer auf die Schnelle realisierbar. Problematisch gestalten sich häufig die große Anzahl an offenen Problemen, die abgehandelt werden müssen. Auch kurz zu betrachtende Aspekte werden länger als nötig offen gehalten. Um sich auf ein Ergebnis fokussieren zu können, könnte es

hilfreich sein, Redebeiträge mit einer konkreten Bitte abzuschließen. Diese Bitte sollte von den Teilnehmern sofort nachgegangen werden können. Dies ist besonders dann wichtig, wenn es sich um Fragestellungen handelt, in denen es um einen Informationsgewinn geht. Es ist zwar etwas ungewöhnlich, aber dennoch eine gute Steuerungstechnik für Diskussionen, wenn Fokussierung notwendig ist.



5.2.2 Seriell oder Multitasking

Aufgaben im Multitasking zu bearbeiten hängt von der Schwierigkeit der Aufgabe ab. Einfache Routineaufgaben können besser parallel abgearbeitet werden, als schwierige Aufgaben. Ebenso kommt es darauf an, wie verschieden die Aufgaben sind. Aufgaben, die in unterschiedlichen Hirnregionen ablaufen, sind leichter parallel abzuarbeiten, als Aufgaben, die dieselbe Hirnregion beanspruchen. Ebenso können unbewusste Prozesse das Multitasking behindert. Dies bedeutet, dass in einem Workshop Smartphones ausgestellt sein sollten, da die Nebentätigkeit an diesen die bewusste Informationsaufnahme ausbremsen. Ebenso ist es untersucht, dass Multitasking nur dann Zeit spart, wenn es sich um Routineaufgaben handelt. Deshalb sollte man bei einem Workshop Aufgaben in Form von Multitasking vermeiden. Den Themenstapel so klein wie möglich halten und ein serielles statt ein paralleles Arbeiten ermöglichen, um die größtmögliche Effizienz aus dem Workshop zu gewinnen.

5.3 Kreative Ideen in der Gruppe finden

5.3.1 Klassisches Brainstorming - Licht und Schatten

Brainstorming ist eine weit verbreitete Technik, um neue Ideen und Ansätze unabhängig von ihrer Bewertung zu finden. Allerdings wird Brainstorming in einer Gruppe schnell ineffizient. Eine größere Gruppe produziert deutlich weniger Ideen, als wenn Gruppenmitglieder einzeln Ideen entwickeln. Dies liegt zum einen daran, dass immer nur eine Person reden kann, aber

auch, dass viele Ideen nicht einmal als Anregung für andere Gedanken taugen. Zudem erzeugt das freie Denken eine Menge Ausschuss. Zu hinterfragen wäre: Wie kann man das besser machen?

5.3.2 Mit System querdenken und Leitfragen nutzen

Ohne Richtungsvorgabe für Denkanstöße ist die Aufgabenstellung eines Ideen-Workshops zu diffus, um brauchbare Resultate zu liefern. Drei McKinsey-Berater sind auf eine Liste mit 21 schlauen Fragen für Produktentwickler gestoßen, die auch als Leitfragen für ein Workshop in Softwarehäusern dienen können:

- Was ist für den Käufer oder Nutzer unseres Produkts oder unserer Dienstleistung der größte Nachteil, den er bewusst oder unbewusst in Kauf nimmt?
- Welche Anwender nutzen unser Produkt auf besonders ungewöhnliche Weise?
- Welche Kunden investieren mindestens 50% des Produktpreises, um es an ihre Bedürfnisse anzupassen?
- Wer kauft unser Produkt in überraschend großen Mengen?
- Wer kämpft mit denselben Problemen wie wir, aber aus einem anderen Grund? Wie gehen diese Unternehmen diese Probleme an?
- Welche Informationen über Kunden und Produktverwertung fallen als Nebenprodukt unseres Geschäfts an, die wir nutzen können, um ein anderes Geschäft radikal zu verbessern?
- Was ist das größte Ärgernis für unsere Kunden und Anwender beim Kauf oder Einsatz unserer Produkte bzw. Dienstleistungen?
- Welche spontanen Verbesserungen haben Kunden an unserem Produkt vorgenommen?
- Für welche unserer Kunden eignet sich unser Produkt oder unsere Dienstleistung am wenigsten?
- Welche Kunden bedient unsere Branche bewusst nicht und warum?
- Welche Kundengruppe wäre richtig groß, wenn wir ein Hindernis aus dem Weg räumen könnten, an das wir bisher noch nicht gedacht haben?
- Was würden wir anders machen, wenn wir vollständige Informationen über unsere Käufer, Anwender, Vertriebskanäle usw. hätten?
- Welche Technologien, die in unseren Produkten eingesetzt werden, haben sich seit der letzten Produktüberarbeitung am stärksten weiterentwickelt und verändert?
- Welche Technologien, auf denen unsere Produktionsprozesse beruhen, haben sich seit der letzten Produktionsprozessüberarbeitung am stärksten weiterentwickelt oder verändert?

Letztendlich sollt man die Anzahl der Teilnehmer in einem Ideenworkshop möglichst gering halten, um Kommunikationsengpässe zu vermeiden. Man sollt die Leitfragen vorbereiten, damit man die Qualität der Ideen erhöht. Auch ist es sinnvoll Teilnehmer einzuladen, die Antworten auf diese Leitfragen geben können, um ein Querdenken innerhalb des Workshops zu ermöglichen.

5.4 Gruppenfeedback geben

In mehrtägigen Workshops sind tägliche Feedbacks innerhalb einer Gruppe wichtig, z.B. laufende Prozesse zu verbessern und ein optimales Ergebnis zu erreichen. Oft ist aber nicht ausreichend Zeit dafür vorhanden oder die Gruppe dafür ist zu groß. Im Folgenden werden vier Techniken für Gruppenfeedbacks vorgestellt. Allerdings unter Einhaltung einer Grundregel: Wer nichts sagen möchte, der wird auch nicht gezwungen oder überredet, sondern gibt einfach das Wort an den Nächsten weiter.

5.4.1 Feedbackrunde

In einer Feedbackrunde sitzen alle im Kreis und geben in einer festen Reihenfolge oder frei durchgehend nacheinander ein individuelles Feedback. Dies kann ohne Vorgaben oder nach folgenden vier Regeln geschehen:

1. Was hat mich unterstützt?
2. Welche Stärken habe ich in der Gruppe wahrgenommen?
3. Was hat mich irritiert?
4. Was wünsche ich mir beim nächsten Mal anders?

Es ist hilfreich die Regeln für alle sichtbar auf einem Flipchart zu platzieren, damit die Feedbackgeber sich auf die Inhalte konzentrieren können ohne die Feedbackstruktur im Kopf behalten zu müssen. Der Vorteil der Feedbackrunde ist die Ausführlichkeit und Dynamik, die entsteht, wenn Feedbackgegner auf ihren Vorräder reagieren können, ohne dabei eine Diskussion zu beginnen. Es hat aber auch wiederum den Nachteil, dass es für die letzten Feedbackgeber oft frustrierend ist, nichts neues sagen zu können. Dies kann man entgegensteuern, indem man bei den Feedbacks bei der Reihenfolge täglich variiert.

5.4.2 Blitzlicht oder Streichholz-Effekt

Bei knapper Zeit kann man ein Kurzfeedback bzw. Blitzlicht durchführen. Dabei werden die Anzahl der Sätze, die eine Person sagen kann, oder die Redezeit begrenzt. Dadurch entsteht eine Fokussierung der Teilnehmer auf das ihrer Meinung nach Relevante. Allerdings ist auch das Risiko vorhanden, dass ein wichtiger Hinweis nicht erwähnt wird, da die Zeit zu knapp ist oder die Teilnehmer eine andere Priorisierung in ihren möglichen

Feedbacks vornahmen. Um die Funktionalität des Blitzlichts sicherzustellen ist eine Überwachungsrolle notwendig. Dies kann zum Beispiel der letzte Feedbackgeber oder der Nächste sein. Eine andere Möglichkeit ist der Streichholzeffekt. Bei dieser reicht man eine Packung Streichhölzer rum und jeder entzündet ein Streichholz und darf Feedback geben, solange das Streichholz brennt. Allerdings besteht die Gefahr, dass wegen dem Überraschungsmoment Inhalte verloren gehen und ist dies auch organisatorisch nicht überall möglich. Es eignet sich gut als Zwischenfeedback, um den Kopf frei zu bekommen und Abstand zu gewinnen vor einer Pause oder anderen Unterbrechungen.

5.4.3 Visuelles Feedback

Beim visuellen Feedback gibt man Klebepunkte auf ein oder mehrere Skalen ab. So kann die Anonymität durch die Feedbackgeber gewährleistet und bei mehrtägigen Workshops Tendenzen und Verbesserungen erkannt und dokumentiert werden.

5.4.4 Feedback über eine Aufstellung

Die emotionaliste Form ist eine Aufstellung der Teilnehmer im Raum. Dabei wird mit Hilfe von Klebeband oder ähnlichem ein Koordinatensystem oder eine Skala am Boden angedeutet. Jeder Teilnehmer muss dann sich so im Raum positionieren, dass es seinem Feedback entspricht. So wird das Feedback der Gruppe in einer Art Bild ermittelt und eignet sich für workshopfahrene Leute.

5.5 Wieder wach werden

Workshops sind anstrengend und ermüdend. Um entstehende Müdigkeit entgegenzusteuern helfen regelmäßige Pausen mit etwas Bewegung, ein eher leichtes Mahl am Mittag und Aufmunterungsspiele, die Geist und Körper aktivieren.

5.5.1 Kissenrennen

Dafür werden zwei Kissen (bei über 12 Personen 3 oder mehr Kissen) benötigt. Die Teilnehmer bilden dabei einen möglichst engen Stuhlkreis und zwei gegenüberliegende Plätze bekommen jeweils ein Kissen. Anschließend müssen die Kissen möglichst schnell an die Sitznachbarn links oder rechts weitergegeben werden. Jemand, der dann zwei Kissen gleichzeitig erhält, scheidet aus. Das Spiel endet, wenn nur noch zwei Teilnehmer übrig sind.

5.5.2 Finger fangen

Hierbei stellen sich die Teilnehmer in einem Kreis auf. Die linke Hand wird dabei mit der flachen Hand nach oben auf Höhe des Ellbogens gehalten und die rechte Hand drückt leicht mit dem Zeigefinger auf die flache Hand des Nachbarn. Bei einem Signal versucht man dann seine rechte Hand wegzuziehen und gleichzeitig mit der linken den Finger des Nachbarn zu fangen. Nach ein oder zwei Wiederholungen passiert es dann in umgekehrter Richtung. Das Spiel erfordert eine ungewöhnliche Konzentration und damit wird der Kopf wieder wach und angeregt.

5.5.3 Papier fangen

Hier muss jeder in einem freien Raum ein Blatt DIN-A4 Papier in der Hand balancieren. Bei einem Signal muss jeder Teilnehmer versuchen das Blatt Papier von anderen von der Hand zu fegen und dabei sein eigenes Blatt weiter in der Hand zu balancieren. Das Spiel endet, wenn nur noch eine Person ihr Blatt in der Hand balanciert. Auch hier wird Konzentration erfordert und hat den selben Effekt wie 5.5.2.

5.6 Ergebnisse sichern und Workshop abschließen

5.6.1 Blick zurück und nach vorne

Im Abschluss kann nochmal ein umfangreicher Feedback durch mehrere Feedbacktechniken oder viel Zeit für ein Feedback geschehen. Dies ist wichtig, um für kommende Workshops zu lernen, was mit der Gruppe oder welche Thematik gut funktioniert hat und was verbessert werden kann. Weiterhin sollt es einen Ausblick geben, der im Workshop initiierte Impulse gibt, die auch dauerhaft genutzt werden sollen können. Das wird in der Regel mit Aufgaben grob vorstrukturiert, die die Teilnehmer in absehbarer Zeit erledigen sollen. Wenn Bedenken dabei geäußert werden können sie ebenfalls als Risiken notiert und protokolliert werden.

5.6.2 Aus Problemen Risiken machen

Am Ende kann es passieren, dass es zu viele ungelöste Probleme gibt und diese eine lähmende Stimmung bei den Teilnehmern erzeugen. Um den entgegenzusteuern, kann man versuchen das ganze aus einem anderen Blickwinkel zu betrachten und die Probleme aus einer negativen-destruktiven Sicht in positiv-konstruktive Risiken umdefinieren. Statt zu sagen "wir haben keine Zeit dafür" sollt man sagen "Was brauchen wir, damit wir der Aufgabe nachkommen können?".

5.6.3 Fotoprotokoll als visuelles Anker

Das Fotoprotokoll dient den Teilnehmern als visueller Anker und wird in der Nachbereitung des Workshops erstellt und verteilt. Die im Workshop aufbereiten Aufgaben werden in ein Planungstool eingepflegt oder als Outlook-Aufgaben oder Erinnerungs E-mails verschickt. So kann man die Planung dokumentieren und es entsteht eine direkte Bindung und Verantwortung zu den Aufgaben.

5.7 Große Gruppen

Ein idealer Workshop besteht aus ca. 5 bis 7 Teilnehmern und ist auf jeden Fall kleiner als 12 Personen. Leider kommt es aber oft zu Workshops mit mehr als 20 Teilnehmern. Um dabei immer noch möglichst effizient und ergebnisorientiert zu arbeiten, werden hier folgende Ideen vorgestellt.

5.7.1 Organisatorische Rahmenbedingungen

Bei einer großen Workshopgruppe, sollten mehrere Räumlichkeiten vorhanden sein. Im Idealfall bestehen sie aus einem großen Raum und mehreren kleinen Räumen, sowie ein Pausenraum. Die Anzahl der Arbeitsräume richtet sich nach der Teilnehmerzahl, so dass sich die Teilnehmer in Gruppen mit maximal 7 Teilnehmern aufteilen können. Die Räume sollten nicht zu weit auseinanderliegen und man sollt ausreichend Arbeitsmaterial in jedem Raum vorrätig haben. Auch eignet es sich den Teilnehmern Freiraum zu geben, so dass sie in einen Garten oder ein anderes Stück Natur entkommen können, um Stress abzubauen. Auch leichtes Essen am Mittag oder ein reichhaltiges Menü am Abend, sowie gemeinsame Freizeitaktivitäten können hilfreich sein.

5.7.1 Effiziente Gruppenarbeit

Häufig ist das Problem, dass bei Teilgruppen verschiedene bzw. voneinander unabhängige Themen bearbeitet oder dieselben Themen bearbeitet werden und dabei eine geringe Akzeptanz bei der Zusammenführung der Ergebnisse bei den anderen vorhanden ist. Dies kann man wie folgt lösen:

- **Gruppenzusammenstellung:** Hier sollt man darauf achten, dass sich Vertreter jeder Interessengruppe in jeder Arbeitsgruppe befinden. Damit besteht eine hohe Chance Ergebnisse zu erarbeiten, die breit akzeptiert werden und nicht zu speziell sind.
- **Themenverteilung:** Man sollt das Arbeiten mit aufeinander aufbauenden Ergebnissen vermeiden. Auch ist es hilfreich Aufgaben zu vergeben auf die die Teilnehmer einer Gruppe nicht spezialisiert sind. Die Qualität der Ergebnisse ist dabei oft überraschend

und können zum Teil wiederverwendet werden.

- **Zwischenabgleiche:** Wenn zwei Aspekte eines Themas parallel behandelt werden müssen, sind Abgleiche zwischen den betroffenen Teilgruppen notwendig. Diese Abgleiche können dabei in 1 bis 5 Minuten in Intervallen von 15 bis 45 Minuten erfolgen.

Autor-Kritiker-Treffen: Ergebnisse zusammenführen

Selten ist es, dass man nur eine Sammlung von Ideen erarbeiten möchte, wenn zwei oder mehr Gruppen mit derselben Aufgabe betraut sind. Meist hat man das Ziel die Ergebnisse weiter auszubauen und einsatzfähig zu machen. Dazu hilft eine Adaption eines Autor-Kritiker-Treffens. Diese Adaption läuft in vier Stufen ab:

- Die Teilarbeitsgruppen (hier Autoren) stellen die zusammengehörigen Ergebnisse ohne inhaltliche Diskussionen nacheinander vor. Dabei sind nur Verständnisfragen erlaubt.
- Zuhörer (hier Kritiker) hören still zu und notieren sich die alle aus ihrer Sicht positiven und kritischen Aspekte der Präsentation.
- Wenn ein Thema abgeschlossen ist, also eine unabhängige Gruppe ihr Teilergebnis oder alle konkurrierende Teilgruppen ihre Ergebnisse vorgestellt haben, legen die Kritiker die gefundenen Aspekte dar. Dabei erst die positiven, danach die negativen. Die Autoren hören dabei nur still zu (außer bei Verständnisfragen) und rechtfertigen sich nicht.
- Die Autoren ziehen sich zurück und werten das Feedback aus. Der ganze Prozess benötigt einen Moderator, da hierbei Emotionen schnell hochgehen können. Ebenfalls wichtig ist es, die Kritikpunkte und die Auswertung zu dokumentieren, damit sie im Fotoprotokoll erscheinen. Bei wichtigen Entscheidungen ist diese Technik bewährt. Damit kann man Architektur und Designentscheidungen optimieren und in der Gruppe verankern, da sich jeder daran beteiligen kann. Bei komplexen Themen kann es sinnvoll sein, ein Autor-Kritiker-Treffen auch mal zwischendurch vor dem Endergebnis auf Basis von Zwischenergebnissen bzw. Teilergebnissen durchzuführen.

5.8 Selbstorganisation in Workshops - OpenSpace

5.8.1 Ideenvielfalt und motivierte Teilnehmer

Um effizient in einer größeren Gruppe zu arbeiten können, bietet es sich alternativ auch an einen Workshop stärker selbstorganisiert durchzuführen. Dabei bietet sich OpenSpace als Konzept bei 12 Teilnehmern oder mehr an. Dies ist eine Konferenzmethode, bei der es darum geht Ansätze für Veränderungsprozesse zu finden. Dabei können Teilnehmer ihre eigenen Themen einbringen und die jeweilige

Arbeitsgruppe dazu gestalten. Dies führt zu besonders schnellen und kreativen Ergebnisse und optimiert die Identifikation der Teilnehmer mit den Ergebnissen.

5.8.2 Fünf Regeln und zwei Rollen

Bei OpenSpace laufen meist drei oder mehr Workshops parallel ab. Dabei gibt es definierte Synchronisationspunkte und ein knappes Regelwerk. Um die Einhaltung dieser zu bewachen benötigt es folgende vier Leitlinien und einem Gesetz:

- **Wer kommt, ist die richtige Person!** Zu einer Veranstaltung kommen nur Teilnehmer, die auch Interesse an dem konkreten Thema verspüren.
- **Offenheit für das, was geschieht!** Man sollt neuen Ideen und unerwarteten Erkenntnissen bzw. Ergebnissen mit Offenheit begegnen.
- **Es beginnt, wenn die Zeit reif ist!** Ergebnisse oder Ideen kommen nicht auf Kommando. Daher wird der zeitliche Rahmen von der Gruppe selbst festgelegt.
- **Vorbei ist vorbei!** Wenn ein Thema voll erschöpft ist, ist es vorbei, auch wenn dafür mehr Zeit vorgesehen wurde.
- **Das Gesetz der zwei Füße.** Wenn eine Person nichts mehr zu einer Veranstaltung lernen oder beitragen kann, verlässt sie diese und sucht sich eine andere Gruppe. Das Gesetz führt zu folgenden zwei wertvollen Ausprägungen: **Hummel**: Dies ist eine Person, die an vielen Arbeitsgruppen teilnimmt. Sie wechselt die Arbeitsräume häufig und bleibt nur solange, wie sie auch etwas lernen und etwas beitragen kann.

Schmetterling: Dies ist eine unentschossende Person, die nicht viel zu etwas beiträgt und sich nicht sofort einer Gruppe anschließt. Oft verweilen sie länger an einem Ort, als andere, sind damit eine perfekte Ideenanlaufstelle und strahlen damit eine gewisse Ruhe und Stabilität in einer eher dynamischen Veranstaltung aus. Als Rahmen gibt es bei OpenSpace nur Leitfragen und keine feste Liste an Themen. Die Prioritäten werden von den Teilnehmern festgelegt. Daher ist es wichtig auch die Mitarbeiter des Kunden und Beraterkollegen dabei zu haben, die für die spätere Umsetzung und Realisierung verantwortlich sind, da nur sie die konkreten Impulse und das notwendige Engagement von Kundenseite mitbringen. Organistatorisch gibt es eine Matrix aus Arbeitsräumen und Zeitscheiben von ca. zwei Stunden Dauer von allen Teilnehmern, die Themen eingebracht haben. dafür sind große Metaplanwände und reichlich Karten bzw. lose Papierblätter geeignet. Nach der Planungsphase gehen alle Teilnehmer in die Workshops, die sie interessieren. Dabei kann sich die Planungstafel immer wieder verändern, da Workshops verschoben, zusammengelegt oder gestrichen werden müssen. Wichtig hierbei ist zu beachten, dass die Änderung nur vom Ersteller selbst vorgenommen werden darf. Jede Arbeitsgruppe ist selbst dafür verantwortlich, ihre Ergebnisse in geeigneter Form zu dokumentieren. Eine dreitägige Open-Space-Veranstaltung beginnt mit einem initialen Sammeln und Planen der Themen. Danach treffen sich die einzelnen Arbeitsgruppen. An den jeweiligen Abenden und Morgenden

der Tage, gibt es als gemeinsame Veranstaltung die Abendnachrichten und Morgennachrichten, wo ein Abgleich zwischen allen Teilnehmern vorgenommen wird. Danach werden die Ergebnisse zusammengeführt und eine weitere Umsetzung wird geplant. Der dritte Tag endet dann mit einer gemeinsamen Abschlussrunde. Weiterhin sind auch Anpassungen am OpenSpace möglich, damit man konkrete Ziele mit vielen Teilnehmern erarbeiten kann oder Voraussetzungen an einen richtigen OpenSpace nicht gegeben sind.

5.8.3 Modifizierter OpenSpace

Um die Stärken des OpenSpace-Konzepts zu nutzen und gleichzeitig sicherzustellen, dass bestimmte Themen mit erforderlichen Ergebnissen bearbeitet werden, besteht eine Möglichkeit darin, weitere Regeln hinzuzufügen, um bestimmte Ziele zu erreichen ohne dabei die Freiheit des OpenSpace zu verlieren.

Vorbereitung Vorab werden Themen gesammelt. Dabei kann jeder Teilnehmer seine Themen in ein geeignetes Medium wie z.B. ein Wiki einstellen. Kurz vor Workshopbeginn werden dabei die Pflichtthemen extrahiert. Die Anzahl der Pflichtthemen sollten dabei 15-20 nicht überschreien, um genügend Freiraum für den Open-Space-Charakter zu überlassen. Weiterhin werden Raum-Zeit-Matrixen als Wände vorbereitet. Dabei können Zeitslots mit verschiedenen Längen angeboten werden. Wenn längere Zeiten benötigt werden, können mehrere Zeitslots belegt werden. Nach jedem oder jeden zweiten Slot sollt eine Ergebnispräsentation und eine Abstimmung der weiteren gewünschten Themen eingeplant werden. Die Präsentation darf dabei nicht mehr als fünf Minuten dauern.

Initiale Planung Zu Beginn eines Workshops werden die bereits bekannten Themen gelistet und weitere neuen Themen gesammelt. Jeder Verantwortliche eines Themas stellt es im Plenum kurz vor und schreibt spätestens dann die dazugehörige Karte und versiert sie dabei mit seinem Kürzel und der gewünschten Dauer seines Zeitslots. Abschließend werden maximal 20% der Themen zum Beispiel durch Führungskräfte in Abstimmung mit Beratern oder der Gruppe als besonders wichtig gekennzeichnet. Diese werden dann in der Planung als Erstes den Slots in der Matrix zugeordnet. Damit wird sichergestellt, dass diese Themen auf der Planungswand landen, wenn es mehr Themen als freie Slots gibt. Nun beginnt die Planung. Zusätzliche notwendige Ausstattungsmerkmale wie z.B. Beamter können ebenfalls auf der Karte und bei der Raumauswahl berücksichtigt werden. Sind mehr Themen vorhanden, als es Slots gibt, wird ein Themenpool auf einer Wand aufgebaut. Wichtig ist es, den ersten Tag konkret zu planen. An den Folgetagen kann sich abhängig von den Ergebnissen noch viel ändern.

Durchführung Nach der Planung kann mit der Durchführung begonnen werden. Dabei dürfen die Zeitslots nicht überschritten werden. Durch die Präsentation und Zwischenergebnisse erfolgt eine gute Ergebnisfokussierung und ermöglicht es anderen

Teilnehmern noch zu späteren Sessions hinzuzustoßen. Ein Ergebnis muss noch während der Session erstellt werden. Dabei eignet sich das Verlaufs- und das Ereignisprotokoll. Bevor die nächste Session beginnt wird die Planungswand aktualisiert. Nach Absprache mit den Themenverantwortlichen können Sessions oder Räume getauscht werden. Auch kann es sich ergeben, dass ein Thema gestrichen wird, da es keinen Teilnehmer interessiert oder viele noch kurzfristig zu einem anderen Thema gehen. Dies ist jedem freigestellt. Auch darf sich ein Verantwortlicher noch kurzfristig für ein anderes Thema entscheiden. Das heißt kurzfristige Anpassungen sind erlaubt. Auf diese Art und Weise laufen die Workshops zum großen Teil selbstorganisiert, ergebnisfokussiert und mit viel Engagement ab. Es besteht nicht die Gefahr der Langweile, da die Motivation der Teilnehmer durch die große Flexibilität in der Planung in der Regel sehr hoch ist und man dementsprechend auf die neuen Erkenntnisse und Bedürfnisse der Teilnehmer reagieren kann.

OpenSpace in der Beratung Mit OpenSpace wird in der Regel nicht sehr häufig gearbeitet. Bei Projektvorbereitungen oder zu Beginn von Projekten kann es allerdings sehr wertvoll werden, um in wenigen Tagen einen umfassenden Blick auf das Projekt zu gewinnen. Man kann dabei Risiken und Möglichkeiten analysieren und eine erste Projektplanung und Realisierungsidee entwerfen. Bei einen solchen ein- bis dreitägigen Workshop kann sich ein Projektteam gut kennenlernen und externe Berater integrieren. Häufig ist es auch so, dass noch weitere erfahrene Experten von Kunden oder Beraterseite hinzugenommen werden, die für das Projekt nicht zur Verfügung stehen. Diese können ihr Wissen mit dem Team teilen und in die Projektartefakte, wie eine Grobplanung, ein Product Blackout und eine Risikoliste, mit einfließen lassen.

6 Methodische Beratung

6.1 Beratung ist nicht gleich Beratung

Das Troja-Prinzip in der methodischen Beratung

Mit dem Begriff des Troja-Prinzip, wird die evolutionäre sowie die revolutionäre Veränderung in einer Gruppe oder der Organisation im Laufe der Zeit beschrieben.

6.2 Unsere Kunden methodisch beraten

Methodische Beratung - ist das nicht Coaching?

Die Methodische Beratung ist dem Coaching sehr ähnlich, beide haben das Ziel eine individuelle Lösung für den Kunden zu erarbeiten. Doch der Unterschied liegt in der Rolle des Experten.

Prozessberatung: Mit der Prozessberatung ist der Aufbau der Beziehung zwischen dem Berater und dem Kunden gemeint, welche es dem Kunden ermöglicht, intern und extern auftretende Ereignisse wahrzunehmen, zu verstehen und darauf reagieren zu können.

Expertenberatung: Bei der Expertenberatung liefert der Berater sein Fachwissen und der Kunden kennt sein Problem schon und hat einen groben Plan zur Lösung dieses Problems. Die Aufgabe des Beraters ist es hierbei, die Informationen zu beschaffen und die Lösung mit den Mitarbeitern zu erarbeiten.

Ambiguitätstoleranz

Im Rahmen einer Beratung kommt es oft zu dem Punkt, das der Berater mehr möchte als der Kunden. Wenn dieser Punkt erreicht wird, benötigt der Berater eine gute Ambiguitätstoleranz.

Unter Ambiguitätstoleranz versteht man die Fähigkeit, die Punkte die man als Berater gerne verändern würde aber der Kunde nicht möchte, so zu lassen wie sie aktuell sind.

Aufgaben eines methodischen Beraters

Ein methodischer hat fünf Aufgaben zu erfüllen:

- Eine kooperative Beziehung zum Kunden aufbauen.

- Klare Beratungsziele mit dem Kunden setzen.
- Den Kunden auf ein lösungsorientiertes Handeln ausrichten.
- Den Kunden lösungsorientiert beraten.
- Den Kunden dabei unterstützen, dass seine Mitarbeiter an den Veränderungen dauerhaft festhalten.

6.3 Erfolgsbarrieren überwinden

Erfolgsbarrieren auf Beraterseite

Die größte Gefahr für Erfolgsbarrieren auf Seiten des Beraters liegen darin, dass der Berater mehr machen möchte als der Kunden. Wenn dies passiert mangelt es dem Berater an Ambiguitätstoleranz.

Beratungsresistenz - Erfolgsbarrieren auf der Kundenseite

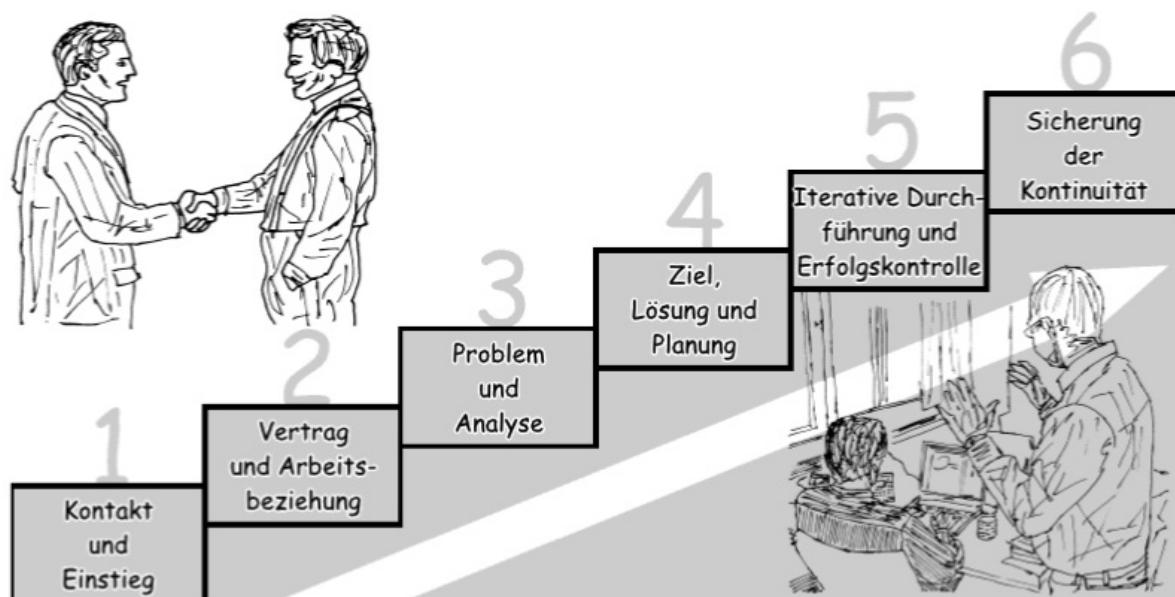
Für das Problem der Beratungsresistenz steht meistens einer von zwei Konflikten im Fokus.

1. Alleine durch die Anwesenheit eines Beraters wird dem Personal des Kunden klar, dass es Probleme gibt und er auf Hilfe angewiesen. Dies führt dann dazu, dass der Kunde ein Gefühl von Hilflosigkeit entwickelt.
2. Es kann zu Differenzen in den Analyseergebnissen zwischen dem Kunden und dem Berater kommen und der Kunde dadurch das Problem und die Lösungswege deutlich anders sieht als der Berater.

7 Methodische Beratung als Prozess

7.1 Sechs Stufen zur erfolgreichen Beratung

Zu einer erfolgreichen Beratung gehören mehrere Stufen, die folgenden Grafik visualisiert diese verschiedenen Stufen.



(6 Stufen der Beratung)

Kontakt und Einstieg

In dieser Phase kommt der erste Kontakt mit dem Kunden zustande, man lernt sich kennen und bekommt einen ersten Eindruck über die Bedürfnisse des Kunden.

Außerdem bekommt der Kunden einen Überblick über die Qualifikationen des Beraters.

Daraufhin werden gemeinsam die Probleme identifiziert, woraus sich dann eine mögliche Zusammenarbeit ergibt.

Vertrag und Arbeitsbeziehung

In der zweiten Phase wird der Vertrag ausgearbeitet und abgeschlossen. Dazu gehört das Festlegen eines ersten groben Ziels, in dem definiert wird, welche Ergebnisse erstrebenswert sind und welche Ergebnisse erreicht werden können. Dadurch wird ein gegenseitiges Verständnis geschaffen. Außerdem wird die Art und Weise der Zusammenarbeit geregelt, sowie die Zeitspannen für die Meilensteine.

Nach Vertragsabschluss beginnt der Aufbau der Arbeitsbeziehung.

Problem und Analyse

In dieser Phase werden die Probleme aus Phase 1 analysiert. Dazu werden Faktoren bestimmt, die das Erreichen der Ziele fördern oder gefährden können. Daraus werden Zwischenziele definiert, sowie alternative Lösungsideen um die Zwischenziele zu erreichen und Entscheidungskriterien um zwischen den Alternativen entscheiden zu können.

Ziel, Lösung und Planung

In Phase vier wird der iterative Beratungsprozess zum Erreichen der Ziele eingeleitet. Dazu wird die Vorgehensweise festgelegt.

Iterative Durchführung und Erfolgskontrolle

In dieser Phase werden Informationen über den Projektfortschritt gesammelt, es wird die Zielerreichung überprüft und es werden tägliche Entscheidungen mit dem Kunden getroffen. Mit Hilfe dieser drei Punkte wird der Beratungsprozess gesteuert und durch die erhaltenen Informationen kann die Planung an das Beratungsprojekt angepasst werden.

Sicherung und Kontinuität

In der letzten Phase wird die Beratung beendet. Es wird möglicherweise eine periodische Überprüfung geplant, um sicherzustellen, dass die Ergebnisse aus dem Projektverlauf umgesetzt und weiterentwickelt werden.

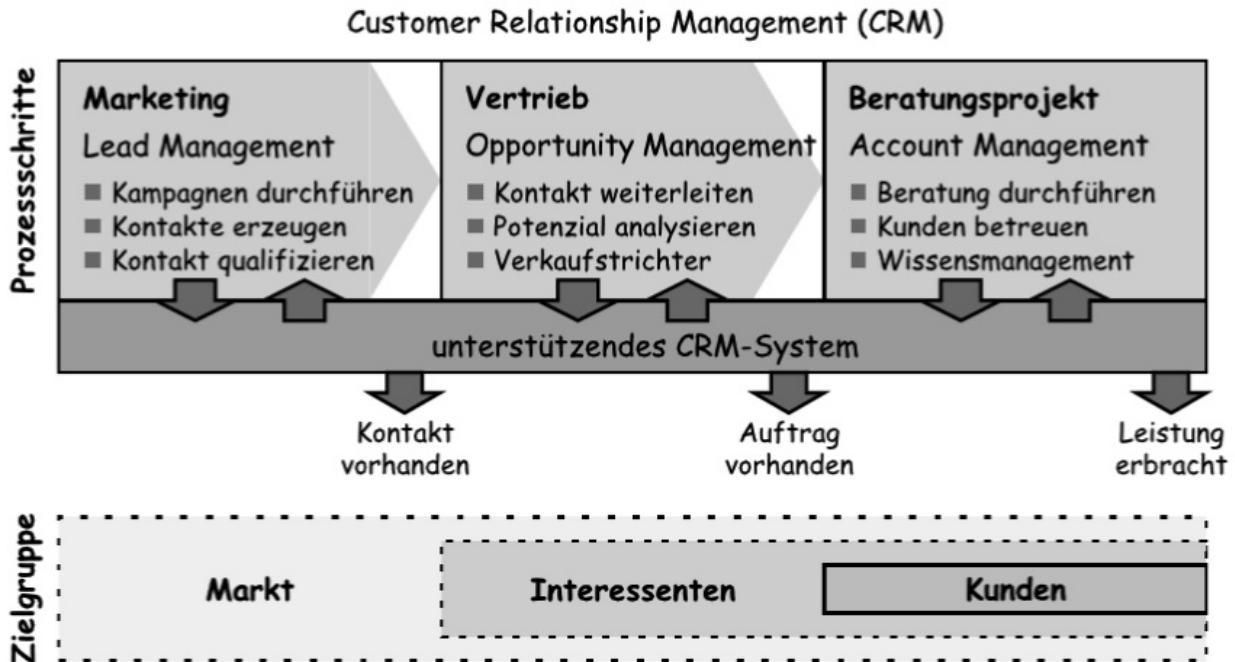
7.2 Marketing, Vertrieb und Projektgeschäft

Produkthersteller oder Dienstleister?

Die Firmenstrategie ist ein wichtiger Punkt in einem Unternehmen und ein wesentlicher Erfolgsfaktor. Deshalb sollten sich Unternehmen auf ein Strategie festlegen. Wenn man sich als Unternehmen als Produkthersteller sieht, ist die oberste Einheit das Produktmanagement und alle anderen Einheiten wie z.B. der Vertrieb ordnen sich unter. Sieht man sich als Unternehmen aber als Dienstleister, ist der Vertrieb die oberste Einheit und alle anderen ordnen sich unter.

Wie spielen Marketing, Vertrieb und Projekt zusammen?

Die drei Organisationseinheiten sind so verbunden, dass das Marketing den Markt adressiert um Interessenten zu finden. Der Vertrieb ermittelt dann aus den Interessenten mögliche Kunden, für welche dann durch die Beratungsprojekte Leistungen erbracht werden. Die folgende Grafik visualisiert noch einmal das Zusammenspiel der drei Organisationen.



(Zusammenspiel von Marketing, Vertrieb und Beratungsprojekt)

Bestandsakquise vs. Neuakquise

Die sehr hohe Bedeutung einer guten Zusammenarbeit zwischen Marketing, Vertrieb und Beratung wird durch den Aufwand bei der Akquise von Neu- oder Bestandskunden noch wichtiger. Das Erzeugen neuer und qualifizierter Kontakte durch den Vertrieb und das Marketing ist deutlich teurer als die Akquise für Bestandskunden.

Vorbereitung auf ein Akquisegespräch

Zur Vorbereitung auf ein Akquisegespräch sollte man ein Ziel festlegen. Außerdem sollte man grob planen, wie das Gespräch ablaufen soll. Des Weiteren sollte man sich Gedanken darüber machen, wie man die eigene Position und die eigene Firma definiert und warum die eigenen Stärken gut zum Kunden passen und warum man besser ist als seine Mitbewerber.

7.3 Prozesswerkzeuge

Prozesswerkzeuge helfen bei der Analysierung von Problemen. Dadurch erhält der IT-Berater ein besseres Verständnis und kann somit effektiv seine beratende Rolle einnehmen.

7.3.1 Das Kontextmodell

Das Kontextmodell bietet eine Grundlage um eine Struktur in ein Beratungsgespräch beim Festlegen der Auftragsanforderungen zu bringen. Durch diese Struktur wird vermieden, wichtige Details zu vergessen und es schränkt die Kommunikation nicht ein. Dabei werden sechs Bereiche behandelt:

- Rahmen: Welche Risiken, Hindernisse oder Rahmenbedingungen sollten beachtet werden?
- Vorhandenes: Was ist bereits vorhanden und wie wurde dies erreicht?
- Benötigtes: Was wird noch benötigt und wie wird dies besorgt?
- Ziele: Welche Ziele sollen erreicht werden und welcher Nutzen soll geschaffen werden?
- Ergebnisse: Welches Ergebnis wird erwartet?
- Aktivitäten: Welche Schritte bedarf es für die Umsetzung?

7.3.2 Das Systemmodell in der Beratung

Das Systemmodell ist mächtiger als das Kontextmodell. Hiermit können ganze Unternehmen bzw. Bereiche analysiert und strukturiert werden. Es werden die Angestellten betrachtet und welche Fähigkeiten und Bedürfnisse diese mitbringen. Ebenfalls werden die Arbeitsmittel der Firma betrachtet, was die Technik, Investitionsmöglichkeiten und Räumlichkeit mit einschließt. Wichtig ist ebenfalls einen Blick auf die Prozessabwicklung und Struktur der Organisation zu werfen.

7.3.3 Retrospektive und Supervision

Mithilfe von Retrospektive und Supervision kann die Qualität und Anpassbarkeit (z.B. aufgrund neuer Anforderungen) während Projekten gesteigert werden. Ebenfalls wird mit den Verfahren die Arbeitsweise analysiert.

Um Retrospektiven durchzuführen, kann ein einfaches Koordinatensystem verwendet werden. Hierbei beschreibt die x-Achse die Zeitabschnitte und die y-Achse die Bewertung (z.B. bei vielen Schwierigkeiten wird das Ereignis unter der x-Achse platziert). Ebenso wird vermerkt, was besonders gut bzw. besonders schwer verlief, verbessert werden kann und wo Hindernisse erkannt wurden. Zu beachten ist jedoch, dass um Verbesserungen bewirken zu können, erst nach dem tatsächlichen Problem gesucht werden muss, um dann eine passende Maßnahme durchzusetzen. Die Maßnahmen sollten hierbei immer schnell umsetzbar sein. Maßnahmen lassen sich auch direkt aus Hindernissen ableiten, so können intern im Team bei der Zusammenarbeit Hindernisse entstehen, die durch spezielle Maßnahmen zu lösen sind oder in den Rahmenbedingungen des Projektes, welche durch spezielle Managementrollen, welche sich der Lösung widmen, zu kompensieren sind.

Die Supervision widmet sich der professionellen Reflexion der Arbeitsweise. So kann der Supervisor zu Rate gezogen werden, um zu erfahren, wie bestimmte Handlungen am besten durchgeführt werden sollten. Um dies zu erreichen wird erst die Sichtweise des Ratsuchenden aufgezeigt und dessen Interpretation der Lage. Dann werden verschiedenste Optionen mit dem Supervisor durchgegangen und ausdiskutiert.

7.3.4 Kundenkultur verstehen

Die Firmenkultur eines Kunden ist in der IT-Beratung von großer Bedeutung. Lediglich Gespräche mit der Geschäftsleitung darüber zu halten ist häufig nicht ausreichend. Zu beachten ist, dass die Antworten meist einem Ideal entsprechen, welches so nicht in der Firma vorherrscht. Es werden nun Ansätze aufgezeigt um ein genaueres und aussagekräftigeres Bild über die Kundenkultur zu erlangen.

Die Bürogestaltung zeigt erste Indizien der Kundenkultur auf. So kann der IT-Berater auf die Dekoration, Arbeitsbedingungen usw. achten um einen Einblick zu erlangen.

Weitere Indizien lassen sich in bestimmten Ritualen finden, beispielsweise bei dem gemeinsamen Gang in die Mittagspause. Dies symbolisiert eine Zusammengehörigkeit im Unternehmen.

Ebenso bietet die Analyse des Belohnungssystems Einblicke in das Arbeitnehmnerbild im Unternehmen. Zu beachten sind Art und Bedingung für eine Belohnung und in wie Fern es den Arbeitnehmer unter Druck setzt.

Die interne Kommunikation ist ebenfalls ein wichtiger Bestandteil. Wichtig ist es darauf zu achten, wie kommuniziert wird und wer bei kniffligen Fragen zu Rate gezogen wird.

Einen guten Überblick über die Zufriedenheit der Arbeitnehmer und dem Interesse der Führungskräfte Arbeitnehmer in dem Unternehmen zu behalten, lässt sich durch das Betrachten der Fluktuation ermitteln. Ist diese sehr hoch, so spricht es meist für eine hohe Unzufriedenheit der Arbeitnehmer. Eine geringe Fluktuation muss jedoch nicht immer für eine hohe Zufriedenheit sprechen, Wirtschaftskrisen und fortschreitendes Alter der Arbeitnehmer lösen denselben Effekt aus.

Weitere Aspekte sind die internen Prioritäten und die Machtausübung. Es sollte analysiert werden wer Prioritäten setzt und ob diese überhaupt wahrgenommen oder überschritten werden. Ebenso sollte überprüft werden wer Macht ausübt und inwieweit die Machtausübung von den anderen Arbeitnehmern ernst genommen wird.

8 Der Sinn in unserer Arbeit

8.1 Der Mensch hinter der methodischen Beratung

Wichtig als Berater ist es, die hier aufgeführten Methodiken nur als Hilfestellung anzusehen. Die wirkliche Qualität steckt in der Persönlichkeit des Beraters.

8.1.1 Erdbeermarmelade: effektiv und effizient

Als Berater wird viel Geduld, Engagement und eine starke Persönlichkeit benötigt um bei Kunden einen souveränen Eindruck zu hinterlassen. Die aufgeführten Techniken können situationsbedingt Unterstützend eingesetzt werden um eine, sich für den Kunden lohnende, Dienstleistung anzubieten.

8.1.2 Unsere Beratungsleistung optimieren

Die im vorherigen Verlauf beschriebenen Methodiken zur Analyse und Reflexion von Arbeitsschritten lassen sich ebenfalls auf die eigene Dienstleistung anwenden. Durch regelmäßige Analysen kann eine erhöhte Qualität der eigenen Beraterleistung geschaffen werden. Ebenfalls ist die Fähigkeit Kunden einzuschätzen enorm wichtig um die Leistung zu steigern. Diese Fähigkeit wird durch Erfahrung gewonnen. Fehleinschätzungen können den Erfolg der Beratung stark beeinträchtigen. Ebenso sollte bei der Beratung nie das eigentliche Ziel aus den Augen verloren werden, beispielsweise beim Festbeißen in der Optimierung einer Tätigkeit sollte immer hinterfragt werden, ob dies wichtig für die Zielsetzung des Kundens ist.

8.1.3 Balance zwischen Beruf und Privatleben finden

Ein IT-Berater wird aufgrund seiner wichtigen Beratungs- und Veränderungsrolle sehr stark eingespannt. Es muss jedoch deutlich gemacht werden, dass die Veränderung erst durch die Akzeptanz des Unternehmens erfolgreich ist. So ist die Aufgabe des Beraters Optimierungsvorschläge zu erteilen und sich dann etwas zurückzuhalten um zu prüfen, z.B. mithilfe von Retrospektive, ob diese durchgesetzt werden.

8.2 Ethik in der Beratung und im Coaching

Ein IT-Berater sollte stets die Ziele des Kunden im Auge behalten, bzw. bei der Zielsetzung des Kundens unterstützen. Dabei ist zu beachten ob die Lösungen, um die Ziele zu erreichen, eher auf gemeinschaftlicher oder individueller Ebene realisiert werden sollte.

8.2.1 Das menschliche Verhalten

Ein wichtiger Punkt als Berater ist es, sich auf die Ansichten des Kundens einzulassen und diese zu analysieren. Als nächstes sollten Veränderungen vorgeschlagen werden, welche die ursprünglichen Ansichten verbessern könnten. Dabei ist zu beachten, den Kunden nicht zu bedrängen oder die eigene Meinung aufzuzwingen, sondern respektvoll und unterschwellig zu versuchen, den Kunden in eine neue Richtung zu lenken. So liegt die Hauptaufgabe oftmals darin, den Kunden auf bestimmte evtl. bereits vorhandene Ressourcen oder Lösungswege, aufmerksam zu machen und ihn damit zur Veränderung zu bewegen.

8.2.2 Kommunikation und Steuerung

Eine respektvolle und konstruktive Kommunikation ist für die Berater-Rolle essentiell wichtig. Fühlt sich der Kunde beispielsweise persönlich angegriffen aufgrund von Schuldzusprüchen, ist dieser weniger bereit neue Möglichkeiten einzugehen oder fühlt sich, als würden neue Möglichkeiten von dem IT-Berater aufgezwungen werden.

8.2.3 Den Kunden professionell beraten und coachen

Es sollte dem Kunden nicht die Möglichkeiten, welche als IT-Berater in dem Unternehmen gesehen werden, aufgezwungen werden, sondern der IT-Berater sollte sich an der Zielsetzung des Kundens orientieren und dabei unterstützen. Es sollten keine Planungen und Handlungen vollzogen werden, für die keine explizite Bewilligung erteilt wurde.

9 Unternehmenskultur greifbar machen

Die Unternehmenskultur ist die Zusammenfassung aller Verhaltensregeln in einer Organisation. Als IT-Berater ist es wichtig diese Unternehmenskultur kennen zu lernen und zu analysieren.

9.1 Graves-Modell - Entwicklung als Spirale

9.1.1 Einführung

Das Graves-Modell befasst sich mit der Entwicklung von Menschen in Bezug auf die zu lösenden Probleme. Die einzelnen Entwicklungsstufen werden als Memen bezeichnet. Dabei beschreibt ein Mem eine Information (z.B. Verhaltensweisen, Ideen usw.) welche durch die Kommunikation ausgetauscht wurde. Es handelt sich nicht zwangsläufig um zwei Personen die sich Informationen weitergeben, es können auch Unternehmen oder Gruppierungen sein. Im Graves-Modell sind diese Memes als Wertesysteme mit festgelegten Regeln (Denkweise, Umweltwahrnehmung, Verhaltensweisen) zu verstehen. Im Verlauf der Entwicklungsschritte werden hilfreiche Memes weitergetragen und neue Eigenschaften hinzugefügt. Dieses Modell findet ebenfalls in der IT-Beratung Anwendung, beispielsweise beim Austausch über Kunden bzw. bei der Kommunikation mit einem Kunden um die derzeitige Lage und mögliche Optimierungsansätze aufzuzeigen.

9.1.2 Meme in Unternehmen

Es werden nun verschiedene Memes aufgezeigt und ihre Vor- und Nachteile genannt. Es bedarf einer konkreten Analyse um auszusagen, ob eine Entwicklung in eine neue Stufe Risiken minimiert und Vorteile einbringt. Hierfür gibt es keine pauschale Antwort, es muss immer je nach Unternehmen entschieden werden. Alle der hier aufgeführten Memes finden Verwendung in Unternehmen.

So gibt es beispielsweise Memes, welche sich durch einen Anführer und einer streng hierarchischen Struktur auszeichnen. Untergebene müssen sich den Vorgesetzten untergliedern. Diese Systeme bringen eine klare Strukturierung, jeder weiß wem zu gehorchen ist und wer befehligt werden darf. Das sorgt ebenfalls für eine schnelle und stabile Entscheidungskraft. Es kann ein zerstörerisches System sein, welches die Flexibilität stark leiden lässt und die Struktur wichtiger als das Ergebnis empfindet.

Andererseits gibt es leistungsorientierte Memes, welche nicht strikt vorgeben welche Person welche Führungsposition einnimmt, sondern es wird dynamisch der am besten für die Position passende, ausgewählt. Hierdurch wird die Effizienz gesteigert und zielorientiert vorgegangen. An vielen wirtschaftstarken Unternehmen wird so vorgegangen.

Es gibt ebenfalls das gemeinschaftsorientierte Meme, welches sich zum Wohle jeden einzelnen einsetzt. Alle haben den gleichen Einfluss und die gleichen Regeln. Hierbei werden Entscheidungen in Gruppen getroffen und nicht durch einzelne. Dieses System bietet eine sehr tolerante Haltung, welches die Kreativität und Fähigkeiten jedes einzelnen nutzt, ohne Individuen einzuschränken. Der große Nachteil ist hierbei, dass Entscheidungen einen langen Prozess erfordern, da es gemeinschaftlich abgesegnet werden muss.

Zuletzt wird noch das Integration Meme betrachtet. Hierbei gibt es selbstorganisierende Teams, welche einen Teamleiter besitzen. Die Stärken und Schwächen einzelner Teammitglieder sind bekannt und werden versucht optimal zu kombinieren. Dies ermöglicht eine erhöhte Flexibilität und geht auf die Stärken einzelner Teammitglieder ein, jedoch gibt es hierbei die Gefahr, den Überblick zu verlieren und das Ziel zu verfehlten.

9.2 Theorie U

9.2.1 Warum noch eine weitere Theorie?

Mithilfe der Theorie U können Entwicklungsprozesse und Zukunftsprognosen dargestellt werden. Es hilft dabei, Probleme von anderen Sichtweisen anzugehen und somit neue Lösungswege aufzubauen. Dazu wird ein Blick in die Zukunft mit einbezogen.

9.2.2 Der Innovationsprozess

Theorie U stellt einen unterstützenden Prozess bereit mit dessen Hilfe Innovationen in Teams erarbeitet werden können. Dabei werden einzelne Schritte durchlaufen welche hier aufgezeigt werden.

Der erste Schritt beginnt damit, dass überprüft wird, ob eine Problemstellung bereits in der Vergangenheit existiert hat und ob die damalige Lösung wiederverwendet werden kann oder eine verbesserte Lösung gesucht werden sollte.

Als nächstes wird das Problem nicht nach den gewohnten Handlungen betrachtet, sondern versucht es zu verstehen und sich so neue, effizientere Lösungen herzuleiten.

Das Hineinversetzen in die Bedürfnisse und Gefühle des Personals, welches sich mit der Problemlösung beschäftigt, wird als nächstes durchgeführt.

Durch das Befassen mit der Problematik aus den vorherigen Stufen und des Verständnisses der Perspektiven von anderen Beteiligten, kommen neue innovative Ideen.

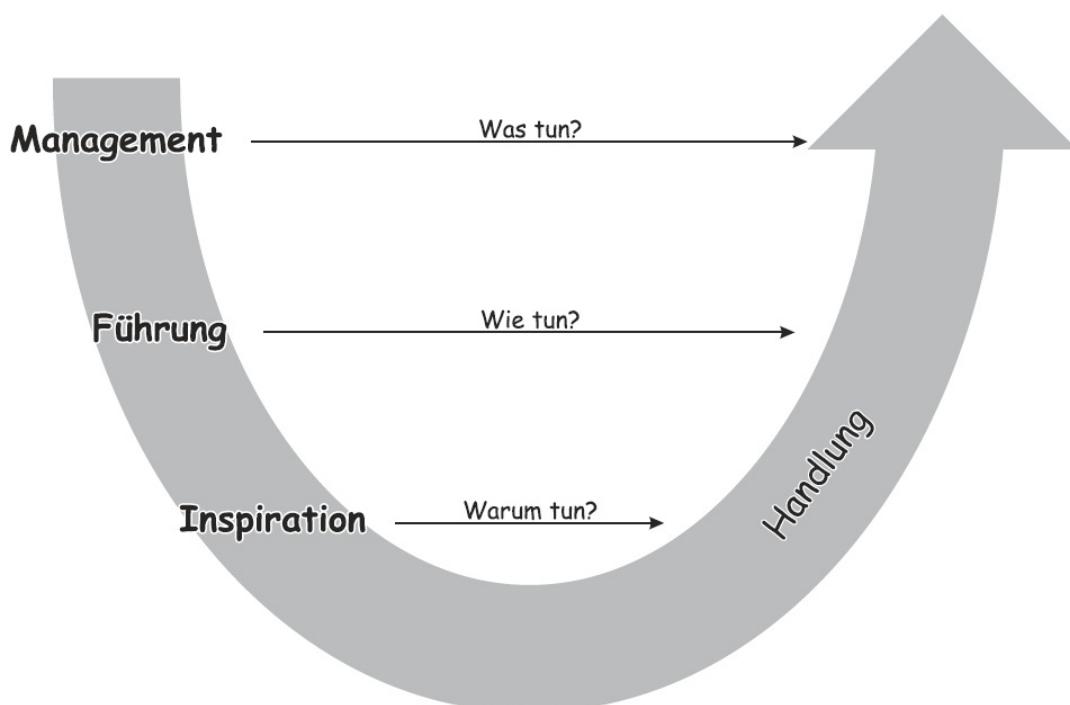
Die restlichen Schritte werden meist im Verlauf der vorherigen bereits durchlaufen. Diese beschreiben die Motivation da eine mögliche, dem Problem entsprechende Lösung gefunden wurde, die Entwicklung eines Prototypens und am Ende die eigentliche Implementierung.

9.2.3 Theorie U und der Coaching-Prozess

Beim Coaching werden die Prozesse, welche im vorherigen Kapitel dargestellt wurden, ebenfalls durchlaufen. Erst werden von dem Berater Informationen erfragt, wie beispielsweise die typische Herangehensweise bei solchen Problemen. Danach wird der Berater verstehen wollen, wie der Kunde das Problem überhaupt wahrnimmt. Ab diesen Stufen beobachtet der Berater nur und betrachtet das Wohlbefinden des Kundens während der Problemlösung bzw. versetzt sich in die Lage des Kundens. Im weiteren Verlauf beteiligt sich der Berater und hilft beim Finden neuer Impulse.

9.2.4 Konsequenzen für die Führung und Beratung

Die Betrachtung der Führung in einem Unternehmen bietet Rückschlüsse auf dessen Unternehmenskultur. Dabei ist zu untersuchen, auf welcher Stufe der Theorie U gerade gehandelt wird und wie diese Befehle bei den Mitarbeitern ankommen. Wird das Theorie U Modell auf die Führung angewandt, so lassen sich drei Stufen ableiten.



Das Management gibt eine eindeutige Zielstellung und kontrolliert das Erreichen dieser Ziele. Im Vordergrund steht stets die Produktenwicklung. Dies ist ein veraltetes Führungssystem, da es zu statisch an die Rahmenbedingungen herangeht.

Die Führung beschreibt das Einbinden von Mitarbeitern in den Managementprozess. Hierbei stehen die Kundenbedürfnisse im Vordergrund. Dieses Führungssystem zeigt sich bei Dienstleistungsunternehmen. Außerdem ist das Führungsprinzip den heutigen Herausforderungen gewappnet, kann jedoch in naher Zukunft unzureichend sein.

Die letzte Stufe ist die Inspiration. Diese ist eine Mischung aus den beiden anderen Stufen und ergänzt eine starke Motivation der Führungskräfte, welche sich auf die Mitarbeiter abfärbt. Dieses System iteriert häufiger über das U-Modell und bietet eine zukunftssichere Perspektive für das Unternehmen. Das System setzt auf eine starke Kundennähe und bindet diesen aktiv in den Prozess mit ein.

Bei der Unterstützung des IT-Beraters den Kunden durch die einzelnen Stufen zu helfen, treten typische Probleme auf. So können beim Brainstorming Konflikte entstehen, wenn beispielsweise Mitarbeiter aufgrund von Urteilen ihre Gedanken nicht aussprechen. Deshalb ist die Aufgabe des Beraters die einzelnen Meinungen der Mitarbeiter zu fördern und Blockaden zu zerstören.

Der Berater sollte auch Einführen, dass Mitarbeiter über ihre Gefühle sprechen und diese vom Team akzeptiert werden. Dazu gehören ebenfalls Kommunikationsübungen.

Ein weiterer Konflikt kann die Erfahrung aus der Vergangenheit sein. Der IT-Berater muss das Ziel bewusst machen und von der Vergangenheit auf die Zukunft lenken.

Werden diese Schritte berücksichtigt, entsteht ein sich schätzendes, performantes Team mit einer starken, zukunftssicheren Herangehensweise.

9.3 Die agile Organisation

Agilität beschreibt in der IT das Reagieren auf gesellschaftliche Veränderungen, welche durch die, in den vorherigen Kapitel genannten Modelle dargestellt wurden. Die Unternehmenskultur und Werte spielen in diese Agilität mit ein und können durch einen IT-Berater optimiert werden.

9.3.1 Scrum-Teams

In Scrum-Teams werden die Umsetzungen der Kundenaufträge durchgeführt. Hierbei gibt es verschiedene Rollen die unterschiedlich vom IT-Berater unterstützt werden sollten.

Der ScrumMaster übernimmt die Prozessverwaltung und moderiert im Meeting. Hier ist es wichtig sich über die Weitergabe von Informationen, Motivation der Projektbeteiligten und die Fähigkeiten des ScrumMasters zu informieren.

Eine weitere Rolle ist die Product Owner Rolle. Diese beschäftigt sich mit der Optimierung des Wertes eines Teams. Dabei befasst sich dieser mit der gesamten Wertschöpfungskette. Hierbei ist es interessant Informationen über einzelne Arbeitsweisen, Arbeitsverhalten und die Resonanz von internen oder externen Projektbeteiligten zu ermitteln.

Ebenso gilt das Entwicklungsteam zu beobachten und zu analysieren. Jedoch wird meist nicht direkt mit dem Team interagiert sondern über den Scrummaster oder Product Owner. Diese gilt es zu beraten, zu unterstützen und auf fehlendes Fachwissen zu informieren.

9.3.2 Leitungsfunktionen in der Linie

Große Unternehmen benötigen in der Regel zusätzlich zum Scrum-Team eine Matrix Organisation, da es Teilgebiete gibt, welche sich sonst nicht abbilden lassen. Dies können beispielsweise Verantwortungs- und Eskalationsebenendarstellungen sein.

Linienführungskräfte haben dabei die Aufgabe, dass bei den Scrum Meetings alles reibungslos abläuft, bieten sich als Bindeglied zur Personalabteilung und dem Betriebsrat an und führen Einzelgespräche mit den Projektbeteiligten.

Der IT Berater muss sich somit mit verschiedenen Linienführungskräften auseinandersetzen. Besondere Unterstützung bedarf es hierbei dem Topmanagement. Diese müssen erst von der Bedeutung agiler Vorgehen und dessen Vorteile unterrichtet werden um diese im Unternehmen durchzusetzen zu können. Dadurch können Unternehmensvorteile entstehen welche Vorteile im Wettbewerb mit sich ziehen. Um dies erfolgreich durchsetzen zu können müssen Angestellte regelmäßig Feedback geben, ob neue Arbeitsweisen effektiv sind oder nicht.

Ebenso sind Gespräche mit der Personalabteilung wichtig, da diese einen genaueren Bezug auf die Vorteile von agilen Werten benötigen. Andernfalls bleiben diese in dem gewohnten Muster stecken und kaufen wie gewohnt Schulungen ein und werten Feedbackbögen aus.

Auch andere, angrenzende Bereiche sollten eine Einführung in die neue Agilität erhalten. Diese können voreingenommen sein und sollten deshalb ebenfalls in die Vorteile eingeweiht werden.

Zuletzt ist noch zu beachten, dass bei der Komplexität und Größe eines Unternehmens meist nicht ein IT-Berater reicht. Es sollten Teams von Beratern entstehen, welche sich optimal abstimmen.

10 Veränderungsmanagement im Überblick

10.1 Warum fallen uns Veränderungen so schwer?

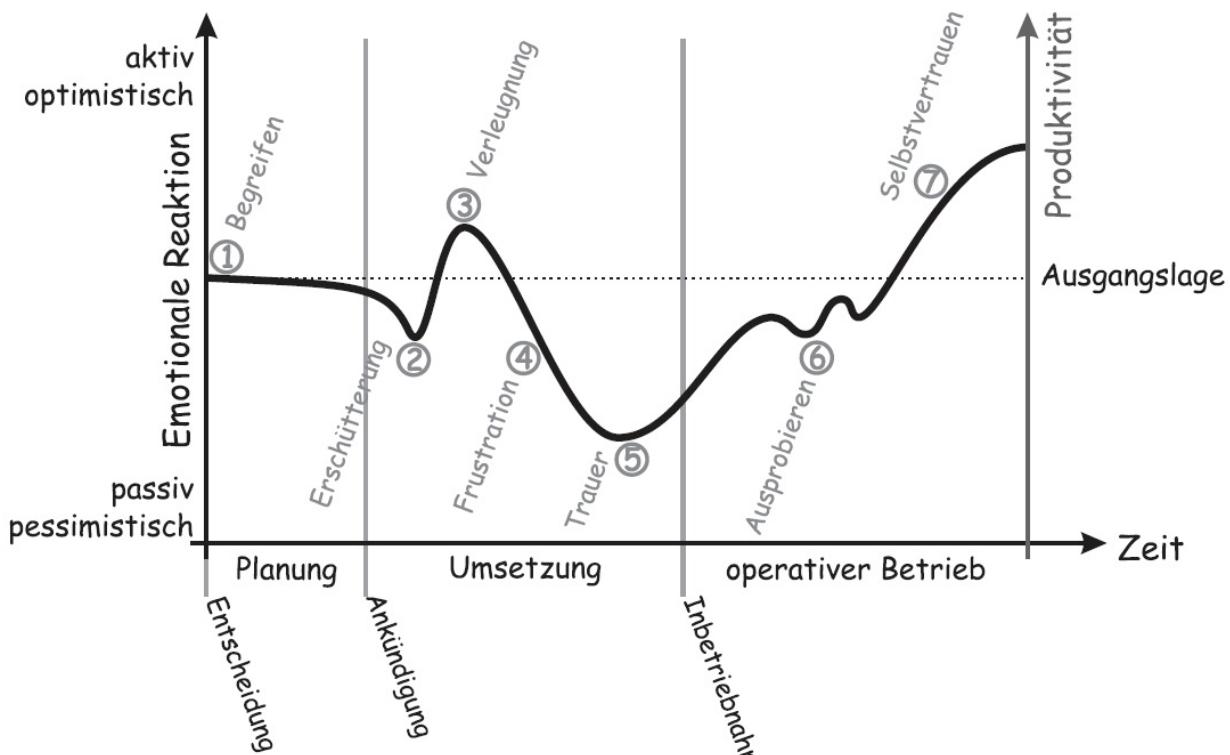
Ein Veränderungsmanagement wird definiert als die kontinuierliche Begleitung von grundsätzlichen Veränderungen. Die notwendigen Maßnahmen im Laufe des Prozesses werden als Projekte durchgeführt.

Jedes IT-Projekt benötigt ein Veränderungsmanagement, da durch neue Geschäftsprozess-Software die Arbeitsweisen der Mitarbeiter oder durch neue Produktionssteuerungen die Produktionsabläufe sich ändern. Beratungsprojekte werden daher fast auch immer von einem Veränderungsmanagement begleitet.

IT-Beratungen sind aufwendig und teuer. Sie lohnen sich aus Kundensicht nur, wenn Themen aus eigener Kraft nicht angegangen werden können oder auf Grund von Rahmenbedingungen nicht in der gewünschten Zeit umsetzbar sind.

10.1.1 Der einzelne Mitarbeiter

Obwohl Weiterentwicklungen unseren ganz normalen Alltag bestimmen, fehlt es oft an einem langfristig erfolgreichen Veränderungsmanagement. Die Folge ist, dass Mitarbeiter teilweise überfordert und frustriert sind, da ihre Erwartungen an die erhofften Verbesserungen nicht erfüllt werden. Die betroffenen Mitarbeiter erleben ein Wechselbad der Gefühle. Dieser Gefühlsverlauf von Mitarbeitern wurde in Form einer Kurve von der Boston Consulting Group veröffentlicht.



(1) Begreifen - Der Mitarbeiter sieht die eigenen Erwartungen bei einer Veränderung als nicht erfüllbar an.

(2) Erschütterung - Der Mitarbeiter ist erschüttert bzw. geschockt.

(3) Verleugnung - Der Mitarbeiter verneint die Veränderung, um sich selbst zu schützen.

(4) Frustration - (5) Trauer - Es setzt ein Verstehensprozess ein und der Mitarbeiter akzeptiert die Lage. Er erkennt, dass er alte Verhaltensmuster aufgeben muss.

(6) Ausprobieren - Der tiefen Trauer entkommt der Mitarbeiter nur durch Ausprobieren neuer Verhaltensweisen. Durch das Aufbauen von neuem Wissen über die Abläufe gewinnt der Mitarbeiter wieder an Sicherheit.

(7) Selbstvertrauen - Der Mitarbeiter gewinnt durch Erfolge wieder Selbstvertrauen.

Der Kurvenverlauf offenbart, dass die Produktivität über die Dauer der Veränderung schwankt und ein dauerhafter Mehrwert sich erst spät einstellt. Der Ablauf in der Veränderungskurve ist unumkehrbar. Die Mitarbeiter reagieren auf diese Unumkehrbarkeit sehr unterschiedlich.

10.1.2 Die Organisation und das Timing

Eine IT-Beratung wird meist auf Managementebene initiiert. Jedoch haben Veränderungsprozesse eine eigene Dynamik, da diese Prozesse Konsequenzen für Mitarbeiter, Linienführungskräfte und weitere Rollen haben. Die Veränderungskurven finden

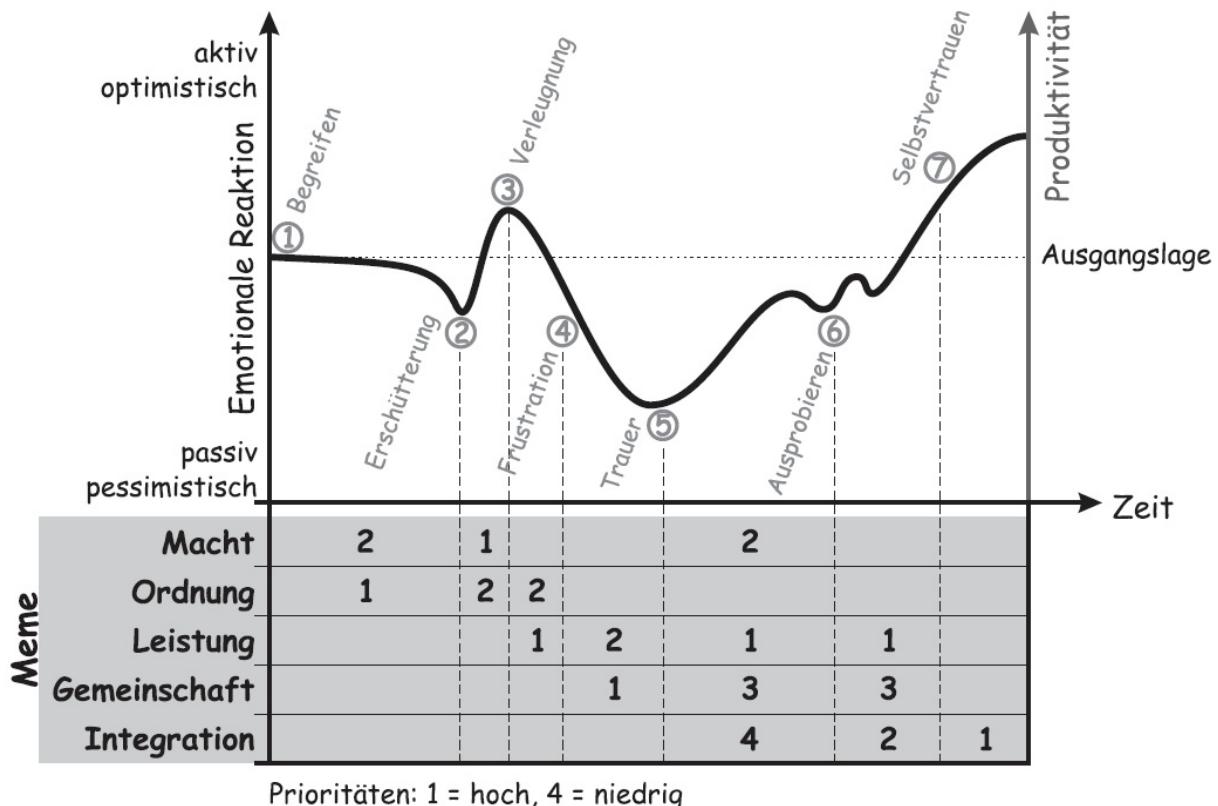
auf verschiedenen Organisationshierarchieebenen phasenversetzt statt. Ein Veränderungsprozess auf einer Ebene kann einen Veränderungsprozess zu einem späteren Zeitpunkt auf einer anderen Ebene initiiieren, um eine Anpassung des betroffenen Aspekts umzusetzen. Die Phasenverschiebung kann zu Problemen der internen Kommunikation eines Unternehmens führen, wenn z.B. das Management die Veränderung abgeschlossen hat und die Mitarbeiter sich jedoch noch in der Trauerphase befinden. Die Kommunikation ist dann oft nicht empfängerorientiert und verfehlt somit sein Ziel.

10.1.3 Was kann Veränderungsmanagement leisten?

Ein gutes Veränderungsmanagement setzt Aktionen erfolgreich um. Bei einer Veränderung durchleben dabei die Mitarbeiter zwar genau die gleichen Phasen des Gefühlverlaufs, jedoch kann die Dauer der Phasen verkürzt und die emotionalen Reaktionen verringert werden.

10.1.4 Veränderungen unterstützen

Um als Berater den Veränderungsprozess zu unterstützen, muss er die aktuelle Phase einer Veränderung bestimmen können und sein Verhalten so anpassen, dass es die Menschen bei der Veränderung bestmöglich unterstützt. Zusätzlich besitzt er die Aufgabe den Informationsfluss zwischen Führungskraft und Mitarbeitern sicherzustellen. Die nachfolgende Abbildung zeigt die Veränderungskurve und den notwendigen Memen.



Vom Begreifen zur Erschütterung - Die Ordnung muss wie vor Beginn der Veränderung beibehalten werden. Die Führungskräfte müssen ihren Mitarbeitern aktiv zuhören und die Beweggründe für die Veränderung erläutern.

Von der Erschütterung zur Verleugnung - Die Mitarbeiter lehnen auf Grund von Angst die Veränderung ab. Die Führungskräfte müssen fest an der Veränderung festhalten und keinen Zweifel auftreten lassen.

Von der Verleugnung zur Frustation - Die Mitarbeiter merken, dass sie ihr Verhalten ändern müssen. Die Führungskräfte müssen sie dabei so gut es geht unterstützen.

Von der Frustation zur Trauer - Führungskräfte können zugeben, dass sie keine detaillierte Lösung haben, und können dies nutzen, um eine engere Kommunikation mit den Mitarbeitern aufzubauen und sie zum Durchhalten zu ermutigen.

Von der Trauer zum Ausprobieren - Die Mitarbeiter verstehen, welches Verhalten ziieldienlich ist. Die Führungskräfte stehen den Mitarbeitern mehr beratend und reflektierend zur Seite. Spitzenleistungen werden honoriert und kommuniziert.

10.2 Veränderungsmanagementmodelle

Analyse des Problems

Bei anstehenden Veränderungen ist es zunächst essentiell wichtig konkrete Informationen zu sammeln und die Punkte zu klären, die zu beachten sind. Es ist immer wieder zu erkennen, dass zu vorschnell über Lösungen nachgedacht wird. Dies führt dazu das Lieblingslösungen einer Person (z.B. zu einer Organisationsstruktur oder bestimmte agile Techniken) gewählt werden, die sich am Ende als unpassend erweisen. Weiterhin folgt aus dieser Lösungsprägung, dass nur noch Informationen betrachtet werden, die zu dem Bild einzelner Personen passen und wichtige Informationen übersehen werden. Ein weiteres Problem ist, dass die Fähigkeiten der betroffenen Personen oder der Einsatz von konkreten Lösungen als viel zu optimistisch angesehen werden. Es kommt folglich zu Verzögerungen bis hin zum Scheitern der Veränderung.

Frühe Kommunikation

Anstehende Veränderungen sollten bereits früh kommuniziert werden, damit sich durch die umfangreiche Kommunikation kein unvollständiges Bild ergibt. Ein vollständiges Bild ist wichtig, damit die Konzeption keine Lücken aufweist und alle wichtigen Stakeholder abgeholt werden. Durch den intensiven Austausch mit relevanten Stakeholdern können Konflikte rechtzeitig aufgedeckt und konstruktiv gelöst werden.

Faule Kompromisse

Faule Kompromisse können zu einem Scheitern des Veränderungsprozesses führen. Häufige Symptome sind, das Bestehen lassen von alten Abläufen oder Rollen oder das Hinausschieben von Entscheidungen, die jedoch jetzt benötigt werden.

Ausprobieren

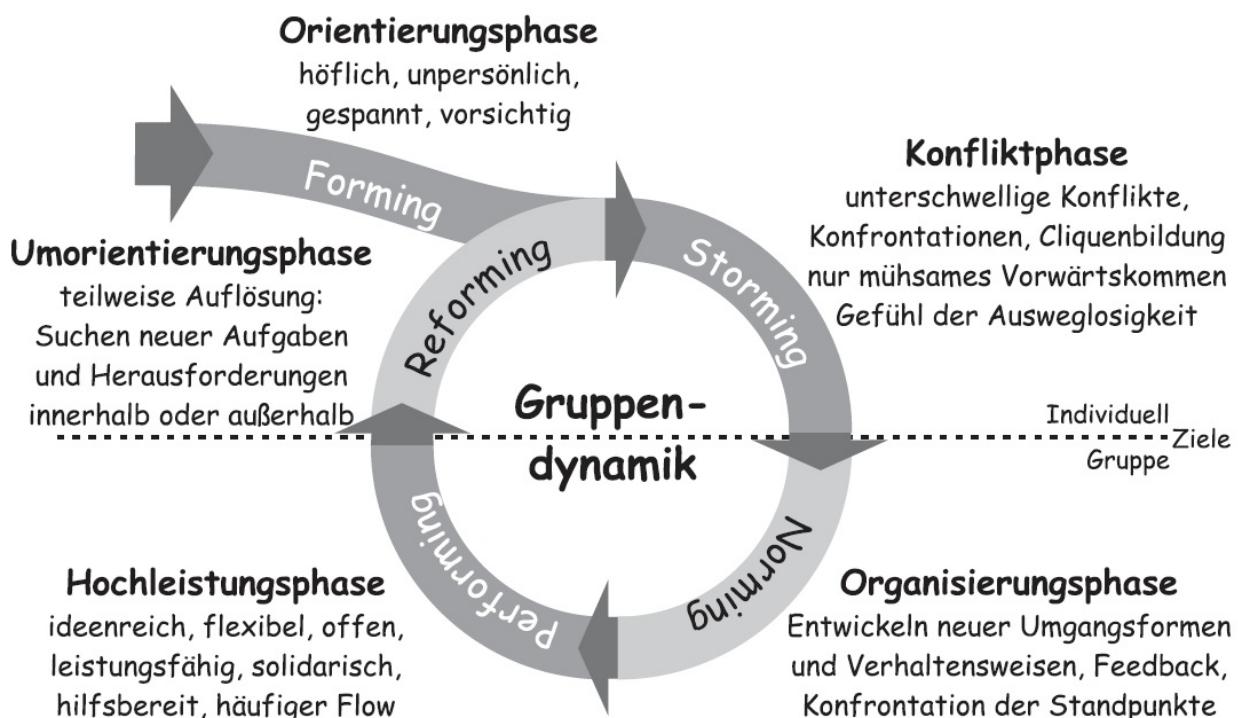
Es wird häufig vergessen, dass Veränderungen viel mit Ausprobieren zu tun haben. Die Folge ist, dass der Druck entsteht, alle Entscheidungen zu einem frühen Zeitpunkt treffen zu müssen. Da jedoch im frühen Stadium die Unsicherheit sehr hoch ist, werden Entscheidungen lieber gar nicht getroffen.

10.3 Dynamik in Gruppen

Die Teamdynamik ist bei Veränderungen ein mächtiger Aspekt. Sie kann Veränderungsprozesse unterstützen, aber auch blockieren. Ein Verständnis über diese Dynamik ist daher für IT-Berater unbedingt notwendig.

10.3.1 Die Teamuhr

Die nachfolgende Abbildung zeigt die Phasen, die jedes Team zyklisch durchläuft.



Forming: Diese Orientierungsphase ist geprägt von Unsicherheit und distanzierten Verhalten. Die Teammitglieder versuchen sich aneinander zu gewöhnen und verhalten sich daher meist vorsichtig.

Storming: In dieser Konfliktphase kommt es vermehrt zu Machtkämpfen. Es bilden sich Cliques und es entstehen Konflikte. Das Team kommt nur mühsam voran.

Norming: In dieser Organisierungsphase entwickeln sich gruppeneigene Umgangsformen und Verhaltsweisen. Die Teammitglieder intensivieren ihre Zusammenarbeit. Jedes Mitglied hat seine eigenen Aufgaben und wird innerhalb des Teams akzeptiert.

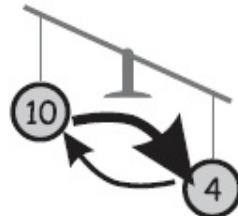
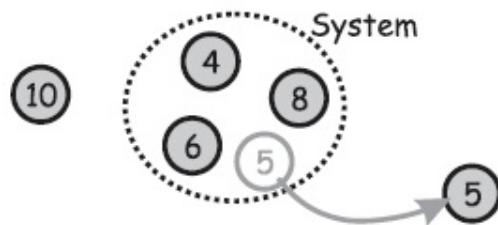
Performing: Die Gruppe ist äußerst leistungsfähig. Die Teammitglieder sind offen, ideenreich, flexibel und offen für Neues.

Reforming: Einige Teammitglieder wagen den Aufbruch zu neuen Ufern. Dadurch haben einzelne Gruppenmitglieder die Chance nachzurücken. Die Gruppenstruktur ändert sich deutlich, sodass es zu Unruhe und Unzufriedenheit kommt.

10.3.2 Systemische Betrachtungen und Grundregeln

Bei einer systemischen Organisationsberatung wird das Zusammenspiel mehrerer Elemente und im Kontext des Systems betrachtet, um eine Lösung für ein Problem zu finden. Die Lösung kommt dabei aus dem System, da die betroffenen Personen die Experten des Problems sind.

In einem System gibt es drei Ebenen, um die eigene Position innerhalb einer Gruppe zu bewerten. Diese Ebenen stehen in Wechselwirkung, da z.B. ein Problem auf einer Ebene Wirkung auf andere Ebenen haben kann.



Bindung: Auf dieser Ebene geht es um die Frage, ob ich neben der formalen Zuordnung zu einer Gruppe oder eines Projektteams, mich auch tatsächlich als Mitglied fühle und von den Kollegen im Team akzeptiert werde.

Ordnung: Die nächste Frage nach der Klärung der Zugehörigkeit ist, ob ich mich selbst im Team am richtigen Platz sehe. Fühle ich mich in meiner Rolle im Team wohl?

Ausgleich: In dieser Ebene geht es um die richtige Balance zwischen Geben und Nehmen. Bin ich der einzige der Überstunden macht und bekomme ich etwas von den Kollegen zurück?

10.3.3 Systemische Ordnung in Gruppen

Als IT-Berater für ein bestehendes Team gilt es die systemische Ordnung in der Gruppe zu verstehen. Ein erster Ausgangspunkt ist dabei die Eintrittsreihenfolge. Weitere Fragen, wie z.B. "Wer bildet den Kern?", "Wer wird akzeptiert und von anderen Gruppenmitgliedern zu Entscheidungen hinzugezogen?", geben ein detailliertes Verständnis über diese Ordnung.

11 Veränderungsmanagement konkret

Zwei Dinge sind mit Veränderungen immer einhergehend: Risiken die bei der Veränderung eingegangen werden müssen und Widerstand, der durch das Verändern hervorgerufen wird. Diese beiden Themen werden in diesem Kapitel behandelt.

Jede Veränderung ist ein Projekt

Da Veränderungen im Kontext von Organisationen durchgeführt werden, sind diese meist mit einer gewissen Komplexität behaftet. Des Weiteren sind verschiedenste Personen an Veränderungen beteiligt - es ergibt also Sinn, Veränderungen als Projekte zu betrachten.

Häufig wird der Fehler gemacht das Hauptaugenmerk auf den Inhalt der Veränderung zu legen. Also zu betrachten, welche Änderungen notwendig sind, welche Techniken hierbei evtl. eingesetzt werden müssen und wie Mitarbeiter geschult werden müssen. Ein vernachlässigter Aspekt sind hierbei die Mitarbeiter selbst. Es ist überlegenswert einen Plan aufzustellen, wann welche Mitarbeiter über die anstehenden Veränderungen informiert werden sollen und in welchem Rahmen diese Informationsverteilung stattfinden soll. Bei größeren Veränderungen hat es sich bewährt einen Veränderungsmanager einzusetzen, der für den notwendigen Informationsfluss sorgt.

Widerstand

Bei jeder Veränderung trifft man auf Widerstand. Oft wird Widerstand von Führungskräften fehlinterpretiert als ein tatsächliches "dagegenstellen" der entsprechenden Mitarbeiter. Häufiger ist jedoch anzutreffen, dass diese Mitarbeiter nur nicht wissen, wie sie sich bei den Veränderungen einbringen können. In der Akzeptanzmatrix werden vier Reaktionstypen gezeigt:

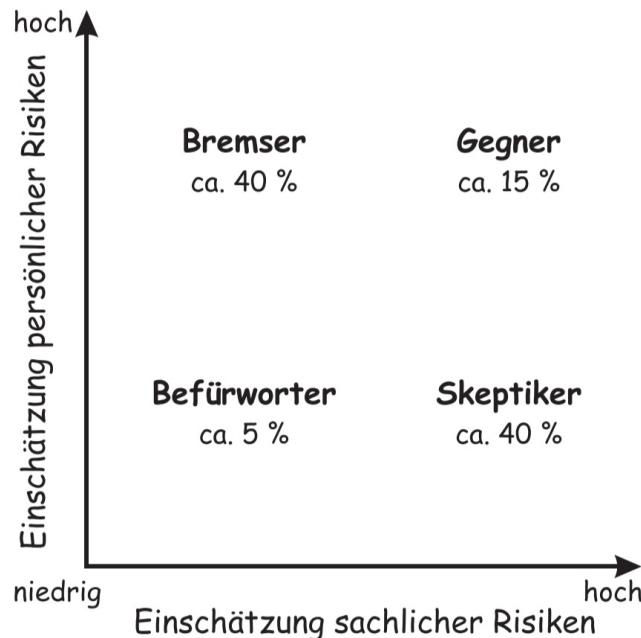


Abbildung 1: Reaktionstypen (S. 187)

Befürworter sind Personen die für jegliche Veränderungen offen sind. Diese Personengruppe hat erkannt, dass Veränderungen notwendig sind und können bei der Durchführung der Veränderungen als Multiplikatoren dienen. (Vgl. S 187)

Skeptiker sind Personen, die (persönliche) Nachteile durch die Veränderungen befürchtet. Diesen müssen fehlende Informationen gereicht werden. Sind bestimmte Informationen nicht vorhanden, können diese gemeinsam mit den Skeptikern erarbeitet werden. Hierbei wird auch eine gemeinsame Vertrauensbasis geschaffen. Einwände der Skeptiker sollten genauer betrachtet werden. In ihnen liegt ein großes Potential die Veränderungen sinnvoll zu verbessern bzw. anzupassen. Durch die Berücksichtigung der Einwände, können aus Skeptikern sogar Befürworter werden. (Vgl. S. 188)

Bremser sind Personen die nicht ausreichend Informationen über die geplanten Veränderungen haben und ebenfalls Angst vor persönlichen Risiken haben. Diese Personengruppe muss mit weiteren Informationen versorgt werden - zu klären ist hier die Frage, welche Informationen genau fehlen. Durch das Beschaffen der Informationen oder das gemeinsame Erarbeiten dieser, können weitere Verfeinerungen an den Veränderungen vorgenommen werden und evtl. sogar aus Bremsern Befürworter gemacht werden. (Vgl. S. 187)

Gegner sind Personen die ausdauernd gegen die Veränderungen sind. Auch bei weiterem Investieren von Energie zur Überzeugung ist keine Einsicht vorhanden. Dennoch sind Einwände der Gegner eine wichtige Informationsquelle. einige der Einwände können durchaus berechtigt sein und verdienen eine genauere Betrachtung. Es muss abgewogen werden, ob ein Einwand ignoriert werden kann oder weiter betrachtet werden muss. Die

Wahrscheinlichkeit, aus dieser Personengruppe Befürworter zu machen ist jedoch gering. Es sollten keine großartigen Energien in die Überzeugungsarbeit gesteckt werden. (Vgl. S. 188)

Stabilität und Veränderungen

Veränderungen sollten keineswegs so häufig wie möglich durchgeführt werden. Damit Mitarbeiter produktiv arbeiten können, muss ein stabiler Zustand erreicht werden. Eines der wichtigsten Ziele von Veränderung ist es, einen besseren stabilen Zustand so schnell wie möglich zu erreichen. Ist dieser erreicht, wird die Funktionsoptimierung der Prozesse angestrebt. Veränderungen sind wieder notwendig, wenn sich die Rahmenbedingungen (also die Umwelt) des Systems verändert haben - wenn sich also beispielsweise der Kundenkreis bzw. das Kundenalter ändert.

Ein System kann verschiedene Systemverhalten aufweisen:

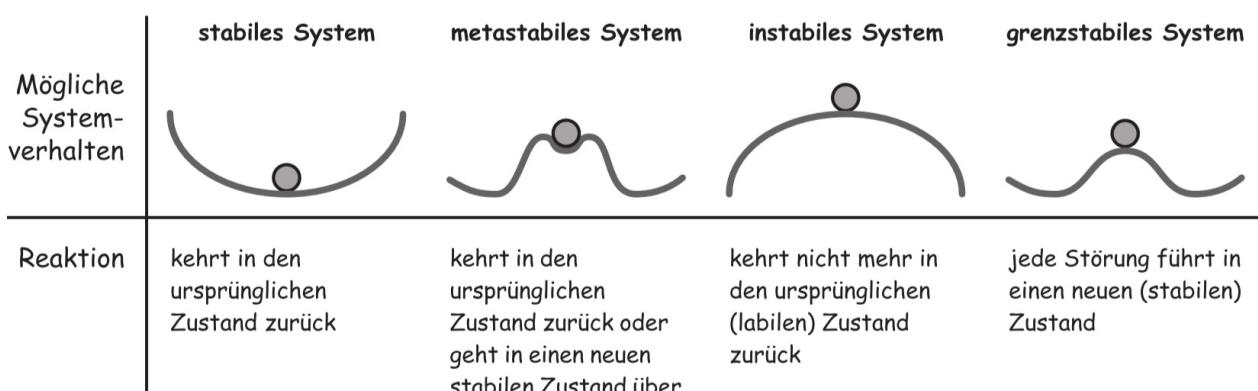


Abbildung 2: Systemverhalten (S. 190)

Stabiles Systemverhalten: jegliche Änderungen (extern, intern) führen zum gleichen stabilen Systemzustand.

Beispiel: Der Projektleiter ist im Urlaub. Während seiner Abwesenheit passt sich das System den neuen Gegebenheiten an. Nach seinem Urlaub haben alle Prozesse die gleiche Gültigkeit. (Vgl. S. 189)

Metastabiles Systemverhalten: das System bleibt bei Änderungen entweder im gleichen stabilen Zustand, oder wechselt in einen anderen, neuen stabilen Zustand.

Beispiel: Der Projektleiter wird ausgetauscht. Der neue Projektleiter hat die Möglichkeit entweder alle bestehenden Prozesse beizubehalten, oder neue einzuführen bzw. bestehende zu ändern. (Vgl. S. 190)

Instabiles Systemverhalten: bei Änderungen oder Störeinflüssen wird das gesamte System zerstört.

Beispiel: In wirtschaftlich schwierigen Zeiten sind die Mitarbeiter stark verunsichert. Ein Mitarbeiter kündigt, woraufhin weitere wichtige Mitarbeiter gehen. (Vgl. S. 190)

Grenzstabiles Systemverhalten: jede Störung führt zwingend zu einer Anpassung des Systems, welche in einem stabilen Zustand endet.

Beispiel: Anpassung von Anforderungen innerhalb eines laufenden Projektes. (Vgl. S. 190)

Störungen und Änderungen der Rahmenbedingungen können also unterschiedliche Auswirkungen auf das System haben. die Wirkung verschiedener Störungen kann jedoch nicht vorausgesagt werden.

Erfolgsschlüssel Veränderungsfähigkeit

Es gibt vier Arten von Veränderungsmanagement, aufgetrennt auf Wirkungszeitpunkt und Fokus (Vgl. S. 191):

Strategisches Veränderungsmanagement: Langfristige strukturelle Vorbereitung auf die Veränderungen

Projektveränderungsmanagement: Projektmanagement von Veränderungsprozessen

Kulturelles Veränderungsmanagement: Veränderungskompetenzen der Mitarbeiter werden erhöht. Die Mitarbeiter stellen sich auf die Veränderungen ein.

Führungsveränderungsmanagement: Mitarbeiter und Führungskräfte werden bei dem Veränderungsprozess geführt und begleitet.

Die letzten beiden Arten von Veränderungsmanagement haben die Vorbereitung der Personen eines Unternehmens im Fokus, wohingegen die ersten beiden die Prozesse genauer betrachten.

Die Ausrichtung der Personen auf Veränderungen sind ein wichtigerer und häufig weniger betrachteter Aspekt bei der Einführung von Veränderungen.

"Viele Organisationen sind aber auf ein kulturelles und Führungsveränderungsmanagement gar nicht ausgerichtet (...)." (S. 191)

Dies stellt Veränderungen vor eine Hürde, kann aber auch die Chance sein, eine neue Veränderungskultur im Unternehmen einzuführen.

Störungen und Identität von Mitarbeitern

Veränderungen können bei zu großem Einwirken in die Abläufe als Störend empfunden werden. Jede Person / Gruppe hat eine individuelle Störungsschwelle. Ein Ziel von Veränderungen ist stets unterhalb dieser Störungsschwelle zu agieren. (Vgl. S. 192,193)

Die Störungsschwelle kann schlagartig durch zu große Veränderungen überschritten werden, oder durch viele kleine Veränderungen, die in Summe über die Störungsschwelle gelangen. Letztere sind wesentlich schwerer nachzuvollziehen, da die einzelnen Veränderungen als solche nicht als Störend empfunden werden.

Als störend werden Veränderungen empfunden, die den Mitarbeiter betreffen - ihn also subjektiv nach der Veränderung schlechter dastehen lassen, bezogen auf Gehalt, Arbeitsinhalte, Arbeitszeiten etc. (Vgl. S. 194)

"Nach Frederick S. Perls (1893 – 1970) wird unsere Identität von fünf Säulen getragen (...):" (S. 195)

- Leiblichkeit (Gesundheit)
- Soziales Netz und soziale Umwelt (Freunde, Familie)
- Arbeit, Leistung, Freizeit (Beruf, Hobbys)
- Materielle Sicherheit (Besitz, Vermögen)
- Werte (Denkweise)

(Vgl. S. 195)

Die Identität wird von allen diesen Säulen gemeinsam getragen. Wird eine Säule geschwächt, so gerät das gesamte Konstrukt der Identität ins wanken. Je stärker die Auswirkungen von Veränderungen auf eine oder mehrere dieser Säulen ist, desto stärker wird auch der Widerstand der einzelnen Person gegen diese Veränderungen sein.

12 Werkzeuge des Veränderungsmanagers

Sounding Board

Bei dem Sounding Board wird durch die Mitarbeiter, welche von dem Veränderungsprozess betroffen sind, Feedback generiert und den Führungskräften bzw. dem Projektmanager der Veränderungsprozess konsolidiert und anonym mitgeteilt. Die Führungskraft kann daraufhin entsprechende Maßnahmen veranlassen.

Ablauf

Zunächst teilen die Mitarbeiter den Sounding Board Mitgliedern ihre Meinungen und Gefühle mit. Die Sounding Board Mitglieder sind eine Auswahl von Mitarbeitern, die den Veränderungsprozess aktiv mitgestalten wollen und von den Kollegen respektiert werden. Die Sounding Board Mitglieder aggregieren die Meinungen zu Feedbackthemen und teilen diese wiederum dem Veränderungsmanager mit. Dieser ist das Bindeglied zwischen dem Projektleiter und den Mitarbeitern im Sounding Board. Es empfiehlt sich diese Stelle extern zu besetzen, sodass die Feedbackthemen deutlich kritischer durch die Mitglieder des Sound Boards dargestellt werden können. Der Veränderungsmanager kommuniziert daraufhin mit der Führungskraft und stellt die Themen anonymisiert vor. Die Führungskraft bzw. der Projektleiter des Veränderungsprojekts kann nun Verbesserungsmaßnahmen ergreifen.

Appreciative Inquiry - Interviews führen

Appreciative Inquiry bezeichnet eine wertschätzende Befragung (Interview) der Mitarbeiter einer Organisation über alle Hierarchiestufen hinweg. Der Interviewer versucht eine positive Interviewatmosphäre zu schaffen und bringt die Wertschätzung jedes einzelnen Mitarbeiters zum Ausdruck.

Der Mitarbeiter wird befragt, welche Verbesserungspotentiale er innerhalb der Organisation sieht, wie z.B. die Qualitätsstandards nachhaltig erreicht werden können. Außerdem wird er zu seiner Meinung und Gefühlen innerhalb der Organisation gefragt und was der jeweilige Mitarbeiter an seiner Arbeit schätzt. Es ist darauf zu achten, dass der Mitarbeiter sich öffnen kann und er darf in seinem Redefluss nicht unterbrochen werden. Gelegentlich muss man sich von dem starren Fragenkatalog entfernen.

Es empfiehlt sich die Auswertung der Interviews mit einem Kollegen zu machen, welcher auch Interviews geführt hat. Zusammen wird die Quintessenz der Interviews erarbeitet, sodass ein Bild der Meinungen im Unternehmen gebildet wird. Dem Auftraggeber werden nun die verdichteten Informationen anonymisiert mitgeteilt.

Veränderungsmetaphern

Bei der Kommunikation mit den Mitarbeitern sind manche Wörter des Veränderungsmanagements negativ konnotiert. Es empfiehlt sich generell eine positiv wirkende Sprache zu verwenden und das Vokabular der Zielgruppe anzupassen. Zum Beispiel ist "Reorganisation" durch die Metapher "Änderung der Rollenmodelle" zu ersetzen. Oder es kann die Ingenieurmetapher "Refactoring" für "Fehlerbehebung" verwendet werden.

Lego und CO. Neue Abläufe simulieren

Ein Veränderungsprozess wird anhand einer Simulation mit dem Kunden durchlaufen. Die Simulation ist dabei eine Konfliktsituation, die mit spielerischen Mitteln erfolgt (z.B. der Bau einer Murmelsprungschanze). Die Rahmenbedingungen, z.B. dass die Konstruktion verteilt geschieht, simuliert das Projektumfeld. Im Anschluss erfolgt eine Nachbesprechung und eine Eigenreflexion der Mitarbeiter, bei der das Erlernte in der Simulation auf das reale Arbeitsgeschehen übertragen wird.

13 Veränderungen und das Troja-Prinzip

Veränderung braucht Redundanz

Bei einem Veränderungsprozess sinkt zunächst die Produktivität der einzelnen Mitarbeiter, steigt daraufhin wieder an, bis sie das alte Level übertroffen hat. Die fehlende Produktivität muss ausgeglichen werden z.B. durch Mehrarbeit des Mitarbeiters oder der Prozess fällt in eine Zeitspanne, in der Leerlaufzeiten vorzufinden sind (Saisongeschäft) und der Mitarbeiter mehr Freiräume hat. Um dies zu umgehen, kann der gesamte Veränderungsprozess in kleinere Prozessschritte untergliedert werden, sodass die Mangelproduktivität deutlich geringer ausfällt.

Insgesamt benötigt der Mitarbeiter Freiräume, damit er die Veränderungen einführen kann. Dies kann auch durch Redundanzen im Betriebsablauf erwirkt werden.

Neue Balance finden

Für das erfolgreiche Durchlaufen des Veränderungsprozesses benötigt es eine gesteigerte Flexibilität der Mitarbeiter. Zunächst muss das Team eine Ist-Analyse durchlaufen, bei der das Team in ein Gruppenfeld eingetragen wird, wie in Abbildung XXX beispielhaft dargestellt ist. Weiter wird durch Personalentwicklung das Team flexibler gestaltet, siehe Abbildung XXX. Dabei können die folgenden vier Barrieren auftauchen:

- "nicht zu erkennen, was gesehen wird"
- "nicht zu sagen, was gedacht wird"
- "nicht zu tun, was gesagt wird"
- "nicht zu sehen, was getan wird".

Der Berater hat die Aufgabe diese Probleme des Teams zu erkennen und sie durch Diagloge zu durchbrechen.

Sofern Standards und Frameworks eingeführt werden, ist dies zunächst ohne eine Individualisierung der Frameworks für das Unternehmen zu geschehen. Daraufhin werden die ersten Erfahrungen mit dem Framework, z.B. Scrum, gesammelt und folgend wird das Framework unterspezifisch angepasst.

14 Fallbeispiel

In der IT-Beratung kommt es häufig vor, dass Kunden nicht mit der eigentlichen Problemstellung zum Berater kommen, sondern bereits mit Lösungsvorstellungen. Ob diese Lösungsvorstellungen sinnvoll sind und das Problem tatsächlich lösen kann von dem Berater erst ermittelt werden, wenn dieser das eigentliche Problem kennt. In diesem Kapitel wird anhand eines Fallbeispiels erläutert, wie als IT-Berater vorgegangen werden kann, um das Problem zu ermitteln und eine entsprechende Lösung mit dem Kunden zu erarbeiten.

Im Fallbeispiel geht es um ein Unternehmen aus der Finanz-Branche, in dem Scrum als Vorgehensmodell eingeführt werden soll. Zu Beginn werden einige Fragen genannt, die dem Kunden zu stellen sind. Durch diese erlangt man an wertvolle Zusatzinformationen.

Ziele:

- **Was soll erreicht werden?**

Die Frage nach dem Ziel wird häufig mit indirekten Zielen beantwortet. Bezogen auf das Fallbeispiel: "Wir möchten UML als Standard in der Entwicklung einführen" statt "Wir müssen unsere Kommunikation verbessern."

- **Welcher Nutzen soll erbracht werden?**

Antworten beziehen sich häufig auf Durchlaufzeiten oder ähnliches. Um im späteren Verlauf feststellen zu können, ob der Nutzen auch tatsächlich erbracht werden konnte, muss eine Ist-Zustandsaufnahme gemacht werden.

- **Wer sind die Nutznießer?**

Die Frage nach den Personen, die von den Veränderungen profitieren sollen. Im Beispiel profitieren sowohl die Entwickler als auch das Management und die Kunden von der Einführung agiler Entwicklungsmethoden. Die Entwicklungszeiten werden durch verbesserte Kommunikation minimiert und damit die Kosten gesenkt.

Ergebnisse:

- **Was soll genau herauskommen?**

Die Frage nach der Form durch die der Nutzen erbracht werden soll.

"Beispiele sind regelmäßige interne monatliche Ergebnisreviews der aktuellen Software mit neuen oder angepassten Features durch die Qualitätssicherung, Produktmanager und andere Stakeholder." (S. 229)

- **Was soll konkret hinterher da sein?**

Ein Beispiel technischer Natur wäre, das ausrollen von Releases innerhalb von Drei-Monats-Zyklen.

Eine Antwort organisatorischer Natur könnte die Ausrichtung und Gruppierung der Entwicklung in Feature-Teams sein.

Vorhandenes:

- **Was haben Sie bereits?**

Diese Frage deckt häufig direkt offene Punkte bei Kunden auf. Es entpuppen sich Unklarheiten.

- **Woher haben Sie es?**

Falls es doch bereits Vorhandenes gibt, stellt sich die Frage nach dem Woher. Ist es in Eigenleistung entstanden? Wurde es ebenfalls durch ein Beratungsunternehmen eingeführt? Liegt die Kompetenz im Hause?

Benötigtes:

- **Was brauchen Sie noch?**

Was ist noch notwendig um mit dem Projekt zu beginnen? Welche Eingangssartefakte sind noch nicht vorhanden?

Rahmen:

- **Welche festen Randbedingungen sind einzuhalten?**

In welchem Budget-Rahmen bewegt sich die Beratung? Welche Mitarbeiter werden beim Veränderungsprozess fest eingebunden? Bleibt es bei der Zahl der Mitarbeiter bzw. sind weitere Einstellungen geplant?

- **Welche Hindernisse sehen Sie?**

Die Hindernisse beziehen sich hierbei direkt auf die einzuführenden Veränderungen. Zum Beispiel, dass die abzustellenden Mitarbeiter eigentlich auch in anderen Projekten mitarbeiten müssen, oder das direkte Probleme in der Produktentwicklung Vorrang gegenüber dem Veränderungsmanagement haben.

- **Welche Risiken befürchten Sie?**

Das Größte Risiko sieht die Geschäftsführung in dem Scheitern des Projektes als Ganzes. Das würde hohe Kosten und geringen bis gar keinen Nutzen bedeuten.

Aktivitäten:

- **Welche groben Schritte wollen wir dazu gemeinsam gehen?** An dieser Stelle werden erste Vorschläge durch die Beratung erbracht. Im Beispiel wird das durchführen erster Scrum-Workshops für die beteiligten Personen genannt. Außerdem wird das Vorbereiten der Produktmanager auf die zukünftige Rolle "Product Owner" und das Vorbereiten der Projektleiter auf die zukünftige Rolle "Scrum Master" genannt.

- **Welche Abhängigkeiten gibt es?**

Im Beispiel wird nur eine Abhängigkeit genannt: Das durchführen aller Workshops vor der eigentlichen Einführung von Scrum. Eine gemeinsame Wissensbasis ist für den Erfolg des Projektes unabdingbar.

Grundlage des Beratungskonzepts

Das Beratungskonzept wird zweistufig aufgeführt. Es wird zwischen der Veränderungsebene und der operativen Ebene getrennt. In der Veränderungsebene werden die notwenigen Maßnahmen erarbeitet und getroffen, um das Unternehmen auf die Veränderungen auszurichten (Management-Ebene). In der operativen Ebene wird die inhaltliche Korrektheit der Durchführung sichergestellt.

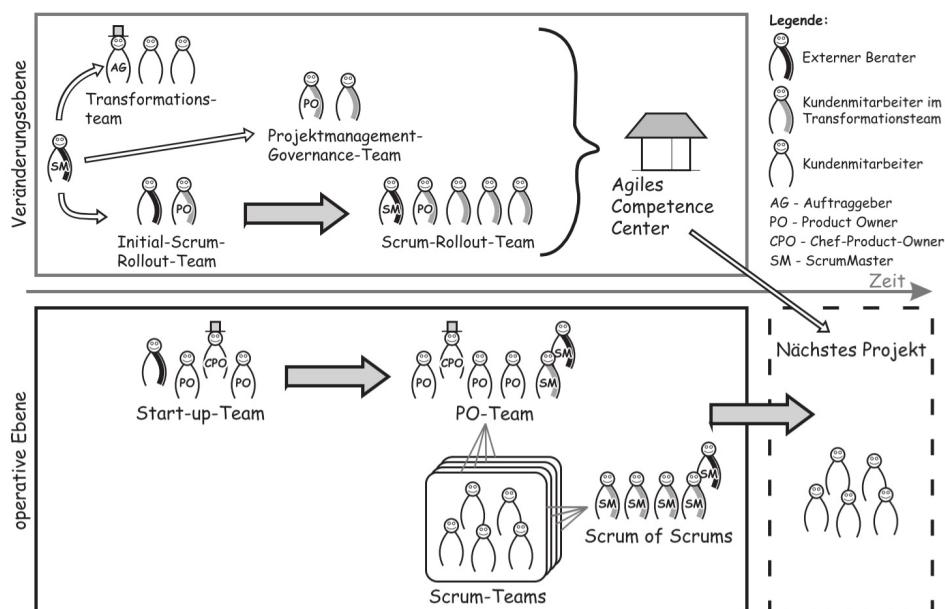


Abbildung 1: Beratungskonzept (S. 235)

Die Gruppen bestehen aus folgenden Personen:

Transformationsteam: Auftraggeber und Führungskräfte

Projektmanagement-Governance-Team: Stellvertreter aus den verschiedenen Bereichen des Unternehmens.

Scrum-Rollout-Team: "Setzt den Veränderungsprozess als eigenständiges Projekt um." (S. 236). Das Initial-Scrum-Rollout-Team leistet hier Vorarbeit (z.B. erstes Backlog füllen).

Start-up-Team: "Bereitet das konkrete Projekt vor und führt die ersten Schritte durch. Dabei entwickelt es die Vision für das Projektergebnis und füllt das Product Backlog." (S. 236)

Scrum-Teams: Hier "erfolgt die Umsetzung des konkreten Entwicklungsprojekts, das aufgrund der Größe in unserem Beispiel als Scrum of Scrums und mit einem Product-Owner-Team realisiert wird."

Bei der Durchführung des Veränderungsprojektes haben die Teams aus der Veränderungsebene den gleichen Iterationszyklus, wie die Teams auf der operativen Ebene. Hierdurch können nach jedem Sprint die Ergebnisse aus den beiden Bereichen direkt aufeinander einwirken. In der operativen Ebene werden Produkte nach dem Vorgehensmodell Scrum entwickelt. Die Anpassung von Scrum an das Unternehmen und das definieren der notwendigen Prozesse findet in der Veränderungsebene statt. Nach jeder Iteration erhält die Veränderungsebene Feedback durch die operative Ebene. (Vgl. S. 237)

Das konkrete Umsetzungskonzept & Probleme lösen

Die Umsetzung erfolgt durch zwei Berater. Jeweils einer pro Ebene:

"Jeweils sowohl auf der Veränderungs- als auch auf der operativen Ebene hat einer der beiden Berater primär das Coaching der ScrumMaster und des Teams zur Aufgabe und der andere das Coaching der Product Owner" (S. 239)

Durch die Berater werden ebenfalls die initialen Workshops und Seminare durchgeführt. Eine weitere Aufgabe ist es aus dem Scrum-Rollout-Team ein eigenes "agiles Competence Center" zu entwickeln. (Vgl. 239)

Veränderungen sind immer mit Problemen behaftet. Eine Aufgabe der Berater ist das Gespür für die Probleme zu entwickeln und diese Probleme entsprechend anzugehen. So wird als Beispiel genannt, dass eines der Scrum-Teams die Daily-Scrums für überflüssig hält. Die Aufgabe des Beraters ist es, das eigentliche Problem mit dem das Team zu kämpfen hat zu finden und die Grundsäulen des Vorgehensmodells (eine Grundsäule von Scrum sind die Daily-Scrums) zu verteidigen. Um das Team in die richtige Richtung zu lenken und das Daily-Scrum als Instrument nicht zu verlieren kann eine Anpassung des Daily-Scrums vorgenommen werden, sodass dieses nicht mehr als überflüssig aufgefasst wird. (Vgl. S. 241, 242)

Soft Skills für IT-Führungskräfte und Projektleiter

Zusammenfassung des Buches:

Titel: Soft Skills für IT-Führungskräfte und Projektleiter - Softwareentwickler führen und coachen, Hochleistungsteams aufbauen

Verfasser: Uwe Vigenschow, Björn Schneider, Ines Meyrose

Verlag: dpunkt.verlag

Jahr: 2016

ISBN: 978-3-86490-395-3, 978-3-96088-001-1 (eBook)

Zusammenfassung von: Philipp Viertel, Andre Kaleja

2 Kommunikation

Ein wichtiges Mittel der Kommunikation stellt die Körpersprache da. In den meisten Fällen ist sie unverfälscht und echt, und verleiht dabei dem Gesagten oft die meiste Bedeutung. Sich mit diesem Phänomen näher zu beschäftigen, ermöglicht es zwei Aspekte besser zu verstehen:

- Die Wirkung durch die Körpersprache auf andere Menschen
- Die Körpersprache des anderen wahrnehmen und zu verstehen wie es bei ihm oder ihr ankommt

Die Körpersprache, also Körpersignale, Mimik und Tonfall, kann gezielt beeinflusst werden, um ein gewünschtes Ergebnis zu erzielen.

Die jeweilige Situation beeinflusst eine gewisse Körperhaltung auch stark. Im Zusammenspiel mit unterschiedlicher Mimik, kann z.B. das Verschränken der Arme eine abweisende Haltung darstellen oder aber eine amüsierte, spitzbübische, wenn die Person dabei z.B. schmunzelt oder lächelt.

Bei der Beurteilung von Wahrnehmungen spielen folgende Aspekte eine wichtige Rolle:

- Die Ebenen der verbalen Aussagen gemäß dem TALK-Modell
- Stimmführung und Spannung der Stimme
- Körpersprachliche Signale
- Äußere Situation

TALK-Modell von Schulz und Thun

Jede Nachricht besitzt vier Aspekte:

- Tatsachenaspekt oder Sachinformation: Worüber wird informiert?
- Ausdrucksaspekt oder Selbstkundgabe: Was gibt der Sender von sich kund?
- Lenkungsaspekt oder Appell: Wozu soll der Empfänger veranlasst werden?
- Kontakt- oder Beziehungsaspekt: Was hält der Sender vom Empfänger?

Der Fokus des Empfängers entscheidet über die Reaktion.

Besondere Aufmerksamkeit ist dem ersten Eindruck zu widmen. Gerade hier, sollte auf die eigene Körpersprache geachtet werden, da der erste Eindruck ein starkes Gewicht hat.

Um bestimmte Dinge oder Sachverhalte näher zu bringen, kann man auch auf Metaphern zurückgreifen. Sie können einen komplexen Sachverhalt einfacher verständlich machen und auch Mitarbeiter besser motivieren.

3 Komplexe Systeme

Gruppen von Menschen und Organisationen verhalten sich ähnlich wie komplexe Systeme. Sie sind relativ unvorhersehbar und die kleinsten Änderungen haben große Auswirkungen. Die Beziehungen sind nichtlinear und beinhalten Rückkopplungseffekte: Veränderungen in einem Teil, lösen Veränderungen wiederum in einem anderen aus. Komplexe Systeme haben ebenfalls ein Gedächtnis und können aus anderen komplexen Teilsystemen bestehen.

Komfortzone verlassen

Der Mensch befindet sich meist in seiner Komfortzone. Verlässt er diese, bzw. wird er mit einer Situation konfrontiert, die von seinen gewohnten Verhältnissen abweicht, kann er leicht in die Panikzone gelangen. Dies bedeutet Stress bis hin zur Panik. Es gibt auch die Lernzone, wo die persönliche Weiterentwicklung statt findet. Hier lernt man, mit neuen Situationen umzugehen.

Ausprobieren

Um neue Dinge zu lernen, ist es wichtig sie auszuprobieren. Auch ein Scheitern ist hilfreich und verweist darauf, was nicht funktioniert und das man einen anderen Weg gehen muss. Im Vornherein ist nicht klar, welche Auswirkung welche Veränderung haben wird. Ein Fehler entsteht nur, wenn man etwas gegen besseren Wissens tut.

Zur Lösung eines Problems gibt es ein fünfstufiges, iteratives Muster:

1. Lösungsideen sammeln
2. Ideen bewerten und priorisieren
3. Handeln und umsetzen
4. Ergebnisse bewerten und Prozesse prüfen
5. Wieder auf Schritt 1 aufsetzen

Gruppendynamik

Jedes Team ist eine Gruppe, doch nicht jede Gruppe bildet ein Team. Team meint eine arbeits- und aufgabenbezogene Gruppe, in der die Mitglieder kooperieren müssen um ein gemeinsames Ziel zu erreichen. Um eine effektive Arbeit zu ermöglichen, lassen sich

Voraussetzungen definieren:

- Gruppe sollte überschaubar sein, ca. fünf Mitglieder
- Unterschiedliche Kompetenzen aber gemeinsames Interesse an der Aufgabe und Ziel
- Es wird die gleiche Sprache gesprochen
- Keine persönlichen Belastungen zwischen den Teammitgliedern
- Team hält sich an Regeln

Es gibt vier (zusätzlich zur Gründung) Phasen der Gruppendynamik:

- Forming
- Storming (Individuelle Ziele im Vordergrund)
- Norming (Gruppenziele im Vordergrund)
- Performing
- Reforming

Das Cynefin-Framework gibt fünf Vorgehensweisen für fünf Umgebungen an:

- Einfach: Beziehung zwischen Ursache und Wirkung ist offensichtlich. Best Practices aus der Schublade verwenden. Einfacher Regelkreis aus Wahrnehmen, Kategorisieren und Reagieren.
- Kompliziert: Beziehung zwischen Ursache und Wirkung erfordert tiefe Analyse oder Spezialwissen. Regelkreis: Wahrnehmen, Analysieren, Reagieren auf der Basis von ausgewählten guten Praktiken.
- Komplex: Ursache-Wirkung kann nur im Nachhinein erfasst werden. Neue, kreative Praktiken werden ausprobiert und erforscht. Ausprobieren, Wahrnehmen, Reagieren.
- Chaotisch: Keine Beziehung zwischen Ursache und Wirkung. Neue und ungewöhnliche Praktiken ausprobieren. Handeln, Wahrnehmen und Reagieren.
- Unordnung: Keine Kausalität ermittelbar, deswegen kann auch kein Vorgehen vorgeschlagen werden.

4 Selbstorganisation und Troja-Prinzip

Eine Hierarchie ist gut zur Bearbeitung unterteilbarer, komplizierter Aufgaben geeignet. In einem gleichberechtigen System sind mehr Menschen eingebunden. Hier sind mehr Menschen an der Lösungsfindung beteiligt. Betroffene werden zu Beteiligten. Das hat folgende Vorteile:

- Motivation: Eigene Interessen haben mehr Bedeutung. Aufgaben sind ganzheitlicher und abwechslungsreicher. Das individuelle Potenzial kann besser entfaltet werden.
- Flexibilität: Hohe Anpassungsfähigkeit an äußere und innere Situationen. Bessere Erkennung von Anpassungsbedarf und mehr Transparenz.
- Lenkbarkeit: Mehr Selbstorganisation.
- Zeitaufwand und Kosten: Anpassungen und Veränderungen laufen schneller ab.

Risiken:

- Überforderung: Selbstorganisation kann zu Ängsten und Problemen des Einzelnen führen.
- Konflikte: Das Konfliktpotenzial ist größer, u.a. wegen dem Fehlen von vorgegebenen Regeln.
- Hohe Anforderung an Führung: Viel Vertrauen in das Team ist notwendig.
- Zeitaufwand und Kosten: Lösungs- und Entscheidungsfindungen dauern länger.

Die Voraussetzungen für eine Selbstorganisation in einem Team lassen sich zusammenfassen unter den Begriffen Toleranz, Sozialkompetenz und Mut.

Troja-Prinzip

Das Troja-Prinzip beschreibt die evolutionäre oder revolutionäre Weiterentwicklung bzw. Veränderung eines bestehenden oder neuen Systems. Das einmal gefundene Optimum in einem Entwicklungsteam ist immer nur von begrenzter Dauer, da sich Rahmenbedingungen und andere Faktoren stetig verändern. Hier müssen also evolutionäre, oder aber, z.B. bei Zeitdruck, revolutionäre Maßnahmen greifen.

Organisation von Teams

Selbstorganisierte Gruppen sind regel- und wertebasiert. Die Führung erfolgt über Regeln und das Vorleben von Werten. Die Führung übernimmt dabei nicht eine einzelne Person. Für kleine Gruppen ist dies einfacher zu realisieren, mittels XP oder Scrum. Schwierig wird

es, wenn das Team wächst.

Man kann eine Organisation auch als Matrix darstellen, eine sogenannte Organisationsmatrix. Sie enthält verschiedene Schichten; Management, strategische und operationale Sichtweisen, Abteilungen, usw. Sie ist primär eine Kommunikationsmatrix. Die Kommunikationskanäle müssen explizit ausgebaut und umgebaut werden. Die Steuerung einer echten Matrix erfolgt über gemeinsame Ziele und Visionen. Die Motivation kommt aus der Identifikation des Einzelnen mit den Zielen und der Vision.

5 Ziele und Prioritäten

Ein Ziel beschreibt einen zukünftigen, erstrebenswerten und sich nicht von alleine einstellenden Zustand. Er stellt allgemein dar, was eine Person, eine Gruppe oder ein ganzes Unternehmen erreichen möchte. Es wird nicht beschrieben, wie das Ziel erreicht werden soll. Vier Punkte dabei sind zu beachten:

- Zielinhalt definieren: Was soll erreicht werden?
- Zielmenge definieren: Wie viel wird angestrebt?
- Zielzeitpunkte definieren: Wann soll es erreicht sein?
- Zielgrund klarstellen: Warum soll das Ziel erreicht werden?

Business Motivation Model (BMM)

Das BMM gibt einige Definitionen vor:

- Vision: Beschreibt das ultimative, womöglich unerreichbare Ziel, das sich ein Unternehmen setzt.
- Zielstellung (Goal): Führt die Vision weiter aus.
- Planziel (Objectives): Messbare, erreichbare und zeitlich begrenzte Ziele
- Mission: Beschreibung des fortwährenden Verhaltens eines Unternehmens, mit dem die Vision erreicht werden soll.
- Strategie: Unterstützt meist eine Zielstellung.
- Taktik: Unterstützt meist ein Planziel.

Ziele schriftlich erarbeiten

Ziele gehören zu den wenigen Dingen, die stets schriftlich zu dokumentieren sind. Dabei wird eine erhöhte Klarheit garantiert und sie können konkret kommuniziert werden. Die Wahrscheinlichkeit der Zielerreichung wird damit bereits erhöht.

Prioritäten setzen

Pareto-Prinzip

Das Pareto-Prinzip besagt, dass eine kleine Menge hoher Werte mehr zum Gesamtergebnis beiträgt als eine große Menge kleiner Werte.

ABC-Analyse

Trifft das Pareto-Prinzip zu, kann zur Vertiefung eine ABC-Analyse durchgeführt werden.

- A - hoch - muss: Alle Aufgaben dieser Prioritätsstufe müssen vollständig erledigt werden.
- B - mittel - soll: Es ist sicherzustellen, dass möglichst viele Aufgaben dieser Prioritätsstufe in ausreichender Qualität erledigt werden.
- C - niedrig - kann: Aufgaben auf dieser Stufe können erledigt werden.

Relative Priorität

Die relative Einordnung von Prioritäten (mittels einer relativen Reihenfolge der Aufgaben), führt zu brauchbaren Lösungen, auch wenn es nur wenig Detailinformationen gibt.

Beispielsweise ist es unklar wieviele Einwohner Reykjavik hat, doch ist die Vermutung richtig, dass es weniger sind als in Berlin und mehr als in Bitterfeld.

6 Erfolgreiche Besprechungen

Bei Besprechungen gilt es folgenden Merksatz festzuhalten: Struktur vor Inhalt!

Es ist wichtig, die Rahmenbedingungen vorab festzulegen. Zeitraum, ggf. Pausen, Nutzung von Räumen etc. Offene Fragen sollen frühestmöglich geklärt werden, damit man sich in der eigentlichen Besprechung auf das Wesentliche konzentrieren kann, ohne aber auch die Pflege von Beziehungen zu vernachlässigen.

Allgemeine Regeln für die Durchführung

Eine Agenda muss festgelegt werden: Worum geht es genau? Planziele zu erarbeiten, Maßnahmen zur Erreichung von Planzielen, oder nur ein Status-Meeting über den aktuellen Stand bestimmter Maßnahmen?

Um den Fokus zu halten innerhalb einer Veranstaltung, hilft es eine Moderation zu verwenden. Die Vorabdefinition von Ergebnistypen ist eine andere Maßnahme um den Fokus zu halten, z.B. priorisierte Listen von Vorschlägen. Wichtige Fragen zur Orientierung sind immer: Für wen wird die Veranstaltung durchgeführt, wer ist der Auftraggeber? Auch ist auf eine klare Formulierung und den Einsatz klar definierter Wörter zu achten.

In der Vorbereitung kann ein Fragenkatalog zusammengestellt werden, um die Ziele näher zu definieren. Einige Beispiele dazu:

Ziel der Besprechung

- Welche konkreten Ergebnisse werden angestrebt?
- Woran kann die Zielerreichung gemessen werden?
- Welcher Zeitrahmen ist für die Zielerreichung zu erwarten?

Teilnehmer an der Besprechung

- Welche der Besprechung dienende Funktion hat jeder der Teilnehmer?
- Sind bestimmte Informationen notwendig, die nur von speziellen Personen beigesteuert werden können?
- Gibt es Teilnehmer, die nur zu bestimmten Teilen anwesend sein müssen?

Agenda für die Besprechung

- Welche Arbeitspunkte sind notwendig, um die Ziele zu erreichen?
- In welche Zeitabschnitte können die Arbeitspunkte unterteilt werden?
- Welche Priorität können den Punkten zugeordnet werden?

Eine Besprechung durchführen

Die Moderation kann die Person übernehmen, die zur Veranstaltung eingeladen hat. Die von der Agenda abgearbeiteten Punkte werden entsprechend markiert. Als erstes sollte ein einfacher Aspekt behandelt werden, um den Teilnehmern ein optimistisches Gefühl zu vermitteln. Bei zu intensiven Gesprächen zwischen Fachexperten sollte der Moderator eingreifen. Er hat zwei Aufgaben: Transparenz und Klarheit schaffen sowie für die Stimmigkeit der Kommunikation zu sorgen.

Besprechung nachbereiten

Protokolle, z.B. in Form von Fotos (Flipcharts, Metaplan-Wände, andere Artefakte) müssen aufgezeichnet werden. Anhand des Protokolls können auch im Nachhinein Beschlüsse nachverfolgt werden.

7 Zeitmanagement

Eine wichtige Frage für das Zeitmanagement ist: Wann unterbreche ich meine Arbeit, um die Säge zu schärfen, damit es danach wieder schneller und kraftsparender vorangeht? Welchen Zeitanteil möchte ich dem Schärfen der Säge widmen? Um das zu lösen, gibt es die Eisenhower-Methode. Aufgaben werden nach zwei Aspekten kategorisiert:

- wichtig: Die Führungskraft kümmert sich selbst darum.
- dringlich: Die Aufgabe ist sofort zu erledigen, um das Projekt nicht zu gefährden.

Mit vier Fragen lassen sich anstehende Aufgaben strukturieren:

- Können wir delegieren und wenn ja, an wen?
- Wann ist eine bestimmte Aufgabe in unserem persönlichen Tagesrythmus am besten zu erledigen?
- Wie oder von wem kann eine Aufgabe effizienter bearbeitet werden?
- Können wir die ganze Aufgabe oder Teile davon weglassen?

Mit Hilfe von Kanban-Boards lassen sich Aufgaben in eine Übersicht bringen. Engpässe und eventuelle Probleme lassen sich hier visuell leicht erkennen.

Störungen blocken

Das ständige Wechseln von Aufgaben oder die parallele Bearbeitung mehrerer Aufgaben ist nicht empfehlenswert und schädlich für die Arbeitszeit. Es ist wichtig, sich Verhältnisse zu schaffen, eine 'goldene Stunde', in der man ungestört an einer Aufgabe arbeiten kann. Ohne Störung von Kollegen, Emails oder anderen Einflüssen.

8 Wie funktioniert Führung

Führung ist ein richtungsweisendes und steuerndes Einwirken auf den anderen, um eine bestimmte Zielvorstellung zu erreichen. Eine formale Führungskraft kann nicht Gleicher unter Gleichen sein.

Führung ist mit einer Form von Macht verbunden. Macht entsteht über einen oder mehrere von vier Wegen: Person und Persönlichkeit - Funktion - Sache, Information und Expertise, und - Festlegung und Position.

Führungsstil

- Aufgabenorientierung
 - Verfahren vorschlagen
 - Ziele setzen
 - Aufgaben koordinieren und die Gruppe organisieren
- Mitarbeiterorientierung
 - Persönliche Beziehungen pflegen
 - Direkte Kontakte zu den Mitarbeitern halten
 - Wünsche, Sorgen und Vorstellungen der Mitarbeiter kennen
- Mitwirkungsorientierung
 - Mitarbeiter um Rat fragen
 - Mitarbeiter in Lösungen bzw. Entscheidungen einbeziehen
 - Kenntnisse und Kompetenzen der Mitarbeiter nutzen

Es gibt beim Führungsverhalten in Gruppen weder die optimale Persönlichkeit noch das optimale Verhalten. Es ist immer der situative Kontext zu berücksichtigen. Gute Führung hängt von der konkreten Situation ab (welches Entscheidungsverhalten ist in welcher Situation angemessen).

Ein Manager nimmt zehn verschiedene Rollen ein: Interpersonell (Repräsentant, Führungskraft, Kontaktpfleger), Informationsgeber (Beobachter, Informant, Sprecher) und Entscheidungsträger (Unternehmer, Problemlöser, Ressourcenverteiler, Unterhändler).

Der Manager, bzw. die Führungskraft ist in der Lage, sehr früh nützliche Hinweise aus den Informationen herauszuziehen aufgrund des viel dichteren Informationsnetzes, welches ihn umgibt.

Führungsebenen

Es gibt meist mindestens zwei Führungsebenen. Jede hat unterschiedliche Bereiche zu verantworten, und ist dabei immer weiter von der konkreten Arbeit entfernt. Das bietet eine bessere Übersicht und hilft Entscheidungen zu treffen, die zu Lasten eines ihrer Teile gehen. Sie bieten auch die Möglichkeit, Eskalationsstufen aufzubauen. Diese sind wichtig, um Konflikte zu lösen, die von den Beteiligten nicht selbst gelöst werden können.

9 Kontakt und Motivation

Direkte Kommunikation

Die Grundlage einer Mitarbeiterorientierung ist der direkte Kontakt. Gerade in bestimmten Phasen eines Projekts ist der direkte Kontakt durch nichts zu ersetzen. Gerade in der Projektvorbereitung und den initialen Aktivitäten zu Projektbeginn.

Hier müsse viele unterschiedliche Informationen zusammengetragen werden um eine tragfähige Projektabschätzung abzuleiten. Eine direkte Kommunikation ist hier unerlässlich. Damit asynchrone Kommunikation funktioniert, ist es wichtig, dass sich die beteiligten Personen persönlich kennen.

Direkte Kommunikation erfolgt in Iterationsschritten bzw. an Meilensteinen. Für den Iterationsauftakt und den -abschluss gilt das Gleiche wie für Projektbeginn und -abschluss. Hier fließen jeweils viele Informationen ein und eine direkte Kommunikation sollte stattfinden.

Motivation

Neben der Bedürfnispyramide von Maslow gibt es auch von Frederick Herzberg eine Zwei-Faktoren-Theorie. Sie beinhaltet den sogenannten Hygienefaktor, der Unzufriedenheit ausschließt, aber keine Zufriedenheit herstellt, und Motivatoren, welche die Motivation zur Leistung verbessern.

Die Motivation ist klassische Aufgabe der Projektleiter und Personalverantwortlichen. Es wird oft auf extrinsische Motivation zurückgegriffen, also äußere Motivation. So eine Motivation ist oft nicht möglich, und wenn, dann nur von kurzer Dauer. Sie erfolgt oft über monetäre Anreize. Es ist wichtig, Demotivatoren (Hygienefaktoren) zu eliminieren. Für die innere, intrinsische Motivation sind die Mitarbeiter selbst verantwortlich.

Agile Projekte sind eine gute Ausgangsbasis für hohe intrinsische Motivation, aufgrund des zugrunde liegenden Menschenbilds. z.B. Menschen handeln zielgerichtet und in positiver Absicht, d.h. um ihre Bedürfnisse zu erfüllen.

Es ist wichtig eine Balance zwischen Aufwand und Ertrag zu finden. Dafür gibt es drei Strategien:

- A-Minus-Strategie: Mit einem Dienst nach Vorschrift wird der Aufwand so lange reduziert, bis er wieder zum Ertrag passt.

- C-Plus-Strategie: Durch eine höhere Kompensation über mehr Gehalt wird versucht, die Diskrepanz zwischen Aufwand und Ertrag in Einklang zu bringen.
- E-Plus-Strategie: Durch eine Steigerung des Ertrags, zB. mehr Anerkennung, wird versucht ein positives Verhältnis herzustellen.

Möglichkeiten der Weiterentwicklung

In der Softwareentwicklung bleibt die Zeit nicht stehen, daher muss man sich regelmäßig weiterentwickeln. Im Wesentlichen stehen uns vier Möglichkeiten zur Verfügung, um uns und unsere Mitarbeiter fachlich und in ihrer Persönlichkeit weiterzuentwickeln:

Seminare:

Sind eine Kombination aus Vortrags- und Übungsteilen. Dienen dazu, schnell und gezielt Wissen zu vermitteln, über Präsenzveranstaltungen, das Internet oder in Form eines Literaturstudiums oder auch durch Mischformen. Der vom Teilnehmer aufgenommene Inhalt ist stark vom didaktischen Konzept abhängig. Meist schafft erst die Umsetzung des Gelernten in die Praxis den Wert. Wir behalten 10% vom Gelesenen, 20% vom Gehörten, 30% vom Gesehenen, 50% durch hören und sehen, 70% vom selbst Gesagten und 90% durch das, was wir selbst machen. Eine Prüfung erzielt nur dann positiven Erfolg, wenn sie mit ihren Inhalten ausreichend praxixnah ist.

Teambildungsworkshops:

Behandeln eher die Individuelle Situation der Gruppe und aktuell konkrete Probleme. Soft Skills und Kooperationsfähigkeiten der Mitarbeiter können nur praktisch gefördert werden. Teambildungsworshops sind gerade zu Beginn einer Gruppenbildung gut geeignet. Ein Risiko besteht darin, dass schwerwiegende und bislang nur vermutete Probleme einzelner oder der Beziehungen zueinander aufgedeckt werden, bzw hochkommen. Das ist durchaus gewollt, erfordert aber auch eine Hohe Kompetenz der Trainer und ausreichend viel Zeit. Soziale Probleme sind stets dringlich und sofort zu behandeln.

Persönliche Förderung durch einen Mentor (Mentoring):

Wissen, Werte und Erfahrungen des Mentors werden an eher unerfahrene Personen weitergegeben. Er bezieht keine neutrale Position, sondern hat eher persönliches Engagement und Bezug zu der Person (väterlicher Mentor). Ein Mentor muss klar und fest im Arbeitsalltag verankert sein. Die Wahl des richtigen Mentor-Mentee-Paars ist entscheidend. Ein Mentoring zum Zeitpunkt der Einarbeitung verkürzt die Zeit, bis ein neuer Mitarbeiter produktiv mitarbeiten kann. Der Mentor muss sich immer wieder klar machen, welche Rolle er gerade einnehmen sollte, Begleiter, Führer, Berater oder Erzieher!

Coaching:

Ist eine Art Hilfe zur Selbsthilfe. Der Coach entwickelt im Rahmen des Coachings seine individuelle Lösung selbst. Coaching stellt eine Win-Win Situation dar, in der eine zielorientierte Teamarbeit zwischen Coach und Coachee erfolgt. Es ist ein nützliches Beziehungsmodell, um den Coachee zu befähigen, selbst ans Ziel zu gelangen. Ein Coaching-Prozess (auch GROW) gliedert sich in fünf Phasen ein:

1. Vertrauen gewinnen, absolute Verschwiegenheit des Coaches
2. Ziele formulieren zu Ausrichtung und klaren Abgrenzung des Angestrebten
3. Entwickeln einer Strategie
4. Öko-Check bzw. Realitäts-Check zur Überprüfung der Lösungsideen
5. Future Pace, der Plan, wann was genau getan werden soll

Einzelne Mitarbeiter weiterentwickeln

Jedes Individuum hat die in der folgenden Tabelle abgebildeten acht Präferenzen/Eigenschaften in sich. Allerdings scheint es laut C.G. Jung und Isabel Meyers unmöglich, diese gleichmäßig und angemessen zu entwickeln.

Neigung	Eigenschaften
<u>Extraversion</u>	gesellig, interagierend, hat viele Beziehungen, interessiert am Äußeren und an Geschehnissen
<u>Introversion</u>	territorial, konzentriert, hat nur sehr wenige Beziehungen, interessiert am Inneren und an der inneren Reaktion
<u>Sensorische Wahrnehmung</u>	auf Erfahrung basierend, realistisch, am Tatsächlichen orientiert, sachlich, praktisch, vernünftig
<u>Intuitive Wahrnehmung</u>	auf Ahnung basierend, spekulativ, am Möglichen orientiert, träumerisch, erfängerisch, einfallsreich
<u>analytische Beurteilung (Thinking)</u>	objektiv, an Prinzipien und Richtlinien sowie Gesetzen orientierend, Umstände und Kriterien beachtend, standfest, gerecht, in Kategorien einteilend, kritisierend, analysierend
<u>geFühlsmäßige Beurteilung</u>	subjektiv, individuell, gesellschaftliche Werte achtend, mildernd, überzeugt, human, harmonisch, anerkennend, anteilnehmend
<u>Urteilen bzw. Folgern (Judging)</u>	entschieden, beschließend, feststehend, vorausplanend, etwas geschehen machend, abschließend, Entscheidungen treffend, planmäßig, vollendet, endgültig, umsetzend
<u>Wahrnehmen (Perceiving)</u>	unentschieden, weitere Informationen beschaffend, flexibel, offen und beweglich, anpassungsfähig, etwas geschehen oder entstehen lassen, freie Wahl lassen, Entscheidungen hinauszögern, abwartend

Abbildung 1: acht Präferenzen/Eigenschaften eines Individuums

Man hat meist auf jeder der 4 Ebenen (zu sehen in Abbildung 1) einen Favoriten, der unsere Persönlichkeit ausmacht. Man sollte versuchen, seine weniger ausgeprägten Präferenzen auszubauen und somit Defizite abzubauen, damit man in möglichst vielen Situationen angemessen agieren kann. Durch Verhaltensänderungen kann man die zu gering ausgeprägten Präferenzen ausbauen und üben. Das ist wie z.B. mit links statt mit rechts zu schreiben.

Weiterentwicklung der Persönlichkeit

Man könnte sich an den folgenden Beispielen orientieren, um die Präferenzen weiterzuentwickeln:

Extraversion (Handlungsspektrum eines eher Introvertierten erweitern):

- In Gruppen spontan sprechen und sich äußern
- sich an einer Diskussion ohne Vorbereitung zu beteiligen
- eine Rede halten
- sich jemandem vorstellen, den man nicht kennt

Introversion (Handlungsspektrum eines eher Extravertierten erweitern):

- ungestörte Zeit alleine verbringen
- in einer Gruppe völlig ruhig sein
- kleine Projekte alleine durchführen

Beurteilend (Handlungsspektrum eines stärker Wahrnehmenden erweitern):

- To-do Liste erstellen und sie nach Priorität sortieren
- einen Plan für eine größere anstehende Aufgabe erstellen

Wahrnehmend (Handlungsspektrum eines eher Beurteilenden erweitern):

- Verbringe ungeplante Zeit und folge deinen Impulsen
- überprüfe im Nachhinein nochmal eine entgültige Entscheidung

Sensorik (Handlungsspektrum eines eher Intuitiven erweitern):

- etwas sehr sorgfältig Beobachten
 - nonverbales Verhalten (als Extravertierter)
 - die Natur (als Extravertierter)
 - den eigenen physischen Zustand (als Introvertierter)
- langsames Essen mit Konzentration auf die Sinne (Introvertierter)
- etwas machen, dass detaillierte Aufmerksamkeit erfordert (Extravertierter)
- alle Fakten zu einem Thema aufschreiben
- beschreibe eine Aktivität Schritt für Schritt

Intuition (Handlungsspektrum eines eher sensorisch Wahrnehmenden erweitern):

- Brainstorming (Introvertierter)
- Tagträumen oder in seinen Phantasien leben (Introvertierter)
- Sich selber in 5 Jahren vorstellen (Introvertierter)
- entwerfe ein neues Design, GUI oder eine Architektur (Extravertierter)

Denken (Handlungsspektrum eines eher fühlenden erweitern):

- Aktivitätsdiagramm oder Ablaufplan für eine funktionale Anforderung (Introvertierter)
- Alle Vorteile und Kosten einer Entscheidung auflisten (INTrovertierter)
- jemandem sagen, was man an ihm schwierig findet (Extravertierter)
- definiere etwas unpräzise (Introvertierter)

Fühlen (Handlungsspektrum eines eher denkend Beurteilenden erweitern):

- eigene Werte klären (Introvertierter)
- Einfühlungsvermögen zeigen (Extravertierter)
- mehr emotionale Wörter benutzen (Introvertierter)
- Dinge auflisten, die man mag oder die einen stören (Introvertierter)
- jemandem ein Kompliment für seine Persönlichkeit machen (Extravertierter)

Zu vielen dieser Beispiele, fällt einem sicherlich sofort ein Kollege oder Bekannter ein, der sich so verhält. Mit der richtigen Teambildung, von zwei oder mehr Personen, kann man Defizite ausgleichen. Außerdem kann es sehr sinnvoll und effizient sein, Menschen auszuwählen, deren Eigenschaften sich in Einstellung und Funktion ergänzen. Denn es ist für einen Menschen alleine fast unmöglich, alle Aspekte ausreichend einzubringen.

Hochleistungsteams aufbauen

Im Folgenden sollen einige Ideen zur Steigerung der Leistungsfähigkeit eines Teams aufgezeigt werden.

Das Kontextmodell zur Teamentwicklung

Jedes Team setzt sich aus vielen Persönlichkeiten zusammen, die jeweils über eigene relevante Soft Skills wie Kommunikations-, Kooperations- und Teamfähigkeit verfügen. Die jeweiligen persönlichen Vorerfahrungen mit anderen Teams machen die Vorgeschichte eines Teams aus. Im Rahmen einer konkreten Aufgabe oder eines Projekts treffen sie aufeinander. Dies kann man in einem Teambildungsworkshop durchspielen. Hier bringt jeder seine eigenen Bedürfnisse mit, die den anderen in der Regel teilweise nicht bekannt sind. Hierbei geht man auf die Wahrnehmungsfähigkeiten der Personen ein. Was nimmt jedes einzelne Teammitglied wahr? Zuletzt benötigt man noch eine Prozesskompetenz, die bereits durch einzelne Personen oder im Rahmen des Teamentwicklungsworkshops von außen hinzukommt. Dadurch erreicht das Team eine Arbeitsfähigkeit, die auf die konkrete Aufgabe bezogen ist. Regeln und Vereinbarungen werden implizit und explizit aufgestellt, so entwickelt das Team eine eigene Identität und die soziale Kompetenz der Mitglieder.

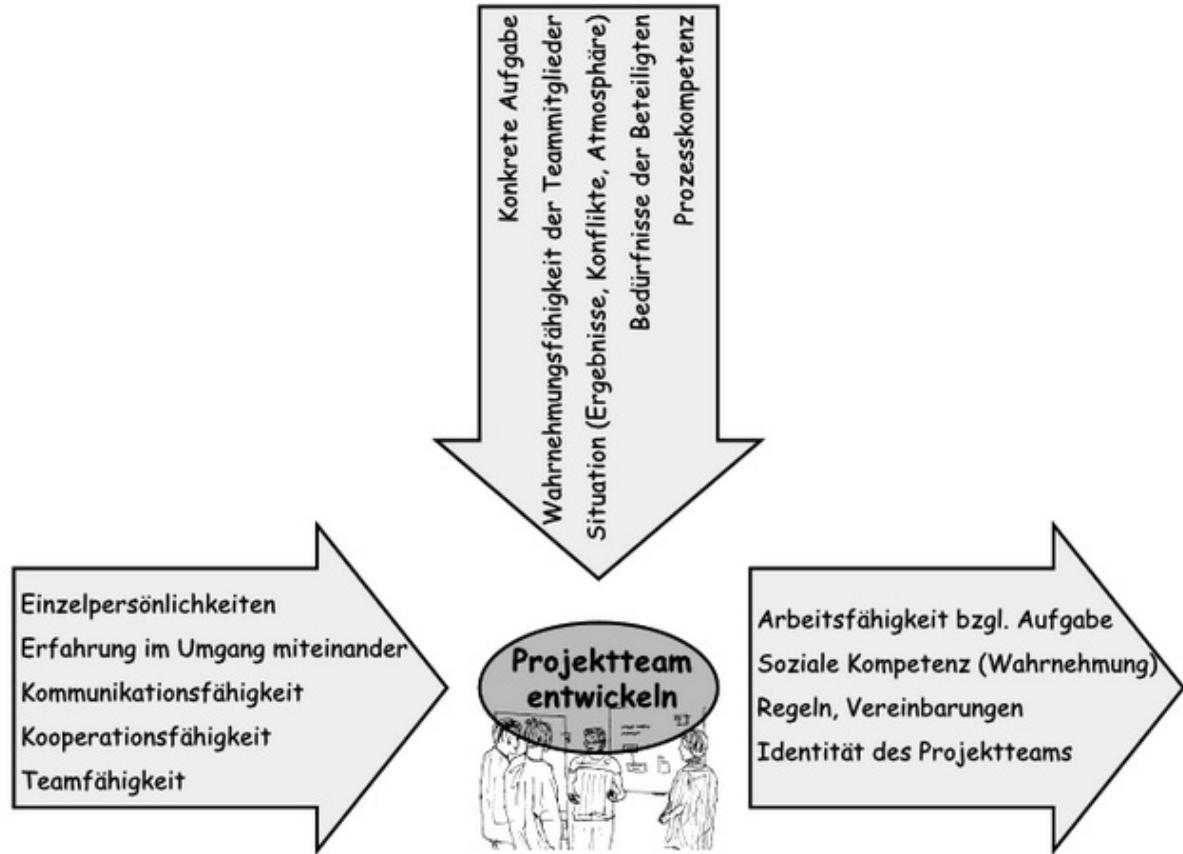


Abbildung 2: Kontextmodell zu Teamentwicklung

Über die Reflexion der Mitglieder über sich selbst und die Wahrnehmung der anderen Mitglieder kann solch ein Team in einer Retrospektive am Ende jedes Schrittes bewusst lernen und sich somit als Projektteam entwickeln. Außerdem ist es wichtig, aufkommende Konflikte unter Teammitgliedern sofort und direkt anzusprechen und eine Lösung zu finden. Auch das Vertrauen in die Kompetenz und das Verantwortungsbewusstsein der Kollegen spielt eine große Rolle, damit man für eine entspanntere, effizientere Arbeitsumgebung im Team sorgen kann.

Gegensätze im Team ergänzen

In der Softwareentwicklung ist man meist mit einem dynamischen, technischen sowie fachlichen Umfeld konfrontiert. Hier lohnt es sich vor allem, wenn man bei der Teamentwicklung nicht nur den leichtesten Weg geht. Die Stärke des Teams ergibt sich aus den typologischen Unterschieden und den individuellen unterschiedlichen Stärken. Dies ist der Weg um Hochleistungsteams aufzubauen.

Was macht ein Team zu einem Spitzenteam?

Manche Projekte sind besonders schwierig oder riskant und von großem wirtschaftlichem Interesse. Wenn besonders hohe Flexibilität und schnelle Reaktionen auf Veränderungen im Umfeld verlangt werden, wird es Zeit über Hochleistungsteams nachzudenken. Ähnlich ist

es auch bei besonders innovativen Themen. Unter einem Hochleistungsteam versteht man heterogene Teams, die ihre unterschiedlichen Stärken situationsbedingt kombinieren und stets flexibel sind. Außerdem können sie mit kurzer Reaktionszeit, eine dem Problem angemessene, überdurchschnittliche Leistung erbringen. In diesen von unterschiedlichen Merkmalen geprägten Teams ist es sehr wichtig, dass die zwischenmenschlichen Beziehungen nicht nur ausreichend sondern vollständig geklärt sind. Ansonsten kommt es zu einer "Explosion", die alles ruinieren kann. Sämtliche Konflikte oder aus Vorgeschichten mitgebrachte Probleme müssen vollständig aufgelöst sein.

Teamleitung solcher Teams

Die Teamleiter solcher Teams müssen höchsten Anforderungen gerecht werden. Sie brauchen Erfahrung in der Moderation von Konflikten bis hin zur Mediation. Zusätzlich müssen sie nah an der Gruppe sein und über eine gut ausgeprägte und sensible Wahrnehmung der aktuellen Gruppe verfügen. Die Ziele des Teams üben eine hohe Attraktivität aus. Im wesentlichen sorgt die Führung also für ein optimales Arbeitsumfeld und die sofortige Klärung gruppeninterner Irritationen.

Software Architecture for Developers: Volume 1

Zusammenfassung des Buches:

Titel: Software Architecture for Developers: Volume 1 - Technical leadership and the balance with agility

Verfasser: Simon Brown

Verlag: Leanpub

Jahr: 2016

ISBN:

Zusammenfassung von: Gamze Soeylev Oektem, Lutz Winkelmann, Yannick Kloss

Was ist Softwarearchitektur?

In diesem Teil werden wir sehen, was der Unterschied zwischen Softwarearchitektur und Softwaredesign ist.

Was ist Architektur?

Es ist nicht einfach, zu definieren, was Architektur ist. Es gibt zwei Begriffe: Architektur als Substantiv und als Verb.

Als Substantiv

Architektur kann als Struktur definiert werden. Es geht um die Dekomposition eines Produkts.

Als Verb

Architektur kann auch als Verb verstanden werden. In diesem Fall geht es darum zu verstehen, was man bauen muss und wie man es bauen muss. Hier liegt der Fokus also auf den Anforderungen.

Architekturtypen

Wenn man verschiedene Menschen fragt, bekommt man viele verschiedene Antworten zum Softwarearchitekturtyp. Es ist egal, was man baut (Softwaresystem, Netzwerk oder Datenbank), man braucht für eine erfolgreiche Lösung zwei Schritte. Zuerst muss man das Problem verstehen. Dann eine Vision zu erzeugen, die man mit jedem Beteiligten des Produktes kommunizieren kann. Es geht um Struktur und Vision, wenn man über Architektur spricht.

Was ist Software-Architektur?

Applikationsarchitektur

Bei der Applikationsarchitektur geht es um die niedrigeren Stufen (lower levels) des Softwareentwurfs. Meistens gibt es nur eine Technologie (Java, Microsoft .Net, etc.). Die Bauteile beinhalten Programmiersprache, Bibliotheken, Frameworks, etc. Bei der Applikationsarchitektur geht es um die Software und die Organisation des Codes.

Systemarchitektur

In einem Softwaresystem gibt es meistens mehrere Applikationen. Diese müssen zusammenarbeiten. Die Mehrheit der Softwaresysteme kommunizieren mit der Außenwelt, daher sind Interoperability und Integration mit den anderen Systemen sehr wichtig. Zusätzlich ist die Hardware auch wichtig. Die Bauteile beinhalten sowohl Software als auch Hardware (z.B. Programmiersprache, Frameworks, Servers, Infrastruktur). Systemarchitektur ist abstrakter als Applikationsarchitektur.

Softwararchitektur

Softwarearchitektur ist die Kombination der Applikation- und Systemarchitektur. Hier geht es um:

- Logging und Exception Handling
- Sicherheit
- Performance, scalability, availability
- Real-world constraint of the environment
- Interoperability/ Integrität,
- Operational, support and maintenance requirements
- ... Wir müssen also einen allgemeinen Blick auf die Software werfen.

Enterprise-Architektur - Strategie statt Code

Bei der Enterprise-Architektur geht es nicht darum, wie Technologie funktioniert, sondern wie Technologie in der Organisation besser benutzt werden kann. Enterprise-Architektur guckt wie man Menschen, Prozesse und Technologien am besten organisiert und einsetzt, damit

die Organisation besser und effizienter funktionieren kann.

Architektur vs. Design

Was ist der Unterschied zwischen Design und Architektur? Design und Architektur sind sehr ähnlich. Jede Architektur ist Design, aber nicht jedes Design ist Architektur. Wichtige Entscheidungen im Design-Prozess, also Entscheidungen die man nicht leicht rückgängig machen kann, sind Architektur - alles andere ist Design. Als Architektur gelten:

- Die Systemform (z.B. client-server, web-based etc.)
- Die Struktur des Softwaresystems (z.B. Komponenten, Interaktionen)
- Die Wahl der Technologien (z.B. Programmiersprachen)
- ...

Ist Softwarearchitektur wichtig?

Guter Code ist nicht immer genug für gute Software. Ein kleines bisschen Architektur hilft um Problemen vorzubeugen.

Ein Mangel an Softwarearchitektur verursacht Probleme

Ohne Softwarearchitektur kann man keine gute Softwarestruktur haben. Ohne gute Softwarearchitektur kann unsere Software häufig zu langsam, unsicher, störungsanfällig, schwer zu warten etc. sein.

Vorteile von Softwarearchitektur

- Ein klare Roadmap für das Team
- Bessere Koordinierung
- Ein Framework für Risikoidentifizierung
- Eine Struktur um Lösungen gegenüber verschiedenen Zielgruppen zu kommunizieren

Braucht jedes Softwareprojekt Softwarearchitektur?

Ja. Nur muss man sich entscheiden, wieviel Softwarearchitektur man braucht. Es hängt von der Grösse und Komplexität des Projekts und der Grösse und Erfahrung des Teams ab.

Die Rolle der Softwarearchitektur

Softwarearchitekt ist eine Rolle, kein Titel.

Architectural Drivers

Ein wichtiger Teil der Rolle als Softwarearchitekt besteht darin, die Geschäftsziele zu verstehen und Anforderungen und Begrenzungen der Umgebung zu managen.

Software designen

Beim Softwaredesign geht es darum zu verstehen, wie die Probleme (bzgl. Anforderungen und Begrenzungen) gelöst werden können. Der wichtigste Teil ist die richtige Technologie zu wählen.

Technische Risiken

Der beste Entwurf und die beste Technologie bedeuten nicht unbedingt Erfolg. Viele Firmen wählen eine Technologie, um Kosten zu sparen. Eigentlich muss man sich fragen, ob die Technologie macht, was sie machen soll. Eine Architektur funktioniert, wenn die nicht-funktionalen Anforderungen erfüllt sind. Eine der wichtigsten Probleme von Software ist, dass sie komplex und abstrakt ist. Daher ist es schwer, sich Software vorzustellen. Wenn möglich, sollte man Architektur testen. Insgesamt geht es darum, technische Risiken vorzeitig zu identifizieren und zu managen.

Architekturevolution

Architektur sollte sich während des Entwicklungszyklusses weiterentwickeln. Deswegen braucht man kontinuierliche technische Führung für die Architektur.

Coding

Manche Firmen wollen nicht, dass ihre Softwarearchitekten Code schreiben. Aber dies ist keine gute Entscheidung. Nur wenn das Projekt sehr gross ist, kann der Architekt keinen Beitrag beim Coding leisten.

Qualitätssicherung

Qualitätssicherung ist die wichtigste Aufgabe von Softwarearchitekten. Allerdings wird diese Aufgabe in den meisten Projekten vernachlässigt.

Zusammenarbeiten oder scheitern

An einem Softwareprojekt sind viele Personen mit unterschiedlichen Interessen beteiligt. Jede dieser Personen beteiligt sich in gewisser Weise an der Entwicklung der Softwarearchitektur. Als Softwarearchitekt muss man mit allen, an einem Projekt beteiligten Personen, zusammenarbeiten.

Sollten Softwarearchitekten Code schreiben?

Softwarearchitekten sollten definitiv Code schreiben.

Code schreiben

Der Architekt muss nicht der beste Coder im Team sein. Aber er sollte Code schreiben können. Wenn Softwarearchitekten sich am Code schreiben beteiligen, hilft dies die Lücke zwischen Entwicklung und Architektur zu schliessen.

Prototypen, Frameworks und Foundations

Wenn ein Architekt nicht die Möglichkeit hat, Code zu schreiben kann er Prototypen entwickeln. Das hilft dabei eine bessere Kommunikation mit dem Team zu entwickeln und um zu sehen, ob die Architektur funktionieren wird. Ausserdem kann er Frameworks und Foundations für das Team kreieren.

Experimentieren und auf dem neuesten Stand bleiben

Der Architekt sollte in seiner Freizeit seine Code-Skills weiterentwickeln und versuchen technologisch auf dem neuesten Stand zu bleiben.

Softwarearchitekten sollten Baumeister sein

Wenn wir uns anschauen, woher das Wort Architekt kommt, sehen wir, dass es auf Latein Baumeister bedeutet. Baumeister mussten sich beweisen, bevor sie Baumeister werden konnten: Sie mussten zuerst alle anderen Jobs auf dem Bau machen, bevor sie Baumeister wurden. Aber nachdem sie Baumeister wurden hatten sie keine Zeit um diese Arbeiten weiterzumachen. Es gibt viele Parallelen zwischen Baumeistern und Softwarearchitekten.

Man sollte vorsichtig sein, wenn man bei der Entwicklung des Codes hilft. Die Entwickler im Projekt nehmen Personen nicht ernst, wenn sie nicht Softwareentwickler im Projekt sind. Sie können sich auf den Schlipps getreten fühlen. Allerdings gibt es auch Unterschiede zwischen Baumeistern und Softwarearchitekten. Zum Beispiel gibt es bei Softwarearchitekten kein Ausbildungsmodell und im Gegensatz zu den Baumeistern müssen die Architekten in verschiedenen Projekten mit verschiedenen Teams arbeiten. Wir brauchen eigentlich ein Ausbildungssystem für Softwarearchitekten, weil jedes Softwareprojekt einen eigenen Architekten braucht.

10 From developer to architect

Es ist schwer zwischen Softwarearchitekten und Softwareentwicklern zu unterscheiden. Ein Softwarearchitekt versucht das Problem zu abstrahieren und das große Ganze wahrzunehmen. Hierdurch soll das Projekt besser skalieren und zu den richtigen Softwaredesign-Entscheidungen führen.

Um einen guten Softwarearchitekten erkennen zu können, sollten die bisherigen Erfahrungen in den Kervoraussetzungen wie Verantwortlichkeit und Führungskompetenz unabhängig voneinander betrachtet werden. Wichtig ist hierbei, dass die Kernkompetenzen an das jeweilige Projekt angepasst sein müssen. Es ergibt sich beispielsweise ein Unterschied daraus, ob ein bestehendes Projekt erweitert oder ein neues Projekt begonnen wird und ob weitere Kernaufgaben wie das Aufstellen nicht-funktionaler Anforderungen wahrgenommen oder als das Problem anderer betrachtet werden.

Als Softwareentwickler nimmt man häufig unbewusst Teilaufgaben eines Softwarearchitekten wahr und lernt somit die nötigen Fähigkeiten dieser Rolle während man seine eigenen Aufgaben bewältigt. Allerdings besteht immernoch der Unterschied, dass man als Entwickler nur etwas zu dem Projekt beisteuert und als Architekt die Verantwortung dafür trägt.

11 Broadening the T

Oftmals wird die Softwarearchitektur als nicht-technische Karriere angesehen, in welcher Diagramme gezeichnet werden, welche das Softwaredesign bestimmen. Allerdings müssen Softwarearchitekten sich genauso gut mit den vorhandenen Technologien auskennen wie die Softwareentwickler. Die Softwarearchitekten müssen immer wissen können ob eine Lösung funktionieren wird und wie diese umgesetzt werden kann.

Oftmals spezialisieren sich Fachkräfte allerdings nur auf eine oder zwei Technologien und arbeiten nur mit diesen, obwohl es für gewisse Projekte sicherlich geeignetere und bessere Technologien gibt. Als Softwarearchitekt muss man deshalb immer den Blick für das Ganze haben und zumindest die Grundsätze von verschiedenen Technologien kennen und verstehen, um für ein Projekt schließlich die passenden und richtigen Technologien auszuwählen.

Häufig kommen Softwarearchitekten aus dem Bereich der Softwareentwicklung und haben somit bereits Erfahrung mit einigen Technologien gehabt. Somit können sie zwischen dem Blick auf das ganze Projekt und einem detaillierten Blick in die Applikation wechseln um Probleme zu erörtern. Da Softwarearchitekten nicht in allen Technologien Spezialisten sein können, ist es wichtig die unterschiedlichen Technologien, welche für eine Problemlösung eingesetzt werden können, zumindest zu kennen und andere Spezialisten zu Rate zu ziehen, mit welchen die Lösung im Detail durchgesprochen werden kann. Die Rolle des Softwarearchitekten kann also auch von mehreren Personen gleichzeitig wahrgenommen werden, um die bestmögliche Lösung zu finden.

12 Soft skills

Ein Softwarearchitekt benötigt nicht nur eine breite und teilweise tiefgreifende Kenntnis der verschiedenen Technologien. Da ein Softwarearchitekt eine leitende Rolle in einem Projekt übernimmt, benötigt er auch diverse Soft skills in diesem Bereich:

- Führungspotential: Der Softwarearchitekt muss die Fähigkeit besitzen sein Team zu einem Ziel zu führen.
- Kommunikation: Es muss eine Kommunikationsbasis geschaffen werden, auf welcher sich Softwareentwickler und Fachpersonal unterhalten können.
- Einfluss: Der Softwarearchitekt muss sein Team durchgehend motivieren indem er diverse Techniken wie psychologische Tricks anwendet.
- Überzeugung: Ohne Überzeugung kann ein Softwarearchitekt seine Pläne nicht durchsetzen.
- Zusammenarbeit: Ein Softwarearchitekt kann kein Spezialist für alle Technologien sein. Er muss sich beraten lassen und auch Kritik von anderen Personen verarbeiten können.
- Lehren: Wenn neue Technologien eingesetzt werden, hat der Softwarearchitekt die Aufgabe diese den Mitarbeitern näher zu bringen.
- Politik: Bei allen Entscheidungen muss die Geschäftspolitik berücksichtigt werden.
- Verantwortung: Schlägt ein Projekt fehl, so kann die Verantwortung nicht auf die Softwareentwickler abgewälzt werden. Die Softwarearchitektur muss stets die Geschäftsziele erfüllen können und dabei die nicht-funktionalen Anforderungen berücksichtigen.
- Delegation: Der Softwarearchitekt hat nicht die Aufgabe ein Projekt alleine fertigzustellen und muss folglich die Aufgaben an qualifiziertes Personal übergeben. Zu berücksichtigen ist jedoch, dass die Verantwortung weiterhin beim Architekten liegt.

Eine sehr wichtige Aufgabe für den Softwarearchitekten ist es, positiv zu bleiben. Alle Projektzugehörigen Softwareentwickler haben eigene Fähigkeiten bezüglich der Softwarearchitektur und betrachten jeden Schritt des Softwarearchitekten. Dieser muss somit versuchen die Kompetenzen der Entwickler komplett auszunutzen und somit die richtigen Entscheidungen zu treffen. Das Gemüt des Softwarearchitekten wirkt sich dabei direkt auf das Gemüt des ganzen Teams aus. Wenn der Architekt in seiner Führungsposition enthusiastisch arbeitet, werden die Softwareentwickler es ihm gleich tun, was dazu führt,

dass die Ziele des Projekts hochqualitativ erreicht werden, was den Softwarearchitekten wiederum enthusiastisch stimmt. Somit ist ein ewiger positiver Kreislauf geschaffen, welcher die Grundlage für eine gute Zusammenarbeit legt.

13 Software development is not a relay sport

Die Rolle des Softwarearchitekten kann im Idealfall durch die Zusammensetzung der verschiedenen spezialisierten Teammitglieder getragen werden. Allerdings haben alle Softwareentwickler zumindest auch Grundkenntnisse aus den anderen Bereichen, welche dazu führen, dass die Entscheidungen hinterfragt werden. Dies führt wiederrum zur Diskussion und verzögert eine Lösungsfindung. Aus diesem Grund wird der Einsatz eines Softwarearchitekten meist unumgänglich, sobald ein Projekt mehrere Technologien umfasst.

Häufig werden in größeren Firmen sogenannte Lösungsarchitekten eingesetzt. Diese planen ein Projekt, geben den Plan dann an die Entwickler weiter und planen ein neues Projekt. Sie sind nicht in die Entstehung der Lösung involviert und oftmals wird die vorgesehene Architektur dann von den Entwicklern angepasst. Hierdurch wird nicht nur die Verantwortung weitergeschoben sondern der Lerneffekt für den Lösungsarchitekten bleibt ebenfalls aus. Der Lösungsarchitekt könnte in jedem seiner Projekte den gleichen Fehler machen ohne dies mitzubekommen und jedes Entwicklungsteam entwickelt für das Problem eine eigene Lösung.

Die Aufgaben eines Softwarearchitekten beschränken sich nicht nur auf die Planung des Projekts, sondern auch auf die Überwachung der Durchführung sowie die Instandhaltung des Teams. Ein Softwarearchitekt sollte während der ganzen Entwicklungszeit eng mit dem Team zusammenarbeiten um nötige Änderungen an der Softwarearchitektur vornehmen zu können.

Ein Team von Softwareentwicklern, welches die Softwarearchitektur selbst durchführt hat dieses Problem zwar nicht, allerdings ist in diesem Fall auch keine Führungsrolle, welche die Verantwortung trägt, abgestimmt.

14 Software architecture introduces control?

Durch eine entsprechende Softwarearchitektur ist eine gute Struktur sowie Einsicht in die Vision des Projekts gegeben. Die Kernaufgaben des Architekten sollen für eine hohe Wartbarkeit bereits während der Entwicklung sorgen. Der schwierigste Teil ist die Abstimmung der Schnittstellen zwischen den verschiedenen Abteilungen. Um Mehraufwand zu vermeiden sollten die betroffenen Abteilungen gut instruiert sein.

Der Softwarearchitekt trägt die Verantwortung ein richtiges Maß an Kontrolle zu finden. Die Extreme sind dabei, dass entweder alle Entscheidungen über seinen Schreibtisch laufen oder dass alle Teammitarbeiter eigenständig Änderungen einbringen können.

Dabei gibt es einige Faktoren, welche beachtet werden müssen.

- Die Kultur des Landes.
- Die Erwartungshaltung der Softwareentwickler an den Softwarearchitekten.
- Die Erfahrenheit des Teams
- Die Teamgröße.
- Die Projektgröße
- Die Komplexität des Projekts.

Eine gute Vorgehensweise ist mit einer niedrigen Stufe der Kontrolle zu beginnen, daraufhin die Reaktionen der Teammitglieder zu bewerten und schließlich das Maß der Kontrolle anzupassen.

15 Mind the gap

Viele Unternehmen haben ein negatives Bild von der Rolle des Softwarearchitekten. Sie behaupten, dass die Architekten eiserne Vorgaben für ein Team setzen und nicht an das letztendlich funktionsfähige Softwareprodukt denken. Hierdurch wird die Reputation der IT geschadet und es entsteht eine Kluft zwischen den Unternehmen und den Softwarefachkräften.

Ein perfektes Szenario würde ein Team mit durchweg gleicher Erfahrung voraussetzen, welches auf allen Ebenen des Projekts eine gewisse Grundkenntnis zur Verfügung stellt. Da dieses Szenario allerdings utopisch ist, wird ein Softwarearchitekt benötigt, welcher das große Ganze im Blick hat. Somit können die zu erfüllenden nicht-funktionalen Anforderungen und die zu nutzenden Technologien von einer Stelle überwacht werden, während die Softwareentwickler in ihrem Spezialgebiet arbeiten können.

Softwarearchitekten werden oftmals auf eine Art Podest gehoben ohne das sie tatsächlich Aufgaben des Softwarearchitekten übernehmen, wodurch ebenfalls eine Kluft zwischen diesem und den Softwareentwicklern geschaffen wird. Hier übernimmt der Softwarearchitekt nur die Führungsaufgaben und arbeitet sonst als einfacher Softwareentwickler weiter. Dies führt dazu, dass die Entwickler die Entscheidungen des Architekten nicht respektieren und die Motivation sinkt.

Um diesem entgegenzuwirken ist es wichtig, dass sich die Rolle des Softwarearchitekten klar von der des Softwareentwicklers abhebt. Bezieht der Architekt die Entwickler zudem in den Architekturprozess mit ein, so wird schnell klar, dass sich die beiden Rollen grundlegend unterscheiden. Umgekehrt sollte der Architekt sich auch bei der Entwicklung beteiligen, um die Feinheiten der Architektur zu verstehen und auch ins Detail gehen zu können.

16 Where are the software architects of tomorrow?

Als Softwareentwickler hat man selten die Aufgabe ein Projekt für ein Team von Grund auf zu planen und zu unterstützen. Diese Aufgaben fallen normalerweise einem Softwarearchitekten zu. Die Überwachung eines Projekts und das Lehren und Leiten der Softwareentwickler während der Entwicklung sind Kernvoraussetzungen für einen guten Teamleiter. Häufig werden die besten Softwareentwickler in Managementrollen versetzt, da diese in der Karriereleiter aufsteigen wollen, während sie als Softwarearchitekt dem Team durch die Expertise einen größeren Nutzen bieten würden. Die somit entstandene Lücke muss daraufhin von anderen Entwicklern gefüllt werden, welche für diese Position nicht so qualifiziert sind. Durch die Umstrukturierung entsteht außerdem eine Unruhe in dem Team, die dazu führt, dass das Team keine Zeit mehr hat sich auf andere Dinge außer das Entwickeln zu konzentrieren.

Hieraus ergibt sich, dass Softwareteams die Möglichkeit haben müssen, eine gewisse Zeit andere Technologien zu begutachten und sich nicht direkt mit dem aktuellen Projekt zu beschäftigen.

Dadurch, dass viele erfahrene Softwareentwickler in Managementrollen gehen, fehlen die leitenden Personen, welche die nächsten Generationen anlernen können.

17 Everybody is an architect, except when they're not

Viele Software-Teams versuchen die Rolle des Softwarearchitekten untereinander aufzuteilen, wobei sich das Wiederfinden der genauen Rollendefinition als schwierig herausstellt.

Hierbei ist es essentiell, dass alle Mitarbeiter die Möglichkeit besitzen, Änderungen an allen Stellen des Projekts durchzuführen. Somit muss auch jeder Mitarbeiter das große Ganze im Kopf haben und sich über die Auswirkungen von Änderungen komplett bewusst sein.

Erst wenn jedes Mitglied eines Teams permanent dazu in der Lage ist, an jeder Baustelle des Projekts zu arbeiten und währenddessen alle nicht-funktionalen Anforderungen erfüllt sind, kann von einem funktionierenden, selbst organisierenden Team gesprochen werden. Hier wird ersichtlich, dass die Idee nicht perfekt umsetzbar ist und dass ein Mitarbeiter niemals in mehreren solcher Projekte gleichzeitig arbeiten könnte. Aus diesem Grund ist die Rolle des Softwarearchitekten, welcher die Verantwortung für ein Projekt übernimmt, unerlässlich.

Bei der agilen Softwareentwicklung wird häufig von einem Softwarearchitekten, welcher das große Ganze im Auge hat abgesehen, da dieses zu Beginn der Entwicklung noch sehr abstrakt ist. Das Problem dabei ist, dass die Entwicklung schnell sehr chaotisch werden kann, wenn niemand darauf achtet wie der Rahmen abgesteckt sein muss um ein Ziel zu erreichen. Es sollte bei einem agilen Projekt nicht ein zu großes Konzept vorgegeben werden, da dieses die Agilität einschränkt. Dies ist allerdings auch nicht die Aufgabe des Softwarearchitekten.

18 Software architecture as a consultant

Als Softwarearchitekt ist es essentiell, dass man sich Wissen in dem Fachbereich des Projekts aneignet und die Komplexität erfasst. Dieses Wissen eignet man sich durch Arbeiten in den Fachbereichen an. Als beratende Firma ist es allerdings nicht möglich, sich in allen Fachbereichen auskennen. Ein Lösungsansatz ist, dass man sich dazu entschließt, nur Firmen aus einem Fachbereich zu beraten. Hierbei verbaut man sich allerdings die Möglichkeiten in andere Firmen zu wechseln. Ein anderer Lösungsansatz setzt starke analytische Fähigkeiten voraus, die es dem Softwarearchitekten ermöglichen, schnell Fachwissen anzusammeln.

Eine weitere Schwierigkeit stellen die Mitarbeiter der zu beratenden Firma dar. Diese müssen unter der Leitung des beratenden Softwarearchitekten arbeiten. Hierzu muss der Softwarearchitekt eine gewisse Autorität ausstrahlen und von der Firma zugewiesen bekommen. Auch entsprechende Soft skills sind dazu notwendig.

20 Architectural drivers

Im Folgenden werden die einzelnen Bestandteile eines Projekts angesprochen, die den Einsatz eines Softwarearchitekten erfordern.

Zu Beginn eines Projekts müssen die funktionalen Anforderungen bestimmt werden, welche zur Erfüllung des Ziels nötig sind. Oft werden diese bei der agilen Entwicklung außer Acht gelassen, was allerdings schnell dazu führen kann, dass das Projekt immer größer wird, da kein Rahmen abgesteckt wurde.

Des Weiteren müssen die nicht-funktionalen Anforderungen wie Performanz, Sicherheit und Verfügbarkeit mit einbezogen werden. Diese meist sehr technischen Qualitätsmerkmale sind schwierig zu messen und zu erreichen und bedürfen eines hohen Zeitaufwands. Um sie zu prüfen muss das gesamte Projekt und nicht nur Teile davon betrachtet werden.

Die Entwicklung eines Projekts wird durch Einschränkungen der zu nutzenden Technologien erschwert. Bei Problemen müssen also entweder andere passendere Technologien gefunden oder eine Lösung für das Problem gefunden werden.

Wichtig ist, dass sich an Prinzipien wie Code-Konventionen gehalten wird. Die Überwachung und die Festlegung dieser Prinzipien obliegt dem Softwarearchitekten.

Bei einem neuen Projekt sollte versucht werden, die bisher genannten Punkte aus vorigen Projekten zu übernehmen. Die bereits gemachten Erfahrungen sind hierbei immer von Vorteil. Es kann beispielsweise von vornherein festgestellt werden ob eine Technologie alle Anforderungen erfüllen kann.

21 Quality Attributes (non-functional requirements)

Nicht-funktionale Anforderungen stellen die Qualitätsmerkmale eines Produkts dar. Diese sind bei fast allen Projekten weitestgehend identisch, können allerdings unterschiedliche Ausprägungen haben. Im Folgenden sind einige der gängigsten nicht-funktionalen Anforderungen aufgelistet.

Performanz

Mit Performanz beschreibt eine möglichst niedrige Wartezeit zwischen einem auslösenden Ereignis (bspw. das Klicken einer Schaltfläche) und der Visualisierung des Ergebnisses (bspw. Darstellung einer neuen Seite). Sollten die Nutzer einer Software sich über zu lange Wartezeiten oder ein zu langsames System beschweren, so sollte versucht werden diesem mit technischen Mitteln oder durch Änderung der Software entgegenzuwirken.

Skalierbarkeit

Wenn beispielsweise eine App auf den Markt kommt, so wird sie zu Beginn voraussichtlich wenig Nutzer haben. Wenn die Zahl der Nutzer dann steigt muss es einfach sein, das System zu erweitern, sodass es den aufkommenden Datenverkehr weiterhin bewältigen kann.

Verfügbarkeit

Die Verfügbarkeit gibt an zu wie viel Prozent der Laufzeit eine Softwarelösung nutzbar ist. Um eine hohe Verfügbarkeit zu erreichen können beispielsweise redundante Systeme eingesetzt werden, welche sich beispielsweise bei einem Stromausfall ersetzen können. Eine Verfügbarkeit von 100% ist hierbei unmöglich. Meist sind es 99% mit einer variierenden Zahl von neunen als Nachkommastelle.

Sicherheit

Sicherheit von Authentifizierung bis hin zum Datenschutz wird unter diesem Begriff zusammengefasst. Um sich vor den gängigen Sicherheitslücken schützen zu können kann man sich bei dem “Open Web Application Security Project (OWASP)” erkundigen.

Desaster Wiederherstellung

Hierbei geht es um die Möglichkeit Daten wiederherstellen zu können, falls ein großes Unglück wie beispielsweise ein Brand eingetreten ist. Auch hierbei ist es wie bei der Verfügbarkeit möglich, durch redundante Datenhaltungssysteme einen gewissen Schutz zu bieten.

Barrierefreiheit

Man sollte das gesamte Spektrum der möglichen Nutzer erreichen können. Somit muss auch die Bedienbarkeit der Software von Personen mit Einschränkungen gegeben sein. Hierbei können die “W3C accessibility standards” einen guten Einstieg liefern.

Wartbarkeit

Wartbarkeit bedeutet die Lesbarkeit und Weiterverwendung des geschriebenen Codes. Diese ist schwierig zu messen und hängt stark von Prinzipien wie Code-Konventionen ab.

Rechtliche Einschränkungen

Man sollte sich darüber im Klaren sein, an welchen Orten eine Anwendung eingesetzt werden kann. Handelt es sich beispielsweise um eine global abrufbare App, so muss sie auch den Gesetzen der Länder entsprechen.

Internationalisierung

Softwarelösungen sollten immer mindestens die englische Sprache anbieten, um auf der ganzen Welt nutzbar zu sein. Bei einer beauftragten Softwarelösung für ein kleines Unternehmen ist dies meist weniger sinnvoll.

22 Working with non-functional requirements

Häufig werden Softwaredienstleistern bei einer Beauftragung keine nicht-funktionalen Anforderungen genannt. Diese müssen dann eigenständig festgelegt und evaluiert werden. Würde man den Auftraggeber beispielsweise nach Verfügbarkeit fragen, so würde dieser sicherlich 100% fordern.

Wenn ein Unternehmen also keine nicht-funktionalen Anforderungen mitgibt, beziehungsweise nur sehr wage formuliert, so sollte man diese durch geschickte Gegenfragen evaluieren. Ein Beispiel wäre die Frage nach der akzeptablen Ausfallzeit im Gegensatz zu der geforderten Verfügbarkeit.

Die nicht-funktionalen Anforderungen müssen immer im Zusammenhang mit der Aufgabe der Softwarelösung betrachtet werden um zu aussagekräftigen Formulierungen zu kommen.

Sollte eine unerreichbare Form einer nicht-funktionalen Anforderung gestellt werden, wie beispielsweise 100% Verfügbarkeit, so sollte man eine Gegenüberstellung von beispielsweise 99% mit unterschiedlichen Preisangaben machen. In diesem Fall entscheidet sich der Kunde im Normalfall für die sehr viel günstigere Alternative.

23 Constraints

Begrenzungen sind in der Realität unumgänglich. Diese können sich beispielsweise in Form von Zeit- oder Kostenbegrenzungen bemerkbar machen. Diesen Begrenzungen ist nur durch Kommunikation mit dem Auftraggeber entgegenzuwirken.

Technologische Begrenzungen hingegen können unter Umständen umgangen werden. Hierzu kann eine betriebsinterne Liste von bewährten Technologien zu Rate gezogen werden. In bereits fertiggestellten Projekten können diese Begrenzungen vielleicht schon aufgetreten sein, wodurch sich eine Technologie für ein neues Projekt als nicht ideal herausstellt.

Oftmals entstehen die Begrenzungen auch durch Kundenwünsche. Beispielsweise wird häufig der Gebrauch von OpenSource Möglichkeiten als zu riskant angesehen, solange diese Technologien keinen etablierten Namen haben.

Begrenzungen können natürlich auch in Form der Mitarbeiter auftreten. Beispielsweise ist es möglich, dass man durch eine zu geringe Mitarbeiterzahl Projekte nicht bewältigen kann. Möglicherweise muss auch erst ein Training angesetzt werden, damit die Mitarbeiter eine neue Technologie erlernen können.

Durch Begrenzungen wird der Softwareentwickler zu einem Umdenken gezwungen, welches dazu führen kann, dass dieser eine neue Technologie kennen lernt, welche er nachfolgend immer benutzt, da diese seine Anforderungen besser erfüllt. Wichtig ist, dass die Begrenzungen immer in die Planung eines neuen Projekts einbezogen werden.

24 Principles

Um eine hohe Konsistenz zu gewährleisten ist es wichtig, *Prinzipien* zu einzuführen um standardisierte Herangehensweisen bei der Entwicklung von Software zu verwenden. Prinzipien lassen sich in Bezug zur Entwicklung und Architektur einteilen.

Entwicklungs-Prinzipien

Für die Entwicklung von Software gibt es unter anderem die nachfolgenden Prinzipien:

- Coding Standards und Konventionen
- Automatisiertes Unit-Testing
- Statische Analyse Tools
- etc.

Architektur Prinzipien

Nachfolgend werden einige Prinzipien vorgestellt, welche sich auf die Struktur von Software beziehen.

- **Schichtenstrategie:** Eine Schichtenstrategie bzw. eine Schichtenarchitektur wird realisiert, in dem die zu entwickelnde Software in voneinander isolierte, Schichten eingeteilt wird. Diese können beispielsweise einen fachlichen Zweck erfüllen. Dadurch wird eine hohe Flexibilität gewährleistet, denn Änderungen bzw. Updates können nur Softwarekomponenten innerhalb einer Schicht beeinflussen.
- **Plazierung der Geschäftslogik:** Aus Performance und Wartungsgründen ist es oft sinnvoll, wenn die Geschäftslogik auf einem Server ausgeführt wird. Im Allgemeinen ist es von Vorteil, wenn so viel Rechenlast wie möglich auf einen Server ausgelagert wird anstatt Performance schwache Clients o. ä. zu beanspruchen.
- **Hohe Kohäsion, lose Kopplung, SOLID, etc.**
- **Zustandslose Komponenten:** Komponenten, welche keine festen Zustände einnehmen, können besser skaliert werden. Die Replikation einer Komponente mit dem Ziel der Rechenlastverteilung ist ein Beispiel der Skalierung.
- **Domain model - rich vs anaemic:** Manche Entwickler bevorzugen einen Programmierstil, mit dem Domain models als Funktionsreiche Objekte entwickelt werden. Beispielsweise eine OOP-Klasse mit fachlichen Daten und allen Funktionen, die mit diesen Daten einhergehen. Andere Entwickler wiederum bevorzugen es, reine Datenobjekte zu verwenden welche von Grobkörnigen Komponenten und Services

verwendet werden.

- **Immer konsistent vs eventuell konsistent:** Viele Entwickler haben festgestellt, dass die Erfüllung von vielen nicht-funktionalen Anforderungen die Konsistenz leidet. Beispielsweise erhöht sich in manchen Fällen die Performance, wenn man Dateninkonsistenz in Kauf nimmt.

25 Technology is not an implementation detail

Technologien, beispielsweise Programmiersprachen, können auch Bestandteil bei der Entwicklung von Software-Architekturen sein und sollten berücksichtigt werden.

Insbesondere bei komplexen nicht-funktionalen Anforderungen können die Technologien eine große Rolle spielen und müssen bei der Architektur berücksichtigt werden. Des Weiteren gibt es in vielen Entwicklerteams constraints hinsichtlich der Fähigkeiten der Entwickler.

26 More layers = more complexity

Die Anzahl der Softwareschichten sollte stets begutachtet werden. Große, komplexe und verteilte Anwendungen werden oftmals in viele Schichten unterteilt. Jede neue Schicht kann Auswirkungen auf die Logik anderer Schichten haben, welche alle berücksichtigt werden müssen. Viele Schichten gehen mit einer hohen Komplexität einher.

27 Collaborative design can help and hinder

In vielen Fällen besteht ein Entwicklungsteam aus Spezialisten in verschiedenen Bereichen: Web Frontend-, Backend-, Database-Developer und mehr. Durch viele Erfahrungen in allen Bereichen einer Fullstack Anwendung sollte eine gute Anwendung entstehen. Die Effektivität eines solchen kollaborativen Vorgehens hängt jedoch von vielen Faktoren ab. Für die Implementierung eines neuen Features können die Spezialisten aus den unterschiedlichen Bereichen des Stacks vollkommen unterschiedliche Meinungen haben. Erfahrung spielt oftmals eine große Rolle.

Kollaborative Herangehensweisen tragen zu einem effektiven Entwicklungsprozess bei. Mitglieder eines Entwicklungsteams haben unterschiedliche Meinungen, welche begutachtet werden müssen.

28 Software architecture is a platform for conversation

Kommunikation ist eines der entscheidenden Elemente in der Softwareentwicklung. Nicht nur innerhalb des Entwicklungsteams, sondern auch mit dem Kunden, Endanwendern und anderen Stakeholdern. Die meisten Probleme und Fragestellungen lassen sich in Verbindung mit anderen Menschen lösen. Neue Gesichtspunkte können hervorgehen, welche auch von anderen begutachtet werden müssen. Anstatt isoliert zu arbeiten ist es wichtig, sich regelmäßig auszutauschen und über Probleme zu sprechen. Kommunikationsfähigkeit ist ein zentrales Element für eine effektive Entwicklung.

30 The conflict between agile and architecture - myth or reality?

Ein häufiges Missverständnis ist, dass sich ein agiles Vorgehen mit der Verwendung von Software-Architekturen nicht vertragen. Das Agile Manifest sieht es im allgemein vor, auf Änderungen zu reagieren anstatt einen konkreten Plan zu verfolgen. Allerdings brauchen agile Projekte, sowie i.d.R. alle Softwareprojekte, eine Architektur. Aus diesem Grund gibt es auch keinen Konflikt zwischen "agil" und "Architektur".

Architektur kann, zusammengefasst, als Struktur und Vision des Projekt verstanden werden. Ein Hauptaugenmerk ist es, signifikante Design Entscheidungen zu verstehen.

Viele Teams, welche ein agiles Vorgehn umsetzen, machen den Fehler, sich zu Beginn auf garkein Design festzulegen und das große Ganze nicht zu analysieren. Stattdessen wird stetig auf Veränderungen reagiert, was zu einem anhaltenden Zyklus aus refactoring führt.

Ein Software-Architekt definiert Grenzen der Entwicklung. Innerhalb dieser Grenzen können beliebige Entwicklungsverfahren verwendet werden. Ein Software-Architekt kümmert sich letztendlich darum, eine Art Framework für die Entwicklung zu erzeugen und Grenzen aufzuzeigen.

31 Quantifying risk

Risiken zu identifizieren ist ein wichtiger Bestandteil des Softwaredesigns. Einige Risiken sind unterschiedlich wichtig. Aus diesem Grund müssen alle Risiken klassifiziert und Prioritäten zugeordnet werden.

Ein möglicher Weg Risiken zu klassifizieren ist, deren Wahrscheinlichkeit für das Eintreten des Risikos sowie dessen schwere der Auswirkung zu untersuchen. Risiken können somit in einer *Probability-Impact* Matrix verteilt werden. Die Wahrscheinlichkeit des eintretens und die schwere der Auswirkung können beispielsweise in *Low*, *Medium* und *High* eingeteilt werden. Nach dieser Matrix können anschließend den Risiken Prioritäten auferlegt werden. Durch die Priorisierung können sich Entwickler auf bestimmte Risiken fokussieren.

32 Risk-storming

Risk-storming ist eine Technik zur Identifizierung von Risiken, welche in vier Schritte eingeteilt werden kann:

1. **Architektur Diagramme zeichen:** Mit Architektur Diagrammen lassen sich Risiken auf verschiedenen Abstraktionsebenen der Software hervorheben.
2. **Identifizierte die Risiken individuell:** Mitglieder des Entwicklungsteams müssen einzeln Risiken finden und aufschreiben. Dadurch werden Risiken durch unterschiedliche Sichten von Spezialisten sichtbar. Die Risiken müssen vom Team, z. B. einer Probability-Impact Matrix, klassifiziert werden.
3. **Risiken auf den Diagrammen zusammenführen:** Die identifizierten Risiken müssen nun auf den Architektur Diagrammen, z. B. mit Klebezetteln, an entsprechenden Komponenten plaziert werden. Dadurch wird visualisiert, an welchen Stellen die höchsten Risiken auftreten können.
4. **Priorisiere die Risiken:** Abschließend werden den Risiken von dem Team Prioritäten zugeordnet.

33 Just enough up front design

In vielen Szenarien wird entweder versucht alle Aspekte der Software-Architektur vor Beginn der Entwicklung zu behandeln oder die Software-Architektur während der Entwicklung zu erstellen. Eine bewährte Methode ist, *gerade genug* zu tun. Wie viel *gerade genug* ist, hängt von vielen Faktoren ab. Ein möglicher Ansatz ist es, sich mit der Architektur vor der Entwicklung solange zu beschäftigen, bis die Struktur und Vision deutlich ist. Letztendlich hängt es stark von dem Entwicklerteam und deren Erfahrungen ab.

Die nachfolgenden Grundsätze können eine gute Grundlage bilden um gerade genug Softwaredesign zu erarbeiten.

1. Struktur

- **Was:** Verstehe die Signifikanz der strukturellen Elemente und wie diese zusammenhängen.
- **Wie:** Designe und zerlege Elemente bis runter zu Softwarekomponenten.

2. Vision

- **Was:** Erstelle und kommuniziere eine Vision, mit der das Entwicklungsteam arbeiten kann.
- **Wie:** Visualisiere die Architektur mit verschiedenen Diagrammen, um verschiedene Abstraktionsebenen darzustellen.

3. Risiken

- **Was:** Identifiziere und behandle die größten Risiken.
- **Wie:** Risk-storming und Experimente.

34 Agility

Ohne philosophisch zu werden, müssen es gute Software-Architekturen erlauben, agile Vorgehensweisen zuzulassen. Agile Vorgehensweisen können auf Veränderungen reagieren und die sich immer-verändernden Anforderungen adaptieren.

35 Introducing software architecture

Die Einführung von Software-Architekturen hat mit ein wenig Disziplin großes Potential für den Erfolg von Softwareteams.

Eines der größten Probleme ist es, dass Software-Architektur in Konkurrenz mit vielen anderen neuen Vorgehensweisen und Technologien stehen, welche täglich erscheinen. Viele Entwickler denken über Architektur nicht so viel nach, wie sie es tun sollten.

Es gibt einige praktische Vorschläge, wie man Software Architektur anderen Entwicklern näher bringen kann.

1. Leute Ausbilden:

Beispielsweise können mit einfachen Workshops den Entwicklern Software-Architektur beigebracht werden.

2. Rede über Architekturen im Rückblick:

In regelmäßigen Besprechungen in denen Projekte reflektiert werden, können Architekturen angesprochen werden.

3. Definition of done:

Für Arbeitseinheiten gibt es i. d. R. Definitionen, wann diese abgeschlossen sind.

Software-Architektur sollte in die Liste der Aufgabeneinheiten aufgenommen werden.

4. Jemanden den die Rolle des Software Architekten geben.

Enterprise Achritecture at Work

Zusammenfassung des Buches:

Titel: Enterprise Architecture at Work - Modelling, Communication and Analysis - Fourth Edition

Verfasser: Marc Lankhorst

Verlag: Springer

Jahr: 2017

ISBN: 978-3-662-53933-0

Zusammenfassung von: Marcel Dzaak, Andrei Günter, Marvin Schirrmacher

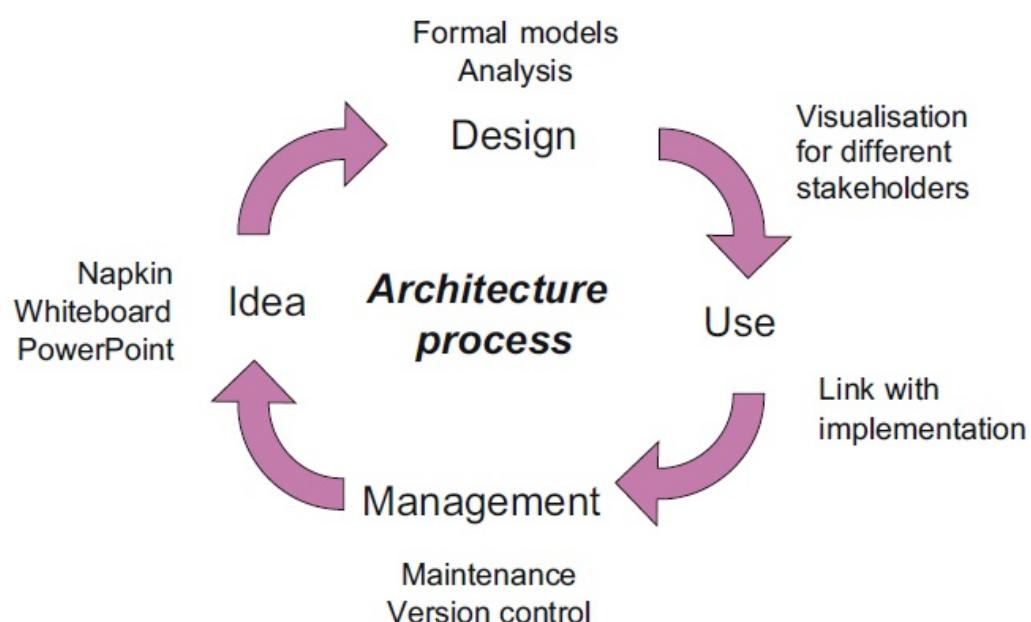
Introduction to Enterprise Architecture

Architecture

Wie im Bauwesen wird auch für den Aufbau eines Unternehmens eine Architektur benötigt. Auch wenn der Begriff Architektur mehrdeutig ist könnte man Architektur als ein fundamentales Konzepts beschreiben, welches auch eine Vision beinhaltet.

Enterprise Architecture & Process

Definition Enterprise: Any collection of organisations that has a common set of goals and/or a single bottom line. Definition Enterprise Architecture: A coherent whole of principles, methods and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems and infrastructure. Eine Enterprise Architecture sollte möglichst flexibel sein und trotzdem sollte es möglich sein die essentiellen Geschäftsmodelle verstehen zu können. Für die Erstellung einer Architektur, ist viel Kommunikation mit unterschiedlichsten Stakeholdern von Nöten. Das folgende Bild veranschaulicht den life cycle einer Architektur (vgl. Buch:Enterprise Architecture at work).



Drivers

Internal Drivers: Effektivität, Business-IT alignment, Grundlage für unternehmensweite Entscheidungen, Komplexitätskontrolle External Drivers: Transparenter Überblick, sodass die Einhaltung von Gesetzen schnell überprüft werden kann. Schnelle Durchführung von Audits.

State of the Art

Enterprise architecture and other Governance Instruments

Enterprise architecture ist ein Teil von mehreren Management Praktiken. Der Zusammenhang, zu einigen ausgewählten Praktiken wird im Folgenden hergestellt.

Strategic Management - Balanced Score Card (BSC)

Fokussiert sich neben der finanziellen Perspektive auf drei weitere Perspektiven: „Customer“, „Internal Business Processes“ und „Learning and Growth“. Metriken, welche aus diesen Perspektiven gewonnen werden, können anhand einer enterprise architecture gespiegelt werden. Auswirkungen und Chancen können hierdurch besser eingeschätzt werden.

Business model development – Business Model Canvas

Liefert abstrakte Beschreibungen eines Unternehmensmodells. Durch eine enterprise architecture können diese Beschreibungen detailliert werden.

Business Architecture – BIZBOK Guide and O-BA

Definition Business Architecture: a blueprint of the enterprise that provides a common understanding of the organisation and is used to align strategic objectives and tactical demands. Business Architecture beantwortet die Frage nach dem „was?“ mit dem Focus auf „capabilities“ des Unternehmens. Je nach Unternehmen wird eine Business Architecture gleichwertig zu einer Enterprise Architecture gestellt. Andere Unternehmen sehen die Business Architecture als einen Teil der Enterprise Architecture und dort im „high-level“ Bereich.

Quality management – EFQM and ISO 9001

Der beschriebene BSC hilft dabei strategische Entscheidungen zu treffen. EFQM unterstützt bei der kontinuierlichen Verbesserung, um diese bestimmte Strategie ausführen zu können. EFQM model besteht aus 9 Kriterien, fünf sind „enablers“ (was tut ein Unternehmen), vier sind „results“ (was erreicht ein Unternehmen). EFQM model ist ein management framework

um neben dem Qualitätsmanagement auch die Leistungsfähigkeit eines Unternehmens zu steigern. EFQM model ist vom Umfang her größer als die ISO 9001. Die ISO 9001 enthält Kriterien für ein gutes „quality management system“ (QMS). Der Standard umfasst Anforderungen vom Planen bis zur Ausführung, sowie Messtechniken und Verbesserungen.

IT governance – COBIT

COBIT ist ein „IT control framework“, welches Praktiken zur Verfügung stellt, um „IT governance“ zu implementieren. Es werden Prozesse und „Control Objectives“ verwendet um Aufgaben der IT zu beschreiben. Eine enterprise archtecture focusiert sich auf die primären Unternehmens und IT Strukturen. COBIT wird angewendet, um zu beschreiben wie die sekundären IT-Funktionen einer Organisation organisiert sein sollten.

IT delivery and support – ITIL

ITIL (IT Infrastructure Library) ist das am besten akzeptierteste Bündel an “best practices” im Bereich “IT service delivery” ein “de facto standard” for IT service management. ITIL bietet praxisorientierte Lösungen um in Organisationen “quality management” zu implementieren. Hilfestellungen gibt es von der Planung bis zur Implementation von Prozessen. Die Hauptbestandteile sind: Service Delivery und Service Support. ITIL ergänzt die „high level control objectives“ von COBIT (was) und kann für deren Implementierung (wie) verwendet werden. ITIL wird in Bezug auf eine enterprise architecture für das Managen von IT Teilen verwendet, dort wo eine enterprise architecture häufig Schwächen aufweist.

IT implementation: CMM & CMMI

Das “Capability Maturity Model” (CMM / SW CMM) wird dazu verwendet, den Reifegrad bspw. eines „software engineering processes“ zu beurteilen und beinhaltet Praktiken diesen zu steigern. Ausgeweitet auf die Integration und nun den kompletten „product life cylce“ umfassend sprechen wir von der „Capability Maturity Model Integration“ (CMMI). In der meist verbreitetsten Form gibt es fünf Reifegrade (maturity levels): Initial (Prozesse sind chaotisch), Managed (Anforderungen fließen in den Prozess ein), Defined (Prozesse sind als Standard beschrieben), Quantitatively Managed, Optimising (Kontinuierliche Prozessverbesserung). Enterprise architecture wird ab CMMI Level 3 hilfreich, wenn nicht sogar notwendig, da Projekte organisationsweit standardisiert ablaufen müssen.

Architecture Methods and Frameworks

Um Anforderungen oder Änderungen jeglicher Art methodisch durch das Unternehmen tragen zu können, sind einige Methoden und Frameworks erstellt worden, um Architekten bei diesem Ziel zu unterstützen.

The IEEE 1471-2000/ISO/IEC 42010 Standard

Solide Basis für Definitionen, Analysen und Beschreibungen einer Systemarchitektur, mit dem Focus auf softwarelastige Systeme. Sehr ähnlich zu Zachman's framework, allerdings gibt dieser Standard mehr die Richtung durch vorhandene Best Practises vor. Der Standard besteht aus Definitionen (what is a architect, system, etc.), einem konzeptionellen Framework und sechs Praktiken zur Beschreibung einer Architektur. Das Framework hilft dabei die Relationen zwischen den einzelnen Bestandteilen zu erklären, die Rolle des Stakeholders zu verstehen und gibt einige mögliche Szenarien an die Hand. Die sechs Praktiken beziehen sich auf die Dokumentation einer Architektur, Stakeholder Identifizierung, Auswahl von Architektursichten, Auswahl von Sub-Architektursichten, Konsistenz zwischen den Sichten, Begründung der verwendeten Architekturen.

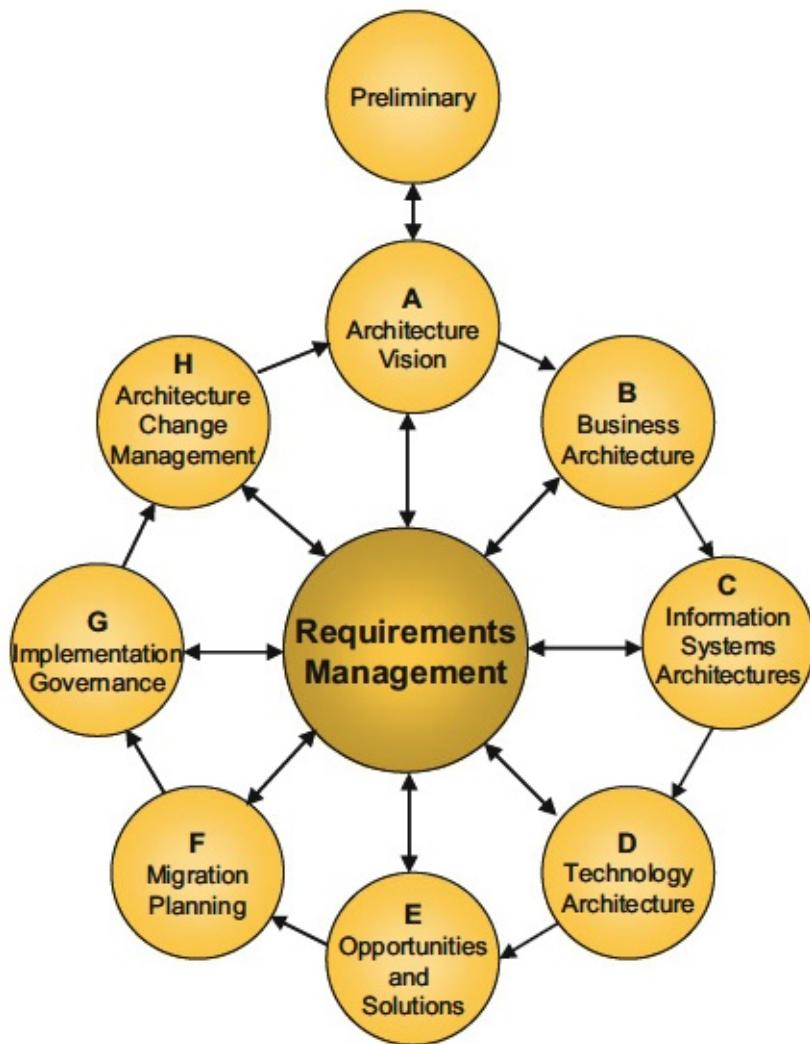
The Zachman Framework

Dient dazu logische Strukturen zu erstellen um ein Unternehmen zu klassifizieren und zu organisieren.

Vorteile des Frameworks sind die Betrachtung des gesamten Unternehmens, die Einfachheit des Frameworks und die Unabhängigkeit von Tools und Methodiken. Nachteile sind die große Anzahl an Zellen und die Relationen zwischen den Zellen sind nicht gut spezifiziert.

The Open Group Architecture Framework - TOGAF

Die Hauptkomponenten sind ein Capability Framework (skills, roles), die "Architecture Development Method (ADM)" (Zyklus zur Entwicklung einer EA), das "Architecture Content Framework" (Top-Level EA) und das "Enterprise Continuum" (Sammlung verschiedener Modelle). Das im folgenden dargestellte ADM von TOGAF ist iterativ über den gesamten Prozess zu sehen (vgl. Buch:Enterprise Architecture at work).



Description Languages

In der Software Entwicklung ist UML die führende Modellierungssprache. Für die Modellierung von Organisationen und Prozessen gibt es eine Vielzahl an Beschreibungssprachen. IDEF: Der Schwerpunkt liegt in der funktionalen Modellierung, Prozessmodellierung, Datenmodellierung Vorteile: Unterstützt die Modellierung einer Vielzahl von architektonischen Views. Nachteile: Keine Kommunikationsmechanismen zwischen den einzelnen Modellen. Anwendungsbereich: Industrie BPMN: The Business Process Modelling Notation (BPMN) ist auf Prozessmodellierung beschränkt. Applikationen und Infrastruktur wird nicht abgedeckt. UML: Hauptanwendungsbereich ist das designen von Systemen. Ohne technische Details kann UML auch für Visualisierungen von "business engineers" und "organisation specialists" verwendet werden. Das Konzept von UML, das es immer ein Object gibt, welches variable verwendet werden kann macht dies möglich.

Foundations

Getting to Grips with Architectural Complexity

Das eine EA notwendig ist, haben die meistßen Unternehmen mittlerweile erkannt, dennoch fehlt es häufig in den Bereichen Design, Communication, Realisation und Management an "einfachen" Lösungen. Würde es hierfür jedoch einfach Lösungen geben, würden wir eine EA nicht benötigen. In derzeitigen Praktiken werden häufig sehr viele verschiedene Modelle und Beschreibungen erstellt, allerdings ohne jegliche Integration in andere Modelle, getrieben durch verschiedenste Stakeholder und deren Bedürfnissen. In Integrationsfragen gibt es für eine Probleme Standards, für andere jedoch nicht.

Describing Enterprise Architectures

In einer EA werden keine Details beschrieben, es wird sich auf die Essentiellen Dinge konzentriert, auf die Zusammenhänge und die Integration. Hierfür werden "models" verwendet. Definition model: A model is an abstract and unambiguous conception of something (in real world) that focuses on specific aspects or elements and abstracts from other elements, based on the purpose for which the model is created.

Stakeholder werden sehr stark von Ihren individuellen Interessen getrieben, wenn diese Architekturen beschreiben oder versuchen zu erstellen. Der Architekt wird bei der Modellierung einer Architektur ebenfalls stark von diesen Interessen getrieben, welche sich stark in der Architektur wieder finden werden. Darüber hinaus wird der Architekt versuchen diese Interessen durch gelernte Konzepte auszudrücken.

Definition Domain: Any subset of a conception (being a set of elements) of the universe that is conceived of as being some 'part' or 'aspect' of the universe. Definition Modelling: The act of purposely abstracting a model from (what is conceived to be) a part of the universe. Der Begriff 'universe' kann als Platzhalter für alles verwendet werden. Je nach Kontext kann das Univerum, also das Große Ganze das Unternehmen sein.

Bereits erwähnt gibt es mehrere Sichten (Views), welche immer von einem individuellen Standpunkt (Viewpoint) aus beschrieben werden. Einfach ausgedrückt, eine View ist was jemand sieht und ein viewpoint sagt aus, von wo derjenige das zu Beschreibende betrachtet. Definition View: expresses the architecture of the system of interest from the

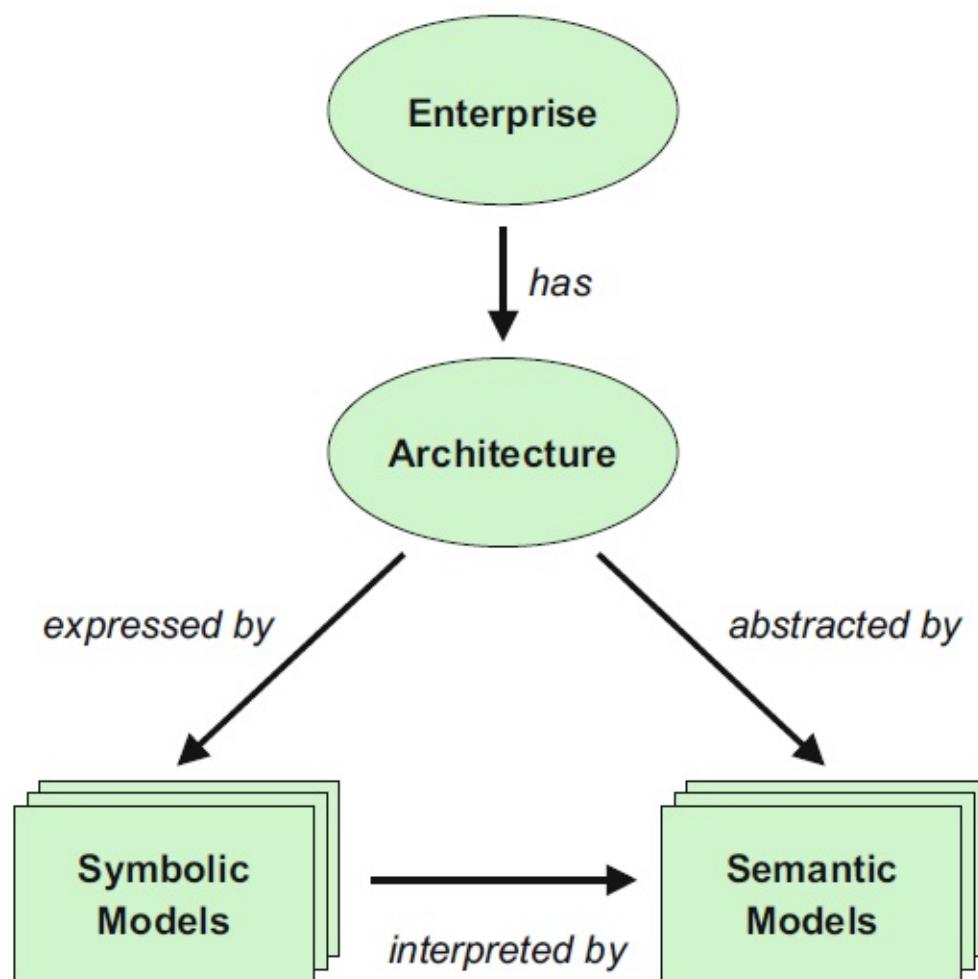
perspective of one or more stakeholders to address specific concerns, using the conventions established by its viewpoint Definition Viewpoint: a specification of the conventions for constructing, interpreting, using and analysing one type of architecture view

Pictures, Models, and Semantics

Im Zusammenhang mit Architektur werden viele verschiedene Sichten erzeugt, die meistens visualisiert werden. Im Folgenden wird versucht den Inhalt unabhängig von der Visualisierung zu sehen. Der Hintergrund ist, dass Visualisierungen häufig Dinge enthalten, die das "model" erweitern und erklären sollen. Es gibt immer etwas in einer Architektur, was unabhängig von der Visualisierung ist, die "Semantic". Die Semantic muss nicht in allen Views übereinstimmen, aber alle Semantiken haben immer etwas gemeinsam.

Symbolic and Semantic Models

Um die Idee zu erläutern, unterscheiden wir zwischen einem "symbolic model" und einem "semantic model". Definition symbolic model: expresses properties of architectures of systems by means of symbols that refer to reality. Definition semantic model: is an interpretation of a symbolic model, expressing the meaning of the symbols in that model (vgl. Buch:Enterprise Architecture at work).



Communication of Enterprise Architectures

Beispiele in denen Architekturbeschreibungen verwendet werden können:
-Analyse von alternativen Architekturen
-Kommunikation zwischen Organisationseinheiten
-Input für unterlagerte Architekturen
-Planung und Budgetierung

Bei all diesen Punkten spielt Kommunikation eine essentielle Rolle. Weiter gedacht ist "system development" ein Wissenstransformationsprozess, wobei die Kommunikation das Mittel zum Zweck darstellt. Definition System development community: a group of objects, such as actors and representations, which are involved in the development of a system. Definition Concern: an interest of a stakeholder with regards to the architecture description of some system, resulting from the stakeholder's goals, and the present or future role(s) played by the system in relation to these goals.

Einige Beispiele für "Concerns" sind:
-Die Anforderungen eines Stakeholders - Verbesserungen, welche durch die Einführung eines neuen Systems greifen.

Das Wissen, welches durch eine "system development community" erarbeitet und ausgetauscht wird, lässt sich in folgende Kategorien strukturieren: Perspektive: Systeme können aus verschiedenen Blickwinkeln betrachtet werden. Gleichbedeutend mit Viewpoint. Scope: Reichweite (Bspw. enterprise-wide, department-wide, etc.) Design chain: Besteht aus Purpose, Functionality, Design, Quality, Costs. Weiterhin gibt es noch "Historical perspective" und "abstraction level".

Explicitness of Knowledge

In der Umgebung des "knowledge managements" gibt es eine grundlegende Unterscheidung von Wissen, zum einen in "explicit knowledge" und "tacit knowledge". "Explicit knowledge" kann sehr gut dargestellt werden, bspw. durch eine Architektur. (Schwerpunkt des Buches) "Tacit knowledge" hingegen kann schlecht durch reine Theorie verstanden werden. Bspw. das Fahrradfahren muss praktisch erlernt werden durch try and error.

Aufgrund dieser Klassifizierung können für "explicit knowledge" weitere Faktoren aufgestellt werden: Formality: Formelle oder informelle Sprache Quantifiability: Ausgedrückt durch Kapazität, Aufwand, Ressourcen, etc. Executability: Ausführbarkeit von Wissen durch

Simulation, einen Prototypen, Animation, etc. Comprehensibility: Verwendung von Präsentationstechniken zur schrittweisen Verdeutlichung des Wissens. Completeness: Die Repräsentation des Wissens kann "complete", "incomplete" oder "overcomplete" sein.

Transformation of Knowledge

Durch das Lernen, Experimentieren, Weiterentwickeln wird immer neues Wissen aufgebaut und weiterentwickelt. Dieses Wissen wird in einer Gemeinschaft weitergegeben. Hierfür gibt es drei Kategorien: -Aware: Eine Person wird sich der Verfügbarkeit von Wissen gewahr durch die Wissensteilung durch eine andere Person. -Agreed: Nachdem das Wissen geteilt wurde, kann die Person entscheiden, ob sie dieses Wissen annimmt und zustimmt (agree) oder nicht. -Committed: Nach der Zustimmung wird sich auf ein gemeinsames zukünftiges Verhalten geeinigt.

Conversation Strategies

Kommunikation ist unerlässlich für das Entwerfen von Architektur. Basierend auf einem Ziel, einem initialen Wissensstand, sowie der Situation der Kommunikation muss hieraus eine "conversation strategy" abgeleitet werden, um möglichst effizient zu Arbeiten. Eine Gesprächsstrategie deckt im optimal Fall die folgenden Punkte ab: Execution plan: Eine Unterhaltung kann aus mehreren Subunterhaltungen bestehen, welche lediglich ein Subziel behandeln. Das Gesamtziel darf nie außer Acht gelassen werden. Description languages: Welche Sprach genutzt werden soll. Media: Welche Hilfsmittel werden eingesetzt. Cognitive mode: Die Art wie Informationen gesammelt werden sollen (analytisch oder experimentell) Social mode: Die Art, auf welcher mit anderen Gesprächsteilnehmern zusammengearbeitet wird. (Expert-driven, Participatory)

Knowledge Goals

Wie bei der "communication knowledge" gibt es auch bei den "knowledge goals" eine Kategorisierung: -Introduction of knowledge (Trainings) -Agreement to knowledge (Einigung der Stakeholder) -Commitment to knowledge (Stakeholder arbeiten nach den Zusagen)

Conversation Strategies

Brown-paper session: Typisches Brainstorming mit PostIt's. Gedanken werden strukturiert und kategorisiert.

Elicitation interview: Klassisches Interview. Der Interviewer versucht durch Fragen an Informationen zu gelangen.

Workshop: Klassischer geführter Workshop. Mitarbeit durch Teilnehmer an einem Modell, sodass direkt Feedback eingearbeitet werden kann.

Validation interview: Kleiner als ein "elicitation interview". Informationen sind bereits bekannt und in einem Modell vorhanden. Diese Informationen werden durch die Interviewpartner gespiegelt und geprüft.

Committing review: Stakeholder entscheiden sich zwischen mehreren Vorschlägen und committen sich darauf.

Presentation: Typische Präsentation (Front-Beschallung).

Mailing: Massenkommunikation um ein Modell vorzustellen oder zu übergeben.

Eine Übersicht bietet das folgende Bild (vgl. Buch:Enterprise Architecture at work).

<i>Conversation Technique</i>	<i>Knowledge Goal</i>		
	Introduce	Agree	Commit
Brown-paper session	++	+	-
Elicitation interview	++	+	-
Workshop	+	++	+
Validation interview	-	++	+
Committing review	-	-	++
Presentation	++	-	-
Mailing	+	-	-

A + indicates that a certain conversation class is well suited for the selected technique of knowledge goals, while ++ indicates that it is particularly well suited. On the other hand, a - indicates that a certain conversation technique is ill-suited for the selected class of knowledge goals

Eine Sprache zur Modellierung eines Unternehmens

Eine Herausforderung in der Modellierung eines Unternehmens ist, dass die Modellierungssprache umfassende Domänen unterstützen sollte. Derzeit benutzen Architekten aus verschiedenen Domänen eigene Methoden und Konventionen, um eine Modellierung zu realisieren. Das führt dazu, dass Modelle nicht formal genug beschrieben sind oder zu komplex und zu spezifiziert für die jeweilige Domäne beschrieben sind. Nicht formale Modelle führen zu Missverständnissen und behindern die Kollaboration von Architekten, während zu komplexe Modelle mit Verständnisschwierigkeiten einhergehen und die Kommunikation mit Laien einschränkt. Weiterhin sind verschiedene Sprachen oft inkompabil zueinander und lassen sich in Kombination nicht sinnvoll visualisieren.

Im Folgenden wird ArchiMate als Sprache zur Modellierung eines Unternehmens beschrieben. ArchiMate bietet Methoden um Domänen gekapselt voneinander zu beschreiben und stellt gleichzeitig die Relationen der Domänen zueinander dar. Weiterhin fokussiert die Sprache das Verständnis verschiedener Benutzer. Beispielsweise können Modelle in verschiedenen Perspektiven betrachtet werden und es werden nur die für den jeweiligen Benutzer (Architekten, Entwickler, Manager, etc.) relevanten Informationen in einer geeigneten Darstellung präsentiert.

[vgl. S. 74-75]

ArchiMate Framework

In diesem Abschnitt werden die Konzepte des ArchiMate Frameworks erläutert. Dabei werden zuerst die Kernkonzepte gemäß der Abbildung 5.1 betrachtet. Die Kernkonzepte folgen dem natürlichen Sprach- und Satzaufbau: Subjekt, Prädikat, Objekt. Wird die Abbildung von rechts nach links spaltenweise betrachtet, so stellt die aktive Struktur (Subjekt, Active structure) das Subjekt dar, welches ein Verhalten (Prädikat, Behaviour) auf eine passive Struktur (Objekt, Passive structure) ausübt.

Bei der Betrachtung der Abbildung 5.1 von oben nach unten gibt es eine interne und eine externe Sichtweise. Dabei stellt der Dienst (Service) eine essentielle Funktionalität dar, die über eine Schnittstelle (Interface) für den Benutzer erreichbar ist.

[vgl. S. 77-78]

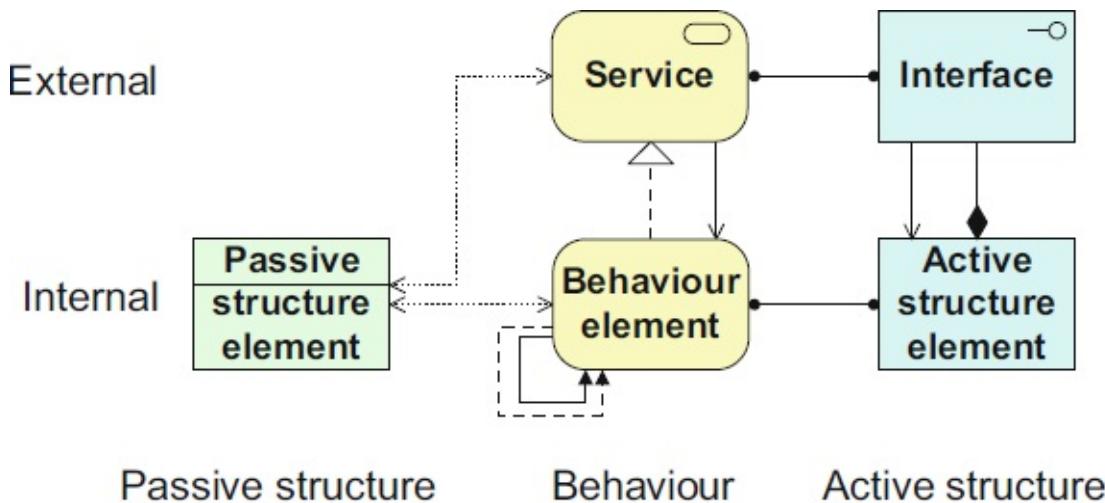


Abbildung 5.1: Kernkonzepte des ArchiMate Frameworks als Metamodell

Die zuvor erläuterten Kernkonzepte spiegeln sich in jeder Ebene des ArchiMate Frameworks wider. Die Abbildung 5.2 bietet eine Übersicht über alle Schichten des Archimate Frameworks in Verbindung mit den Kernkonzepten und hilft beim Design und der Beschreibung einer Unternehmensarchitektur. Hierbei stellen die Schichten Business, Application und Technology, die Kernschichten des Frameworks dar. Die Kernschichten modellieren die essentiellen Vorgänge im Unternehmen, welche notwendig sind um einem Kunden einen Dienst bzw. ein Produkt anbieten zu können. Im Anschluss werden die einzelnen Aspekte der Abbildung 5.2 beschrieben:

- **Passive structure, Behaviour, Active structure:** Stellen die Kernkonzepte des Archimate Frameworks dar.
- **Motivation:** Zur Modellierung der Gründe, weshalb bestimmte Entscheidungen für die Architektur getroffen wurden.
- **Strategy:** Zur Modellierung von Fähigkeiten des Unternehmens (Capabilities), der Ressourcen des Unternehmens und dem Kurs, welchen das Unternehmen anstrebt (Strategie, Taktik).
- **Business:** Modellieren der Prozesse für die Herstellung der Produkte, welche das Unternehmen für die Kunden bereitstellt.
- **Application:** Modellieren der Applikationen bzw. der Software, welche Businessprozesse vereinfachen oder unterstützen.
- **Technology:** Modellieren der Technologie, welche die Infrastruktur zur Realisierung der Applikationen im Unternehmen bietet.
- **Physical:** Modelliert physikalische Beschaffenheiten des Unternehmens wie Ausrüstung, Materialien und Transport.
- **Implementation & Migration:** Modellierung von Projektportfolios, Lückenanalyse (auch: Gapanalyse) und Tranistions- und Migrationsplanung.

[vgl. S. 76-79]

	Passive structure	Behavior	Active structure	Motivation
Strategy	resources	courses of action, capabilities	resources	
Business	business objects	business services, functions and processes	business actors and roles	
Application	data objects	application services, functions and processes	application components and interfaces	stakeholders, drivers, goals, principles and requirements
Technology	artifacts	technology services, functions and processes	devices, system software, communication networks	
Physical	material		facilities, equipment, distribution networks	
Implementation & migration	deliverables	work packages	plateaus	

Abbildung 5.2: Konzept des ArchiMate Frameworks

Kombinieren anderer Methoden und Standards mit ArchiMate

ArchiMate soll herkömmliche Methoden der Modellierung nicht ersetzen, sondern eine Schnittstelle zwischen den Methoden bieten. Dabei werden einzelne Methoden der Modellierung in Kombination mit ArchiMate verwendet. Domänen-spezifische Sprachen, wie bspw. UML für Software oder BPMN für Business-Prozesse, können durch ArchiMate in Relation zueinander gesetzt werden. Somit bietet ArchiMate eine für Laien verständliche abstrakte Sicht auf eine Vielzahl von Domänen und deren Relationen zueinander, während die herkömmlichen Methoden innerhalb der Domänen eine weitere Detailtiefe für Experten zulassen (siehe Abbildung 6.1).

[vgl. S. 123]

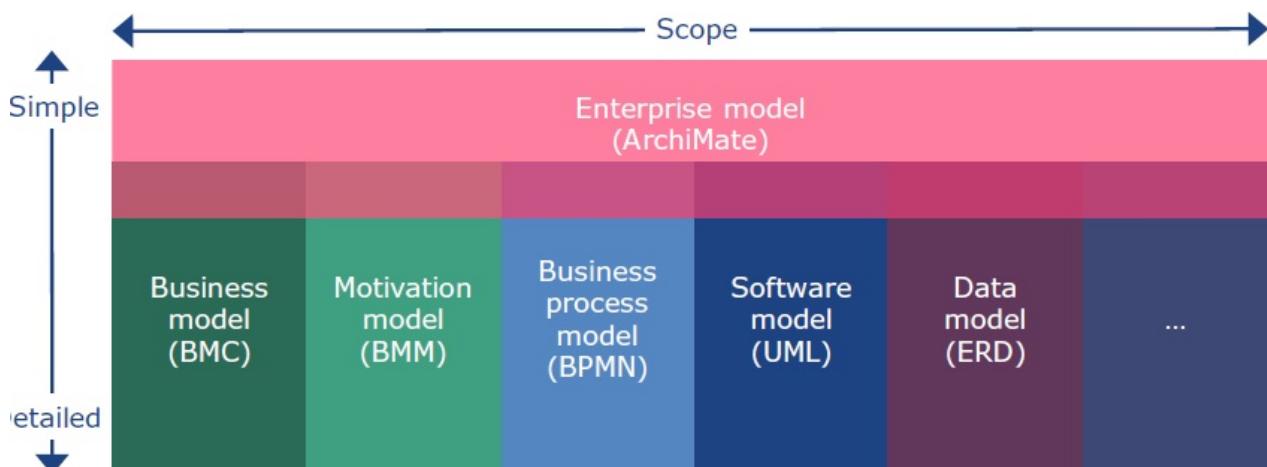


Abbildung 6.1: ArchiMate als Schnittstelle zu herkömmlichen Sprachen der Modellierung

Business Motivation Model

Das Business Motivation Model (BMM) beschreibt eine detailliertere Sicht für eine Business Motivation und steht in Relation zu der zuvor erwähnten Schicht "Motivation" des ArchiMate Frameworks. Tabelle 6.1 stellt die Elemente von BMM den Elementen von ArchiMate (AM) gegenüber.

[vgl. S. 125]

<i>BMM</i>	<i>ArchiMate</i>
Vision, Desired Result (Goal, Objective)	Goal
Mission, Course of Action (Strategy, Tactic)	Course of Action
Directive (Business Policy, Business Rule)	Principle, Requirement, Constraint
Assessment	Assessment
Influencer	Driver
Potential Impact	Outcome

Tabelle 6.1: Gegenüberstellung BMM zu AM

Business Process Model and Notation

Business Process Model and Notation (BPMN) gilt als Standard in der Modellierung von Business Prozessen. ArchiMate dient zur Modellierung von "High-Level-Prozessen" und deren Relation zum Kontext des Unternehmens. BPMN bietet hier eine detaillierte Sicht auf den kompletten Informationsfluss innerhalb eines Prozesses, welcher vom Kontext des Unternehmens gekapselt ist. Tabelle 6.2 stellt die Elemente der BPMN denen von ArchiMate gegenüber.

[vgl. S. 133-134]

<i>BPMN</i>	<i>ArchiMate</i>
Participant/Pool, Lane	Business Actor, Role, Application Component
Collaboration	Business/Application Collaboration
Process	Business/Application Process
Sequence flow	Triggering
Data association	Access
Inclusive and parallel gateways	Junction
Exclusive and event-based gateways	Or-junction

Tabelle 6.2: Gegenüberstellung BPMN zu AM

Unified Modeling Language

Unified Modeling Language (UML) ist ein Standard in der Modellierung von Software und ist die Inspiration einiger Konzepte von ArchiMate. Die Schicht der "Applikation" ist mit UML streng gekoppelt und bietet daher sehr ähnliche Elemente. Jedoch gibt es feine Unterschiede zwischen UML und ArchiMate. In ArchiMate ist bspw. die Richtung der Relationen häufig umgekehrt. Auf Ebene der Architektur liegt der Fokus auf dem allgemeinen Verständnis, ob ein Dienst (Service) nun bereitgestellt oder aktiv von einem Benutzer abgerufen wird ist in ArchiMate irrelevant, während der Richtung von Funktionsaufrufen innerhalb einer Software eine wichtige Bedeutung beigemessen wird. Ein weiterer wichtiger Unterschied ist, dass ArchiMate zwischen Dienst (Service) und

Schnittstelle (Interface) für diesen Dienst unterscheidet, während in der UML nur die Schnittstelle als solche vorhanden ist. Die Unterscheidung zwischen Dienst (Service) und Schnittstelle (Interface) ist in ArchiMate von Bedeutung um für Laien verständlich darstellen zu können, dass mehrere Schnittstellen dafür zuständig sein können den gleichen Dienst zu unterstützen. Tabelle 6.3 stellt eine Gegenüberstellung der Elemente aus UML zu ArchiMate dar.

[vgl. S. 135-138]

<i>UML</i>	<i>ArchiMate</i>
Actor	Business Actor, Role
Use Case	Requirement + Service
Component	Application Component
Class	Business Object, Data Object
Collaboration	Application Collaboration
Node, Device, Execution Environment	Node, Device, System Software
Artefact	Artefact
Interface	Application Interface + Service
Aggregation, Composition, Generalisation	Aggregation, Composition, Specialisation

Tabelle 6.3: Gegenüberstellung UML zu AM

Entity Relationship Diagram

Das Entity Relationship Diagram (ERD) dient zur Modellierung von Daten und ist aus Sicht einer Architektur zu spezifisch, bspw. sind Fremdschlüssel oder Kardinalitäten aus dem ERD ungeeignet um Informationsflüsse von Daten innerhalb einer Architektur darzustellen. Somit kann das ERD verwendet werden, um technisch relevante Detailtiefen für Experten der Datenmodellierung darzustellen. Dennoch sind die Konzepte eines ERD geeignet, um passive Strukturen in ArchiMate zu modellieren. Eine Gegenüberstellung der Elemente ist in Tabelle 6.4 dargestellt.

[vgl. S. 139]

<i>UML</i>	<i>ArchiMate</i>
Actor	Business Actor, Role
Use Case	Requirement + Service
Component	Application Component
Class	Business Object, Data Object
Collaboration	Application Collaboration
Node, Device, Execution Environment	Node, Device, System Software
Artefact	Artefact
Interface	Application Interface + Service
Aggregation, Composition, Generalisation	Aggregation, Composition, Specialisation

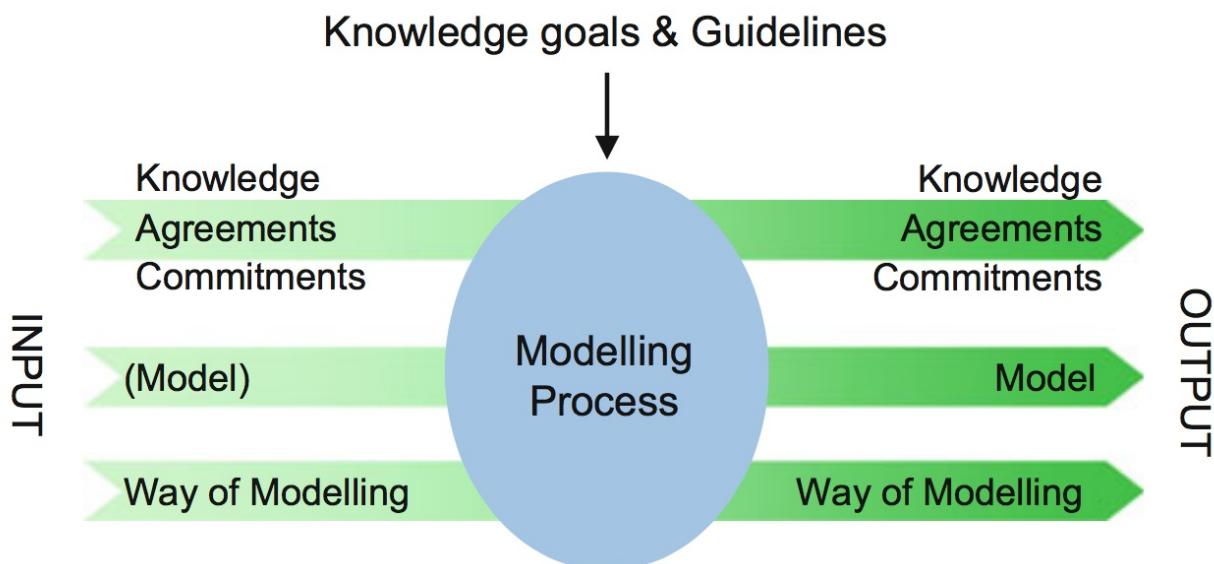
Tabelle 6.4: Gegenüberstellung ERD zu AM

Guidelines for Modelling

Es gibt nicht die eine einzige wahre und richtige Modellierung. Wichtig ist jedoch, dass jede Modellierungsart in sich konzeptionelle Integrität gewährt. Dabei soll die Grundidee eines Modells, unabhängig von dessen Komplexität, immer schnell und einfach verständlich sein. So sollen auch Personen ohne tieferes Wissen stets das Grundgerüst eines Modells nachvollziehen können.

Der Modellierungsprozess

Im Kontext des Buchs ist in ein Modell immer ein abstraktes Abbild eines (Teil-)Aspekts der realen Welt. Die Modellierung ist dabei ein Transformationsprozess, bei dem reale Aspekte getreu übernommen und nicht verändert werden sollen, sondern nur gefiltert und konkretisiert werden (vgl. Abbildung, der Informationsgehalt soll sich nicht ändern). Das ermöglicht es reale Problemstellungen verständlich zu kommunizieren; entweder für Personen, die das Modell lesen sollen, und auch für die, die es kollaborativ erstellen.



Beim Modellieren werden typischerweise iterativ folgende Schritte durchlaufen:

- Erstellung und Definition des Zwecks einer Modellierung in Absprache mit den Stakeholdern

- Auswählen einer oder mehrerer Sichtweisen, was die zu berücksichtigenden Inhalte des Modells bestimmt
- Erstellung und Strukturierung des Modells, was stets in Kombination geschieht
 - Bei der Erstellung können zusätzliche Information durch weitere Gespräche mit den Stakeholdern eingeholt werden
 - Die Strukturierung dient der Aufteilung und Vereinfachung von komplexen Abbildern, sowie der Aufdeckung von sich wiederholenden Mustern aber auch Inkonsistenzen
- Visualisierung des Modells; dabei stets überlegen, welche Sichtweise in jedem konkreten Prozessabbild am sinnvollsten ist und wie diese am besten dargestellt werden kann (textuell oder visuell)
- Verwendung des Modells, durch Präsentation bei den Stakeholdern
 - Validierung durch das Einholen von Stakeholder-Feedback
 - Zusagen des Stakeholder einholen, ob der Entwurf übernommen werden soll
 - Weitere Stakeholder informieren
- Stetige Überarbeitung und Aktualisierung des Modells

Hilfreich bei der Modellierung ist die Definition allgemeingültiger Begriffe, sogenannte Modellierungsaktionen. Diese lassen sich aus den typischen Aufgaben ableiten. Die Aktionsbegriffe dienen als einfaches Vokabular. Der Modellierende selbst behält damit einen strukturierteren Überblick über seine Handlungen. Des Weiteren soll er damit sein eigenes Vorgehen stets objektiv und rational reflektieren können. Definieren lassen sich folgende Aktionen:

- Ein neues Element in einem Modell einführen, das relevant sein könnte
- Ein Element verfeinern
 - Klassifizieren nach bestehenden Kategorien oder eine neue Kategorie erstellen
 - Element beschreiben durch einen Text und/oder weitere Subelemente
- Ein Element verwerfen; dabei aber festhalten, warum man es entfernt und was man ggf. positives mitnehmen kann
- Ein Konzept oder eine Beziehung abstrahieren; dabei können ggf. unwichtige Detailinformationen verwirftverworfen werden
- Ein neues Element übersetzen bzw. ein passendes Element in einem anderen Modell identifizieren, dass das selbe aussagt, ggf.:
 - Zusätzliche Elemente für eine geeignete Übersetzung erstellen oder ersetzen
 - Diese Elemente mit den Element des anderen Modells verknüpfen
 - Eine vermittelnde Sprache zwischen den beiden Modellen modellieren
 - Übersetzungsregel definieren
 - Mögliche Missverständnisse festhalten
- Alle Aktionen sollten dokumentiert werden

Richtlinien für's Modellieren

Für die verschiedenen Aspekte beim Modellieren lassen sich jeweils folgende Richtlinien formulieren.

Zielorientierte Modellierung

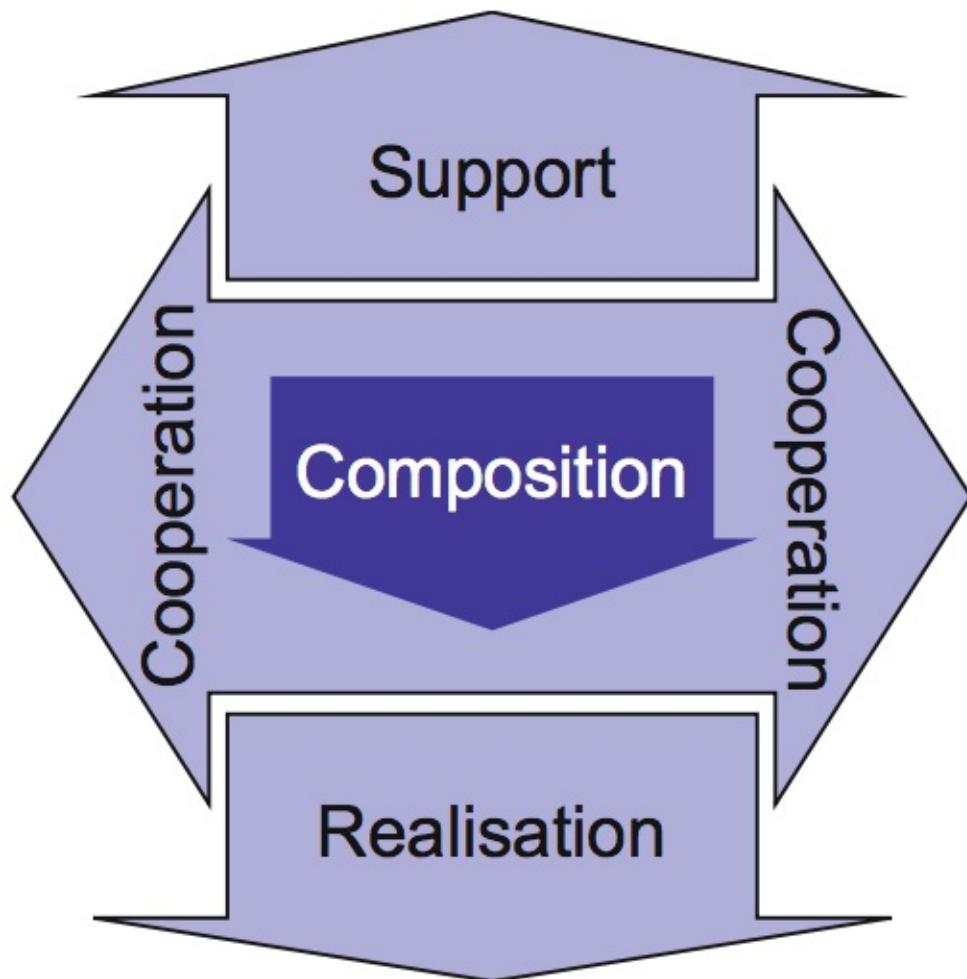
- Ein Modell soll Antworten auf Fragen liefern
- Man soll einen klaren Unterschied zwischen dem Modell und seiner Visualisierung machen
- Quantitätsmaxime: Ein Modell soll so informativ, wie nötig, aber nicht informativer, als nötig sein
- Qualitätsmaxime: Man soll nichts modellieren, was man für falsch hält oder für was man nicht genügend Belege hat
- Relevanzmaxime: Nur wirklich relevante Dinge modellieren
- Maxime der Art und Weise: Unklare Aussagen und Mehrdeutigkeiten vermeiden und sich kurz und prägnant formulieren
- Iterativ modellieren
- Dynamisch modellieren
- Ökonomisch im Model sein
- Ökonomisch in der Sichtweise sein
- Konzepte erkennbar machen
- Strukturen erkennbar machen
- Konsistenz innerhalb eines Modells bewahren
- Konsistenz zu referenzierten Modellen bewahren
- Modelle so korrekt und vollständig wie nötig machen
- Verschiedene Aspekte orthogonal betrachten

Bevor man startet

- Gibt es klar definierte Stakeholder?
- Ist die Zielsetzung explizit beschrieben?
- Trägt eine Enterprise-Architektur zur Erreichung der Zielsetzung bei?
- Ist der Rahmen klar, von dem, was modelliert werden soll?
- Ist klar, ob die Ist- oder die Soll-Situation modelliert werden soll?
- Sind alle Informationen zur Erstellung des Modells verfügbar?
- Sind die realistischen Erwartungen an den Enterprise-Architekten klar?

Inhalte

- Die Sichtweise wählen, die der Zeilsetzung genügen
- Stets den Fokus behalten
- Weniger wichtige Dinge und Ausnahmen vernachlässigen
- Sich nicht davor scheuen, bereits modellierte Elemente wieder zu verwerfen
- Konsistente Zwischenversionen mit den Stakeholdern besprechen
- Von einem Element ausgehend beginnen zu modellieren; dabei sind vier Richtungen möglich:
 - nach innen (hin zum inneren Aufbau eines Elements)
 - nach außen (hin zu den unterstützten Elementen)
 - nach unten (hin zu Realisierungen anderer Elemente)
 - seitwärts (hin zu kooperierenden Elementen)

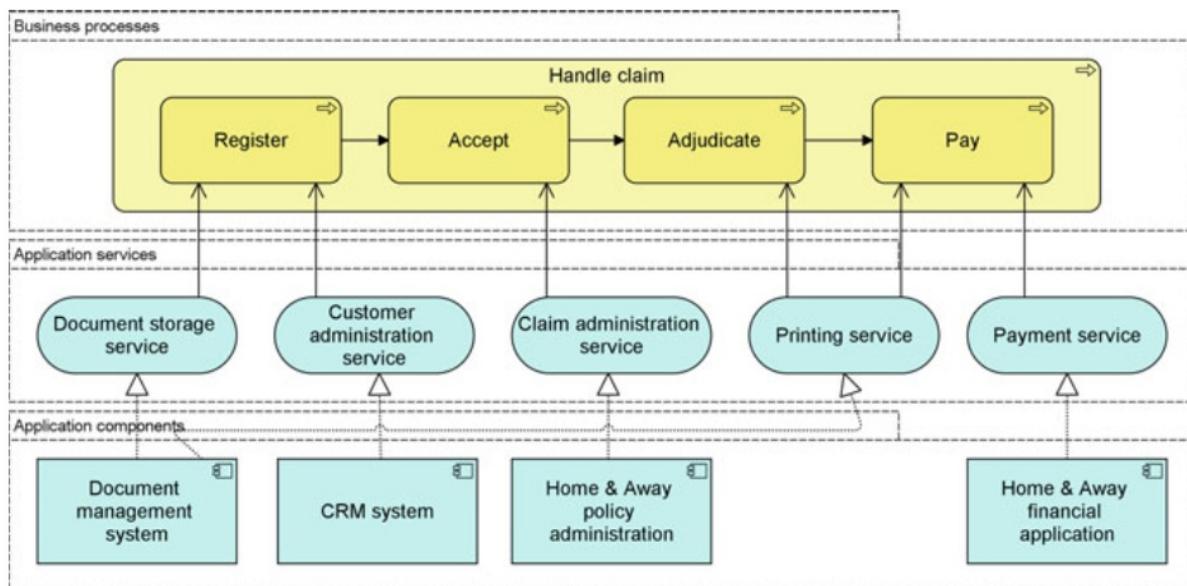


Abstraktion

- Zuerst Schlüsselkonzepte und -beziehungen auf einem abstrakten Ebenen festhalten
- Nur eine begrenzte Zahl an vordefinierten Abstraktionsebenen verwenden; die Ebenen sollten auf Basis der Modellierungsziele definiert werden
- Abstraktionsebenen konsistent halten

Struktur und Visualisierung

- Ein Modell so selbsterklärend wie möglich machen
- Internes und externes Verhalten von einander trennen
- Layer verwenden



- Nach Phasen gruppieren
- Nach Produkt oder Dienst gruppieren
- Nach verwendeten Informationen gruppieren
- Nach geographischen Standorten gruppieren
- Unabhängige Komponenten von einander trennen

Konstruktiver Umgang mit Kommunikationsverlusten

Es kann immer vorkommen, dass ein Stakeholder ein Modell ab einem bestimmten Punkt nicht mehr versteht und es somit zu einem sogenannten Kommunikationsverlust kommt. Solche Zustände sind nicht gut, können aber produktiv genutzt werden, um wieder auf die Richtige Bahn zu kommen. Dazu muss das Modell hinsichtlich Lesbarkeit und Wirkung überprüft werden und je nachdem Korrekturen vorgenommen werden.

Bei der Lesbarkeit können folgende Probleme auftreten:

- Das Modell wird nicht verstanden, da unbekannte Begriffe und Konzepte verwendet werden

werden. Lösung: Die dem Stakeholder bekannte Begriffe verwenden und das neue Vokabular vorstellen.

- Das Modell wird falsch verstanden, weshalb der Stakeholder falsche Schlussfolgerungen zieht. Lösung: Die dem Stakeholder bekannte Begriffe verwenden und das neue Vokabular vorstellen.
- Das Modell hat keine intuitive Struktur für den Empfänger. Lösung: Die Logik der Modelle des Stakeholders analysieren und dessen Struktur adaptieren.
- Das Modell hat eine unklare Struktur oder Notation und verursacht Fragen. Lösung: Strukturen (anhand von bekannten Beispielen) erklären.
- Die Visualisierung des Modells lenkt von der eigentlichen Nachricht ab. Lösung: Visuelle Ausschmückungen in Form von übertriebenen Diagrammen und Farben zugunsten der Inhaltsvermittlung weglassen.

Bei der Vermittlung und Wirkung können folgende Probleme auftreten:

- Die Statusberichte bleiben aus, da der Stakeholder nie Zeit hat. Lösung: Organisatorisches Problem, das nicht in diesem Buch behandelt wird.
- Das Modell hat eine zutreffende, aber ungewollte Nachricht. Lösung: Auf ein organisatorisches Problem, s.o.
- Das Modell ist irrelevant, ist also gut aber für die Problemstellung nicht von Bedeutung. Lösung: Vgl. 'Das Modell beinhaltet überflüssige Elemente'
- Das Modell ist überflüssig, da der Architekt zu viele irrelevante Aspekte betrachtet. Lösung: Problemstellung reflektieren und darauf fokussieren.
- Das Modell ist zu komplex. Lösung: Mehr Abstraktionsebenen verwenden.
- Das Modell ist nicht konkret genug. Iterieren und Modell konkretisieren.
- Das Modell ist nicht umfassend genug, da nicht alle notwendigen Informationen abgebildet sind. Lösung: Weitere Aspekte des Stakeholders berücksichtigen ohne das Modell zu komplex zu machen.
- Das Modell ist nicht richtig und beinhaltet falsche Annahmen oder Begründungen. Lösung: Informationen neu sammeln und auf Richtigkeit prüfen.

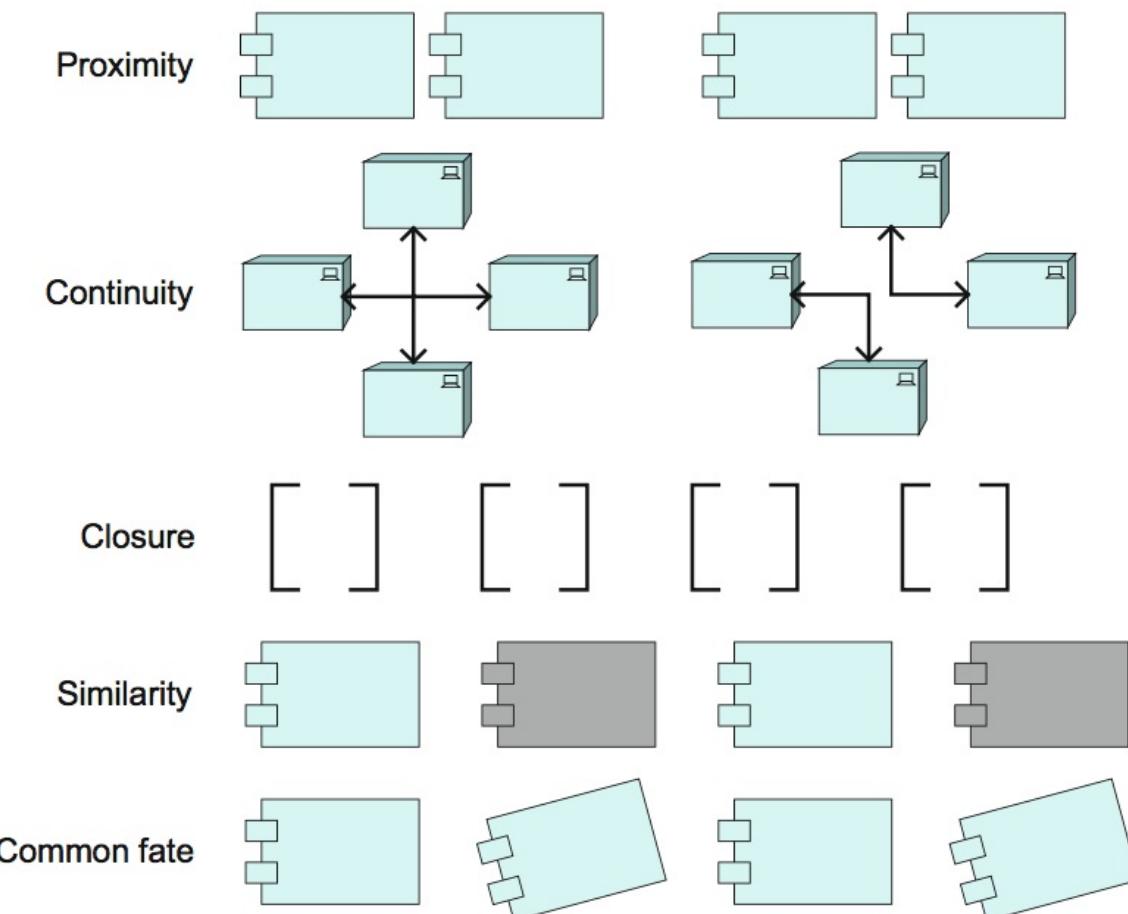
Lesbarkeit und Anwendbarkeit eines Modells

Die Lesbarkeit eines Modells hängt hauptsächlich von dessen Komplexität ab, weshalb diese möglichst gering gehalten werden muss. Dazu sollte die Anzahl der Elemente, der Elementtypen und der Beziehungen gering gehalten werden.

Zudem sollte die visuelle Komplexität reduziert werden, indem möglichst mehrere Sichtweisen verwendet werden, die Stakeholder-spezifische Teilespekte ein- oder ausgeblenden. Des Weiteren hilft auch hier das Hinzufügen weitere (Zwischen-)Ebenen, die Bereiche voneinander trennen.

Berücksichtigt man zudem die Gestalt-Prinzipien (vgl. Abbildung), kann das Bild ohne Reduzierung von Elementen besser lesbar gemacht werden:

- Nähe (Proximity): Elemente, die näher beieinander stehen, werden intuitiv als zusammengehörend wahrgenommen
- Kontinuität (Continuity): Möglichst gerade Linien und nicht abknickende Linien verwenden
- Abschluss (Closure): Symmetrien und Stringenz werden besser wahrgenommen
- Gleichheit (Similarity): Objekte, die gleich aussehen, werden als zusammengehörend wahrgenommen und solche, die gleiche Größe haben, als von gleicher Priorität wahrgenommen.
- Drehung (Common fate): Wenn Objekte gleichen Aussehens unterschiedliche Orientierung haben, werden alle zusammen nicht mehr als eine Gruppe, sondern entsprechend der sich wiederholten Orientierungen, als mehrere einzelne Gruppen wahrgenommen.

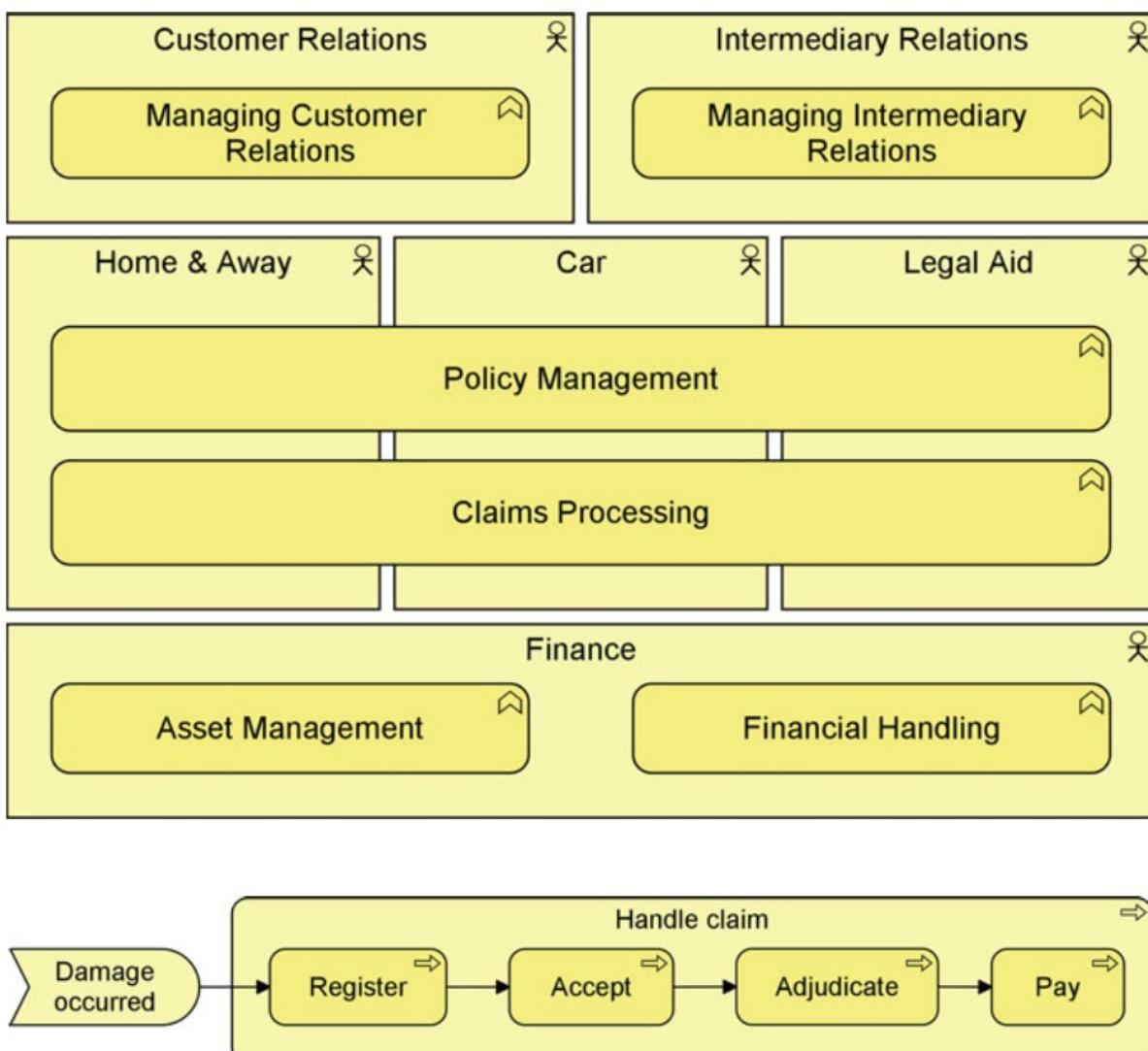


Darüber hinaus erhöhen die Repräsentationskonventionen die Lesbarkeit von Modellen. Es geht um die Optimierung und richtige Verwendung von Layout, Symbolen, Farben und Texten.

Das Layout ist das wichtigste Attribut bei der Gestaltung von verständlichen Modellen.

Wichtige Punkte sind:

- freie, weiße Flächen verwenden, was Diagramme eleganter erscheinen lässt und Änderungen besser zulässt
- zwischen Normal- und Ausnahmefällen unterscheiden, indem man dem normalen Prozess eine eigenen Ebene lässt und Ausnahmebehandlungen parallel darunter oder darüber separat abbildet
- Symmetrien verwenden, um Ähnlichkeiten darzustellen
- Zeitabhängigkeiten von links nach rechts modellieren
- Sich überschneidende Linien vermeiden



Durch die Verwendung von realistischen und nachvollziehbaren Symbolen (Zylinder für Datenspeicher, Strichmenchen für Personen und Rollen) lassen sich Grundelemente ohne weitere Beschriftung erklären. Wichtige Punkte dabei sind:

- Ähnliche Zeichnungen für ähnliche Konzepte verwenden

- Die Liniendicke variieren, um wichtige Beziehungen zu unterstreichen

Mit Farben lassen sich Zugehörigkeiten, Gruppierungen sowie Prioritäten gut darstellen.

Wichtige Punkte sind:

- Farben nutzen zur Betonung von Elementen
- Farben nutzen, um Zusammengehörigkeiten darzustellen
- Farben nutzen, um Emotionen zu erzeugen
- Nicht zu viele Farben nutzen

Zuletzt ist auch die Wahl der richtigen Worte entscheidend, um Modelle schnell und leicht nachvollziehbar zu machen. Wichtige Punkte dabei sind:

- die Verwendung von Domänen-spezifischer Terminologie (die Sprache des Stakeholders)
- Namenkonventionen beachten (Verben für Aktionen, Nomen für Ressourcen)

Viewpoints and Visualisation

Architecture Viewpoints

Im Kontext von Architekturen dienen Viewpoints (Standpunkt) der Betrachtung einzelner Aspekte einer Architektur, die im Interesse der Stakeholder sind. Die Viewpoints sollten dabei nicht nur uni-direktional, sondern auch bi-direktional bei der Kommunikation mit den Stakeholdern anwendbar sein. So können Viewpoints sowohl zu Präsentationszwecken wie auch als Diskussionsgrundlage im Gespräch zwischen Architekten und Stakeholdern dienen. Dabei gibt es entsprechende Viewpoints für die verschiedenen Stakeholder-Interessen:

- Obere Managementebene: Steht das neue System nicht der Unternehmensstrategie im Widerspruch und hat es einen Mehrwert; unter Berücksichtigung des (finanziellen) Aufwands?
- Mittlere Managementebene: Wie ist die aktuelle Lage unter Berücksichtigung des computergestützten Business-Prozesses?
- Endnutzer: Wie wirkt sich das neue System auf die Aktivitäten des voraussichtlichen Nutzers aus?
- Architekt: Ist die Wartbarkeit des Systems gegeben?
- Produktmanager: Braucht es neue Technologien? Müssen Prozesse oder Applikationen angepasst werden? Wie sicher ist das System?
- Projektmanager: Wo besteht die Abhängigkeit zwischen den Business Prozessen und den zu entwickelnden Applikationen? Wie performant sind die Prozesse?
- Systementwickler: Wo sind Änderungsbedarfe?
- Systemadministrator: Wie verändern sich die Aufgaben durch die neuen Prozesse?

Nach dem ISO/IEC/IEEE 42010:2011-Standard ist ein Viewpoint eine Spezifikation der Konventionen bei der Erstellung und Benutzung von Views. Zu erwähnen sind die Frameworks Zachman framework, Krutchens 4+1 View Model und TOGAF, die allesamt verwendbare Viewpoints definieren.

Models, Views und Visualizations

Nach dem Prinzip Separation of Concerns werden bei der Präsentation einer Architektur (Viewpoint) die drei Komponenten Model, View und Visualization unterschieden. Das Modell beinhaltet alle Informationen der Architektur, die durch die View gefiltert und aufbereitet

werden, um in der Visualization dargestellt zu werden. Die Visualization bestitzt dabei keinerlei Logik, sondern dient nur der Anzeige (vgl. Abbildung).



Visualisierung und Interaktion

Die Modifizierung einer Architektur geschieht typischerweise direkt im Modell, was dessen Konsistenz gewährleisten kann. Darüberhinaus gibt es jedoch auch Werkzeuge, die eine Änderung direkt vom View aus erlauben. Das Werkzeug projektiert die Änderung der View möglichst intelligent auf das darunterliegende Modell (vgl. Abbildung).



Erstellung, Auswahl und Verwendung von Viewpoints

Die gegebenen Frameworks bieten nur eine begrenzte Anzahl an verschiedenen Viewpoints, was nicht immer genügt, um Modelle den Stakeholdern passend zu visualisieren. Die Erstellung individueller Viewpoints ist möglich, jedoch aufwendig und somit kostspielig. Es muss also entschieden werden, ob gegebene Viewpoints genügen oder neue erstellt werden müssen. Dazu werden die Anforderungen klassifiziert (vgl. Abbildung). Bei der Klassifizierung betrachtet man Zielsetzung und Inhaltsebene.

	Stakeholder	Zielsetzung	Beisp
Gestalten	Architekten, Software-Entwickler, Geschäftsprozessgestalter	navigieren, gestalten, Gestaltungsentscheidungen unterstützen, Alternativen vergleichen	UML-Diagr BPMN-Diag Flowcharts, Diagramme
Entscheiden	Manager, CIO, CEO	Entscheidungsfindung	Querverwei Landscape Liste, Bericht
Informieren	Mitarbeiter, Kunde	erklären, überzeugen, Feedback einholen	Animationen Cartoons, Prozessillus Grafik

	Stakeholder	Zielsetzung	Beispiele
Details	Software-Entwickler, Prozessverantwortlicher	gestalten, verwalten	UML-Klassendiagramm, Testbed procedure diagram
Zusammenhänge	Produktmanager	Abhängigkeiten analysieren, Änderungsauswirkungen	Views, die Zusammenhänge verdeutlichen: 'realise', 'assume'
Überblick	Enterprise-Architekt, CIO, CEO	Changemanagement	Landscape diagram



Richtlinien bei der Verwendung von Viewpoints sind:

- Scoping: Ein oder zwei Viewpoints auswählen, die zu modellierenden (Sub-)Domänen wählen und Rahmenbedingungen festlegen
- Creation of views: Inhalte für den Viewpoint erstellen oder extrahieren
- Validation: Zusammen mit den Stakeholdern die Viewpoints prüfen (lassen)
- Obtaining commitment: Verbindliche Zusage der Stakeholder für einen Viewpoint einholen
- Informing: Weitere Stakeholder informieren, die nur sekundär betroffen sind

Basic Design Viewpoints

Zur Übersicht seien im Folgenden die gängigen Viewpoint-Arten erwähnt und auf die Erklärungen und beispielhaften Diagramme im Buch verwiesen:

- Introductory Viewpoint
- Organisation Viewpoint
- Actor Cooperation Viewpoint
- Business Function Viewpoint
- Product Viewpoint
- Service Realisation Viewpoint
- Business Process Cooperation Viewpoint
- Business Process Viewpoint
- Information Structure Viewpoint
- Application Cooperation Viewpoint
- Application Usage Viewpoint
- Application Behaviour Viewpoint
- Application Structure Viewpoint

- Technology Viewpoint
- Technology Usage Viewpoint
- Implementation and Deployment Viewpoint
- Physical Viewpoint

Motivation Viewpoints:

- Stakeholder Viewpoint
- Goal Realisation Viewpoint
- Goal Contribution Viewpoint
- Principles Viewpoint
- Requirements Realisation Viewpoint
- Motivation Viewpoint

Strategy Viewpoints:

- Strategy Viewpoint
- Capability Map Viewpoint
- Ressource Map Viewpoint
- Outcome Realisation Viewpoint

Implementation and Migration Viewpoints:

- Project Viewpoint
- Migration Viewpoint
- Implementation and Migration Viewpoint

Architecture Analysis

Architecture Alignment

Tool Support

Case Studies

Beyond Enterprise Architecture

Softwareprojekte der Studenten

Projektname: **VR** Projektleiter: Tim Jastrzembski Projektteam: Tim Jastrzembski Github-Repo: <https://github.com/tjastrzembski/ViReCo.git>

Projektname: **AR** Projektleiter: Andrei Güter Projektteam: Andrei Güter Github-Repo:

Projektname: **Modell getriebene Systementwicklung** Projektleiter: Projektteam: Jonathan Jansen, Oliver Nagel Github-Repo:

Projektname: **Continous Software Engineering** Projektleiter: Projektteam: Lukas Taake, Marcel Dzaak, Marvin Schirrmacher Github-Repo:

Projektname: **AI**

Projektleiter:

Projektteam: Daniel Beneker, Sven Schirmer, Yannick Kloss

Github-Repo: [Twitter Miner](#)

Projektname: **Embedded Computing** Projektleiter: Gamze Söylev Öktem Projektteam: Nils Kohlmeier, Wladimir Streck, Gamze Söylev Öktem, Jonas Wiese, Justin Jagieniak Github-Repo: <https://github.com/nkohlmeier/Spezielle-Gebiete-zum-Softwareengineering.git>

Projektname: **Fullstack Development** Projektleiter: Projektteam: Timo Rolfsmeier, Niklas Harting, Alexander Schwietert, Tolga Aydemir, Malte Berg, Lutz Winkelmann, Fabian Lorenz, Benjamin Schmidt

Pflichtenheft: [PDF](#) GitLab-Group: [ShareBase](#)

Projektname: **Cloud Security** Projektleiter: Philipp Viertel Projektteam: Philipp Viertel, Andre Kaleja Github-Repo: <https://github.com/pviertel/cloudsecurity.git>

Projektname: **Software-Architektur** Projektleiter: Christian Holzberger Projektteam: Christian Holzberger Github-Repo: <https://github.com/cHolzberger/2017-Sw-Eng-Projekt.git>

VR

Projektleiter: Tim Jastrzembski

Projektteam: Tim Jastrzembski

Github-Repo: <https://github.com/tjastrzembski/ViReCo.git>

Einführung

Das Thema VR gewinnt in der heutigen Zeit immer mehr an Beliebtheit. Aber was ist VR?

Definition



(Quelle: <http://static.giga.de/wp-content/uploads/2015/05/The-Witcher-3-einstieg.jpg>)")

Der Begriff **Virtual Reality** (kurz: **VR**) ist eine vom Computer generierte (alternative) Realität, welche fiktiv und bzw. oder unsere Wirklichkeit annähern kann. Dabei sind zahlreiche Computerspiele ein gutes Beispiel dafür: Sei es ein Rollenspiel (*Final Fantasy*, *The Witcher*, *World of Warcraft (WoW)*), ein Abenteuerspiel (*Tomb Raider*), Actionspiel (*Half-Life*), Egoshooter (*Counter Strike*) oder Simulationen für verschiedene Berufe (*Euro Truck Simulator 2*, *Microsoft Flight Simulator X*, *Landwirtschafts-Simulator 17*), jedes dieser Spiele basiert auf einer Welt, die entweder frei erfunden (z.B. die Welt von Azeroth in **WoW**) oder an unsere Wirklichkeit angelehnt ist (z.B. die genannten Simulationsspiele). In der heutigen Zeit wird ferner die sogenannte Immersion (engl.: immersive - eintauchen, eindringen) im Zusammenhang mit der **VR** gebracht. Mit Geräten wie dem **Head-Mounted Display** (Datenhelm, kurz: **HMD**), haptische Controller (z.B. Datenhandschuhe) und Sensoren zur Bewegungsverfolgung wird versucht, der **VR** ein noch realistischeres Gefühl zu geben, indem man einerseits die Wahrnehmung der Wirklichkeit reduziert (siehe Bild) und andererseits die Möglichkeiten an Interaktionen mit der virtuellen Umgebung so einfach und natürlich wie möglich gestaltet. Dadurch wird nicht nur eine Immersion, sondern auch

eine Präsenz erzielt. Denn im Gegensatz zu Bildschirm, Maus und Tastatur, die einem nur Einblick von außerhalb in die virtuelle Realität ermöglichen, geben die **HMDs** einem das Gefühl, mittendrin zu sein. [1] [2]

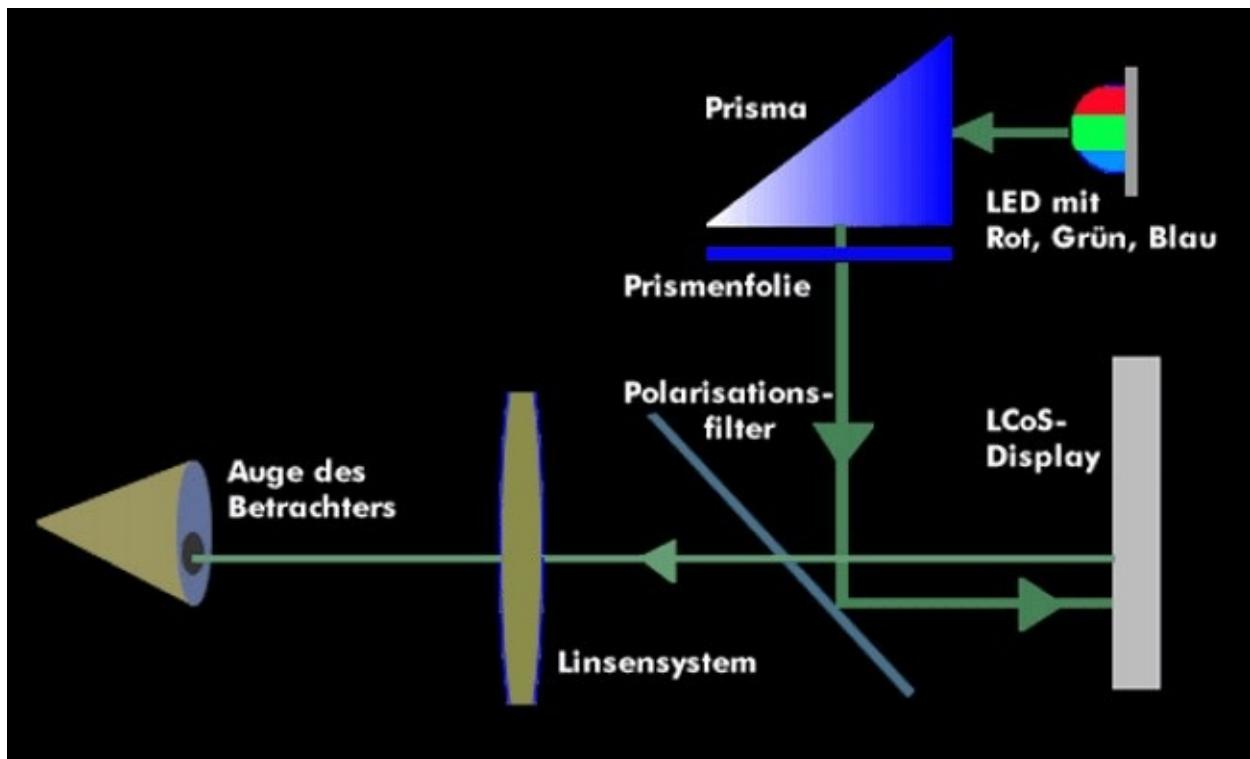
Head-Mounted Display



(Quelle: <https://www.theverge.com/2016/4/5/11358618/htc-vive-vr-review>)")

Wie der Name vermuten lässt, sind Head-Mounted Displays (**HMD**) Projektionseinheiten, welche auf dem Kopf getragen werden. Sie fallen je nach Einsatzgebiet in verschiedenen Größen aus (z.B. Brillenform oder Helmform) und haben verschiedene Darstellungsweisen. Dementsprechend können **HMDs** sowohl für Augmented Reality (siehe **AR**) als auch für **VR** genutzt werden. **HMDs** nutzen speziell für **VR** die Eigenschaften des stereoskopischen Sehens [3]: Indem sie auf zwei Microdisplays, welche sich nah an den Augen befinden, das gleiche Bild, jedoch mit geringfügig unterschiedlichen Blickwinkeln projizieren, entsteht eine Tiefenwahrnehmung der abgebildeten Szene, welche die Szene realistischer erscheinen lässt. [4]

Funktionsweise



(Quelle: <http://www.itwissen.info/lex-images/prinzip-der-hmd-projektion.png>)")

Wie die Abbildung zeigt, erfolgt bei dieser Projektionsweise in 4 Schritten [4]:

1. Drei Leuchtdioden für das RGB-Spektrum erzeugen sequentiell Licht.
2. Die Farben werden separat nacheinander über Polarisationsfilter auf das LCoS-Display geworfen.
3. Das Display reflektiert das eintreffende Licht auf die Linse.
4. Das Licht wird durch die Linse je nach Einfallswinkel gebrochen und auf das Auge geworfen.

Quellen:

[1] <http://www.giga.de/konsolen/oculus-rift/specials/virtual-reality-was-ist-das-definition-brillen-games-technologie/>

[2] http://www.ard.de/home/ard/Was_ist_Virtual_Reality/3364362/index.html

[3] https://de.wikipedia.org/wiki/Stereoskopisches_Sehen

[4] <http://www.itwissen.info/Datenhelm-head-mounted-display-HMD.html>

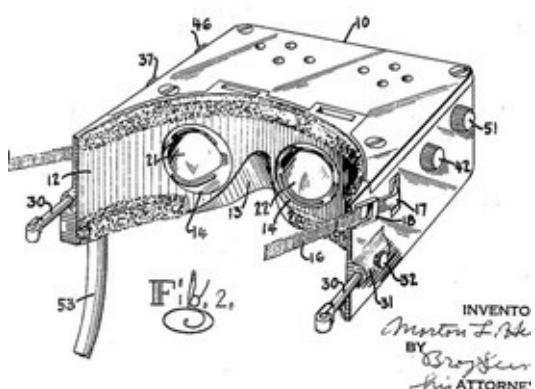
Geschichte der VR

Die Geschichte der **VR** streckt sich über 50 Jahre, obwohl der Begriff selbst erstmals 1982 in einem Roman auftauchte (*The Judas Mandala* von Damien Francis Broderick [10]) und 1987 im *Oxford English Dictionary* aufgenommen wurde. Die erste nachweisbare Annäherung eines Apparates, welches eine **VR** simulieren sollte, wurde um 1955 von Morton Heilig erfunden und sieben Jahre später präsentiert: Der *Sensorama*. Dieser Automat war in der Lage, stereoskopische Bilder, Stereosound, Erschütterungen, Gerüche und Wind zu simulieren, wodurch viele Sinne des Menschen gleichzeitig angesprochen werden konnte. Jedoch war M. Heilig mit seiner Erfindung der Zeit zu weit voraus, sodass er diese nicht vermarkten konnte. [5] [6]



(Quelle: <http://www.tomshardware.de/fotoreportage/291-virtuelle-realitat-vr-vr-brille-virtual-reality-hmd.html#s2> ")

Auch sein 1960 patentiertes **HMD**, welches heutigen Modellen verblüffend ähnlich aussieht, wurde nie gebaut.



(Quelle: <http://www.tomshardware.de/fotoreportage/291-virtuelle-realitat-vr-vr-brille-virtual-reality-hmd.html#s3> ")

Das erste echte **HMD** erfand Ivan Sutherland um 1968: *The Sword of Damocles*. Jedoch war der Prototyp derart schwer, dass dieser an der Decke befestigt werden musste. Aufgrund der geringen Rechenleistung der Computer waren die generierten Bilder jedoch bescheiden, weswegen Projekte wie *Build-IT* erst wesentlich später realisiert werden konnten. So wurde die **VR** erst wieder in den 80er-Jahren für Militär und Raumfahrt für Projekte interessant (*Super Cockpit VCASS* und *NASA VIEW*). Aber auch kommerziell blieb sie nicht unbemerkt. Durch Romane wie der *Neuromancer*-Trilogie von William Gibson [7] oder Filme wie *Tron* stieg das Interesse fortgehend, sodass spätestens Anfang der 90er viele Entwickler wie **Nintendo**, **SEGA** oder **Forte** ihr eigenes **HMD** auf den Markt bringen wollten. Aufgrund des zu hohen Preises, mangelnder Anwendung und z.T. geringer Darstellungsauflösung verkauften sich die Geräte nur schwer, sodass die meisten Vertreiber deren Verkauf eingestellt hatten. Erst 20 Jahre später mit der Entwicklung der **Oculus Rift** wurde die Idee von **HMDs** für den kommerziellen Markt wieder aufgegriffen. [5] [6] In diesen Jahren stieg auch die Rechenleistungskraft kommerzieller Computer, sodass sie imstande waren, auch komplexere 3D-Welten anzuzeigen. Mit der Einführung von *OpenGL* (ab 1992) und *DirectX* (ab 1995) wurden auch fortan Programmierschnittstellen zur Entwicklung von 3D-Anwendungen gepflegt, auf welche auch später Szenengraphen und Spiel-Engines wie die *Unreal Engine* aufbauen, welche nun die Entwicklung von **VR**-Umgebungen massiv vereinfachen. Aber auch in Sachen Interaktion hat sich in diesen Jahren viel getan: Die innovative Idee der *Wii* von **Nintendo** im Jahre 2006, das Spiel durch eigene Bewegung zu steuern, hat viele Entwickler inspiriert, ähnliche Konzepte auszuarbeiten (z.B. *Playstation Move*, *Xbox Kinect*). [5] [6] Auch das Verständnis bzgl. **VR** hat sich weiterentwickelt. So werden in Reki Kawaharas *Sword Art: Online*[8] zukünftig mögliche **HMDs*** und eine **dazugehörige VR*** präsentiert, die jedoch Gefahren wie geistige Schäden oder auch Tod bergen kann. Das bekannteste Beispiel jedoch ist die *Matrix**-Trilogie der Wachowski-Geschwister [9], welche auch philosophische Fragen zu den schon genannten Problemen aufwirft (z.B.: Sind wir auch in einer Matrix gefangen?).

Quellen:

- [5] <http://www.tomshardware.de/fotoreportage/291-virtuelle-realitat-vr-vr-brille-virtual-reality-hmd.html#s14>
- [6] <http://www.vrnerds.de/die-geschichte-der-virtuellen-realitaet/>
- [7] <https://de.wikipedia.org/wiki/Neuromancer-Trilogie>
- [8] https://de.wikipedia.org/wiki/Sword_Art_Online
- [9] https://en.wikipedia.org/wiki/The_Matrix
- [10] <http://www.technovelgy.com/ct/content.asp?Bnum=713>

Stand der Technik

Softwaretechnisch gibt es aktuell zahlreiche (Spiel-)Engines zur Entwicklung von **VR**-Anwendungen (vgl. [11]) wie z.B. **Unreal Engine 4** oder **Unity**, welche auch eine gut dokumentierte Umgebung zum Entwickeln von 2D/3D-Anwendungen bieten. Diese nutzen derzeit die aktuellste Version von *DirectX* (12), *OpenGL* (4.5) und mittlerweile auch auf den OpenGL-Nachfolger *Vulkan* [12].

Hardwaretechnisch gibt es heutzutage mehrere Geräte, mit der man ein immersives **VR**-Erlebnis erreichen kann. Nennenswert sind dabei die **HTC VIVE**, **Oculus Rift** und **Sony Morpheus**. Als kostengünstige Alternative gibt es für Smartphone-Besitzer die Möglichkeit, sogenannte *Cardboards* (z.B. die von **Google**) oder **Samsung Gear VR**, sofern man Besitzer eines **Samsung Galaxy S6** oder höher ist.

Bzgl. Interaktion gibt es auch Zahlreiche Möglichkeiten. Neben dem normalen Gamepad (z.B. Xbox360-Controller) liefern die meisten **HMD**-Hersteller interaktive Controller mit (**Oculus**, **HTC**). **HTC** arbeitet aktuell an Knuckles-Controller, welche die Interaktion durch die Hand vereinfachen und verbessern soll [13]. Alternativ kann man die Sets mit experimentellen Interaktionsmöglichkeiten erweitern. So haben Forscher an der UC San Diego vor kurzem einen **VR**-Handschuh entwickelt, welcher virtuellen Tastsinn wiedergeben soll [14]. Wem dies aber zu unrealistisch ist, kann sich ein ähnliches System wie *Anvio VR* bauen: Dieses System wirbt damit, dem Nutzer komplett Bewegungsfreiheit zu geben. Zusätzlich wird durch die *Multiplayer*-Möglichkeit und durch das Ganzkörper-Tracking ein „*unglaublicher Level an Immersion erreicht*“(vgl. [15b]). Die Entwickler, *The Void*, sprechen sogar nicht mehr von **VR**, sondern von **Hyper Reality**. Ausgelegt ist das System für **VR**-Spielhallen[15].

Quellen:

[11] https://de.wikipedia.org/wiki/Liste_von_Spiel-Engines

[12a] <https://wiki.unrealengine.com/Vulkan>

[12b] [https://de.wikipedia.org/wiki/Vulkan_\(API\)](https://de.wikipedia.org/wiki/Vulkan_(API))

[13] <https://www.golem.de/news/steamvr-valve-zeigt-knuckles-controller-1706-128553.html>

[14] <http://www.vrnerds.de/vr-handschuh-reloaded-kuenstliche-muskelkammern-fuer-gefuehlsechte-interaktion/>

[15a] <https://vrodo.de/virtual-reality-spielhalle-the-void-eroeffnet-offiziell/>

[15b] <https://vr-world.com/anvio-vr-multiplayer-erfahrung-ganzkoerper-motion-tracking/>

Einsatzgebiete



(Quelle: <https://cdnwebqastorage.azureedge.net/dimensions/gb-reservations.jpg>)")

Kommerziell gesehen tritt **VR** meist in Form von Spielen auf. Im privaten Haushalt können diese durch Einsatz von **HMDs** immersiv verstärkt werden. Alternativ kann man in bestimmten Regionen **VR**-Spielhallen besuchen.[15]

In der Medizin kann sie eine sichere Alternative zur Behandlung von Phobien dienen: Die Konfrontationstherapie würde rein virtuell verlaufen, wobei der Patient sich selbst in einer sicheren Umgebung befindet. [17]

Im Militär findet sie unter anderem in einer Fallschirmspring-Simulation (*DoDAAM*) bereits Verwendung: Der Anwender kann den Sprung ohne gefährliche Konsequenzen lernen. [17]

Aber auch im Berufswesen findet sie zahlreiche Verwendung: *Arch Virtual* soll ein unterstützendes Tool zur 3D-Planung für Architekten sein. [17]

Bildungstechnisch kann jedes Simulationsspiel gezählt werden. Dabei können solche als *Serious Game*[16] aufgefasst werden. Beispielsweise kann *Microsoft Flight Simulator X* in Verbindung mit *VATSIM Germany*[19] eine gute Grundlage zum Pilotentraining sein. Als weiteres *Serious Game* kann man *ChernobylVR*[18] nennen. Hier kann man die Gegend um Chernobyl erkunden, ohne sich selbst den gefährlichen Strahlungen auszusetzen.[17]

Quellen:

[16] http://www.seriousgames.de/?page_id=160

[17] <https://www.androidpit.de/die-verschiedenen-einsatzgebiete-fuer-virtual-reality?nocol=1>

[18] <https://survivethis.news/chernobyl-vr-serious-gaming-fuer-guten-zweck-spenden/>

[19] http://www.vacc-sag.org/?PAGE=first_steps_pilot

Motion Sickness

Wem beim Rückwärtsfahren oder beim Lesen während der Autofahrt schlecht wird, weiß ungefähr worum es geht: **Motion Sickness** (fachl.: Kinetose) ist nichts anderes als die Reisekrankheit. Diese kann bei der immersiven Nutzung in der **VR** auch auftreten. Das Problem besteht darin, dass man visuell eine Bewegung wahrnimmt, das Innennetz jedoch diese Bewegung nicht bestätigt, weshalb der Körper von einer Vergiftung oder Halluzinationen ausgeht und diese zu bekämpfen versucht. Dies kann sich durch Symptome wie Schwindelgefühle, Übelkeit, Erbrechen, Schweißausbruch, Kopfschmerzen oder Orientierungsprobleme bemerkbar machen. Man spricht in dem Fall von der **Simulations- oder VR-Krankheit**. Dies kann unter anderem durch eine schlechte Implementierung der Bewegung oder durch schlechte Hardware verursacht werden. [20]

Tipps zum Verbeugen von Motion Sickness

1. Bildrate >90FPS (*frames per second*). Je niedriger die FPS, desto wahrscheinlicher wird die **Motion Sickness**
2. Latenz <20ms. Ähnlich wie bei niedrigen FPS führen hohe Latenzen zu Bildverzögerungen, die in **Motion Sickness** ausarten können.
3. Vermeidung schlechter Kameraführung.
4. Verschiedene Bewegungsmöglichkeiten bieten (Teleportation, Analog-stick).
5. Guten Rechner nutzen.
6. Fixpunkte suchen (bei häufigen Drehen und Wenden).
7. Pausen einlegen.
8. Bei Auftreten von Symptomen pausieren, bis Symptome verschwunden. [20]

Quellen:

[20] <https://vr-world.com/was-tun-bei-motion-sickness-in-vr/>

Augmented Reality

1 Einführung

2 Grundlegende Technologien

3 Konzept

4 Zusammenfassung

Projektleiter: Andrei Günter

Projektteam: Andrei Günter

Github-Repo: <https://github.com/aguentner/artranslator>

1 Augmented Reality

Dieses Dokument zeigt die Vorteile von Augmented Reality (AR) auf und beschreibt die bisherigen Fortschritte auf diesem Gebiet in Wissenschaft und Industrie. Weiterhin werden die fundamentalen Elemente eines AR-Systems dargestellt und zu einem Konzept zusammengefasst. Das Konzept spiegelt die wesentlichen Herausforderungen zur Realisierung eines AR-Systems wider und kann somit als generelles Framework für die Architektur eines AR-Systems betrachtet werden.

2 Einführung

Dieser Abschnitt leitet die Thematik AR ein, indem ein Beispiel einer AR-Anwendung einige Vorteile dieser Technologie aufzeigt. Im Anschluss werden bestehende AR-Systeme durch eine Unterteilung in verschiedene Aspekte in einen Überblick gebracht.

2.1 Motivation

Seit dem letzten Jahrzehnt erweckt AR großes Interesse in Wissenschaft und Industrie. AR stattet eine reale Umgebung mit virtuellen digitalen Informationen aus. Somit können Benutzer zusätzliche Informationen zu einem Gegenstand oder zu einer Umgebung abrufen.

Beispielsweise könnte eine AR-Anwendung bisherige Anleitungen zur Montage von Maschinen ersetzen und bereichern. Ein Problem bisheriger Anleitungen ist, dass eine Projektion der Inhalte auf die Realität nicht möglich ist. Somit sind Leser der Anleitung dazu gezwungen die Anweisungen über ihre Vorstellungskraft auf die reale Umgebung zu übertragen. Weiterhin können genannte Komponenten der Anleitung nicht direkt in der Realität auffindbar sein oder es fehlt eine geeignete Visualisierung, um diese Komponenten exakt und schnell zu lokalisieren.

Eine AR-Anwendung könnte Arbeiter in der Montage von Maschinen unterstützen, indem ein Monteur Projektionen von virtuellen Komponenten auf einer realen Maschine angezeigt bekommt. Darüber hinaus könnten diese virtuellen Komponenten so animiert werden, dass sie die Art und Weise der Anbringung repräsentieren. So lassen sich die Inhalte der Anleitung für die Montage direkt in der realen Umgebung visualisieren. Dem Monteur wird das Arbeiten durch entsprechende Projektionen erleichtert und der Prozess der Montage wird durch ein besseres Verständnis beschleunigt.

Weiterhin können neue Mitarbeiter durch intuitive AR-Anwendungen besser und schneller geschult werden. Neben den oben genannten animierten Visualisierungen können AR-Anwendungen jegliche Form von virtuellen Informationen annehmen, zum Beispiel auch reine textuelle Informationen. Dadurch lassen sich Begrifflichkeiten leicht mit den entsprechend visualisierten Prozessen verknüpfen. Folglich ist das Erlernen der Begrifflichkeiten für neue Mitarbeiter intuitiver und direkt mit der praktischen Erfahrung verbunden.

Das obige Beispiel durchleuchtet nur grob einige Vorteile von AR-Systemen und dient zur Veranschaulichung. Im Folgenden werden verschiedene Aspekte von bestehenden AR-Systemen beleuchtet.

2.2 Stand der Technik

AR wird besonders in den Bereichen Gaming, Navigation, Guiding und in interaktiven Medien verwendet [1, 2]. Bestehende AR-Systeme unterscheiden sich grundsätzlich in der Verwendung von Geräten zur Anzeige von virtuellen Inhalten sowie in der Darstellung der virtuellen Inhalte. Wie in Abbildung 1 dargestellt, unterscheiden sich die Geräte wie folgt [2]:

- *Hand-Held*: Die virtuellen Inhalte werden über ein Gerät dargestellt, welches der Benutzer in der Hand hält (z.B. Smartphone oder Tablet).
- *Head-worn*: Die virtuellen Inhalte werden über ein Gerät dargestellt, welches der Benutzer auf dem Kopf trägt (z.B. Smart-Glass oder Head-Mounted-Display).
- *Spatial*: Die virtuellen Inhalte werden direkt in die reale Welt projiziert (z.B. Beamer), hierbei ist der Benutzer nicht auf das Tragen eines Gerätes angewiesen.

Weiterhin werden in Abbildung 1 verschiedene Formen der Darstellung von virtuellen Inhalten angeführt [2]:

- *Video*: Die realen Objekte und die virtuellen Inhalte sind miteinander verschmolzen und die Sicht des Benutzers findet komplett auf einer digitalen Ebene statt.
- *Optical*: Über eine Perspektive auf die reale Welt werden virtuelle Inhalte auf reale Objekte überlagert.
- *Retinal*: Virtuelle Inhalte werden über einen schwachen Laser direkt auf die Netzhaut des Benutzers projiziert.
- *Projector*: Virtuelle Inhalte werden direkt auf reale Objekte projiziert.
- *Hologram*: Virtuelle Inhalte werden als Hologramm dargestellt.

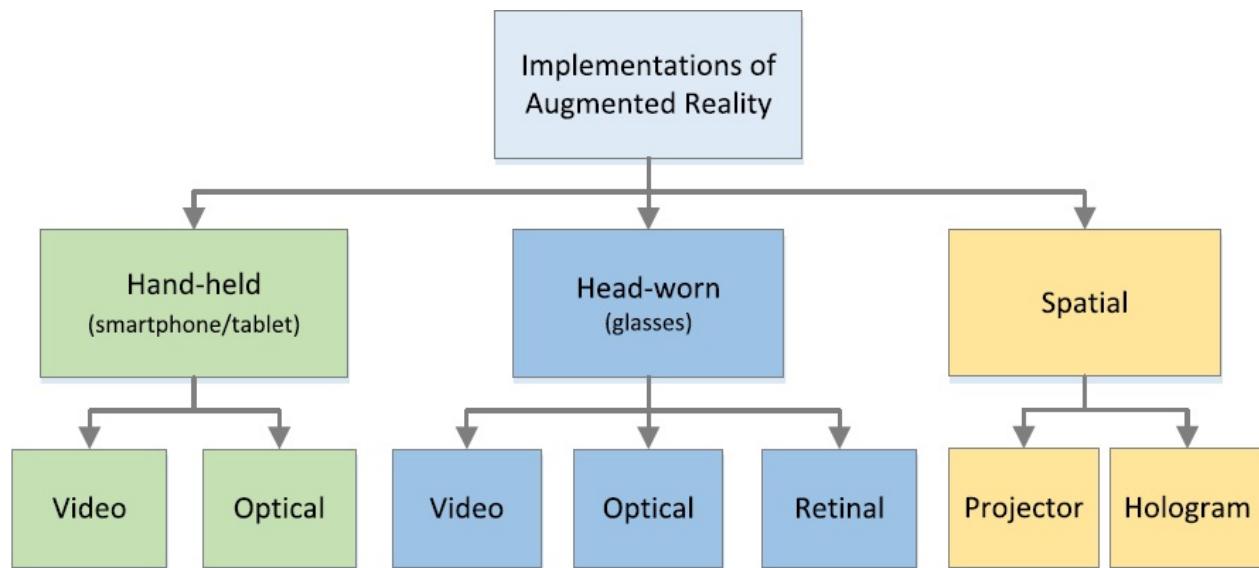


Abbildung 1: Geräte zur Anzeige und zur Darstellung von virtuellen Inhalten in AR-Systemen [2]

3 Grundlegende Technologien

Dieser Abschnitt erläutert die für AR notwendigen Technologien im Detail. Dazu wird zu Beginn eine Definition für AR erarbeitet. Anschließend folgt eine Beschreibung drei fundamentaler Bausteine von AR-Systemen: Tracking, Darstellung und Interaktion [3].

3.1 Augmented Reality

Es ist allgemein anerkannt, dass T. P. Caudell und D. W. Mizell den Begriff Augmented Reality mit ihrer Arbeit für ein Head-Mounted-Display begründeten [4]. Das Head-Mounted-Display wurde als Prototyp eines Assistenzsystems für Monteure im Flugzeugbau konzipiert.

Während der letzten Jahrzehnte haben sich weitere Ansätze etabliert und damit hat sich auch die Definition von AR stetig verändert. Daher wird für dieses Dokument eine Abgrenzung von herkömmlichen AR-Systemen zu pervasiven AR-Systemen vorgenommen. Herkömmliche AR-Systeme sind Prototypen für speziell zugeschnittene Szenarien und auch nur in diesem Kontext verwendbar, während pervasive AR-Systeme ein breites Einsatzgebiet abdecken und je nach Kontext die Sichtweise und den virtuellen Inhalt für den Benutzer anpassen [1]. Für diese Abgrenzung wird folgende Definition herangezogen:

- **Pervasive AR:** Pervasive Augmented Reality ist eine kontinuierliche und allgegenwärtige Benutzerschnittstelle, welche eine dreidimensionale physikalische Welt mit digitalen Informationen erweitert, während der Kontext des Benutzers beobachtet wird und der virtuelle Inhalt vom Kontext abhängig angepasst wird [1].

3.2 Tracking

Um virtuelle Inhalte in einer dreidimensionalen realen Umgebung an den passenden Positionen darstellen zu können, nutzen AR-Systeme unterschiedliche Tracking-Methoden. Die einfachste Form des Trackings nutzt auf realen Objekten angebrachte Marker, z.B. in Form von QR-Codes. Ein großer Nachteil ist jedoch, dass der virtuelle Inhalt vordefiniert auf die Marker angepasst werden muss. Darüber hinaus ist die Anbringung von Markern auf realen Objekten nicht immer möglich.

Bei der Verwendung eines mobilen Gerätes (bspw. hand-held oder head-worn) schafft hier das 3D Tracking Abhilfe. Hierbei ist das Tracking für die Verfolgung der Trajektorie des Gerätes und zur Bestimmung der Pose des Gerätes zuständig. Wenn gleich aussehende

Objekte unterschieden werden müssen, ist die Trajektorie und die Pose des Gerätes von großer Bedeutung. Am Beispiel einer Maschine lässt sich dies mit mehrfach vorkommenden Komponenten verdeutlichen. Wenn eine AR-Anwendung einen Mitarbeiter bei der Wartung dieser Komponenten unterstützt, so ist es von immenser Bedeutung, dass keine Komponenten verwechselt werden.

Insbesondere bei der Verwendung von mobilen Geräten, haben sich das Tracking mit GPS [5] und das kamerabasierte Tracking etabliert [6, 7].

3.3 Darstellung

Die Darstellung ist ein weiteres Kernelement in AR-Systemen. Die richtige Kalibrierung der Position und die richtige Ausrichtung von virtuellen Inhalten bestimmt die Zufriedenheit des Endanwenders.

Virtuelle Inhalte können wie im Abschnitt 2.2 beschrieben dargestellt werden. Im letzten Jahrzehnt wurden überwiegend Head-mounted-Displays verwendet [3]. Jedoch sind hand-held- und head-worn-basierte Ansätze oftmals unkomfortabel. Ein weiterer Nachteil bei head-worn-basierten Ansätzen liegt in den geringen Akkulaufzeiten der aktuell verfügbaren Geräte.

Ansätze über Projektionen oder Hologramme sind sehr benutzerfreundlich, jedoch nur schwer in dynamischen Umgebungen umzusetzen und mit hohen Kosten verbunden.

3.4 Interaktion

Der virtuelle Inhalt passt sich dem Kontext des Benutzers an, es findet also eine Interaktion mit dem System statt. Ob das System diese Anpassung automatisch vornimmt oder der Benutzer eine Eingabe tätigen muss ist hier nicht relevant. Bereits eine Bewegung eines Benutzers oder das Verändern der Umgebung kann zu einem neuen Kontext führen, diesen Kontext muss das AR-System beobachten und entsprechend reagieren. Ein Beispiel einer Interaktion kann ein von einem Projektor angezeigter virtueller Schalter sein, der sich betätigen lässt wenn der Benutzer seine Hand über den Schalter hält.

Das kontinuierliche Beobachten und das Reagieren auf den Kontext des Benutzers wird als Ubiquitous Computing bezeichnet, während die Interaktion aus obigen Beispiel die Bezeichnung tangible bits trägt [3].

Ein weiterer Schlüssel für eine benutzerfreundliche Interaktion und somit auch für die Akzeptanz von AR-Systemen, stellt das Verständnis für soziale, psychologische, kulturelle und organisatorische zwischenmenschliche Interaktionen dar [3]. So könnte bspw. eine AR-Anwendung entwickelt werden, mit der es möglich ist aus sprachlichen Informationen emotionale Informationen zu gewinnen und diese in einer für den Menschen verständlichen Form darzustellen [3].

Auch in der Interaktion führen hand-held-basierte Ansätze zu Herausforderungen in Bezug auf die Benutzerfreundlichkeit. Die Darstellung von virtuellen dreidimensionalen Objekten auf einer zweidimensionalen Schnittstelle (bspw. Tablet) kann in einigen Fällen die Benutzer verwirren.

4 Konzept

Dieser Abschnitt bringt die zuvor erläuterten Technologien in eine konzeptionelle Übersicht. Das hier vorgestellte Konzept kann als Framework zur Entwicklung von Architekturen für AR-Systeme dienen und wird als *AR-Basis-Zyklus* bezeichnet. Die Abbildung 2 veranschaulicht die notwendigen Herausforderungen, um ein AR-System zu realisieren. Hierbei greifen die drei dargestellten Basis-Elemente in einem sich wiederholenden Zyklus ineinander. Das Tracking bestimmt die für die Darstellung notwendigen Positionen von virtuellen Inhalten. Während diese Prozesse erfolgen, kann sich der Kontext für die virtuellen Inhalte durch eine Interaktion verändern. Diese Interaktion hat unmittelbaren Einfluss auf die Umgebung des AR-Systems und erfordert ein kontinuierliches Tracking.

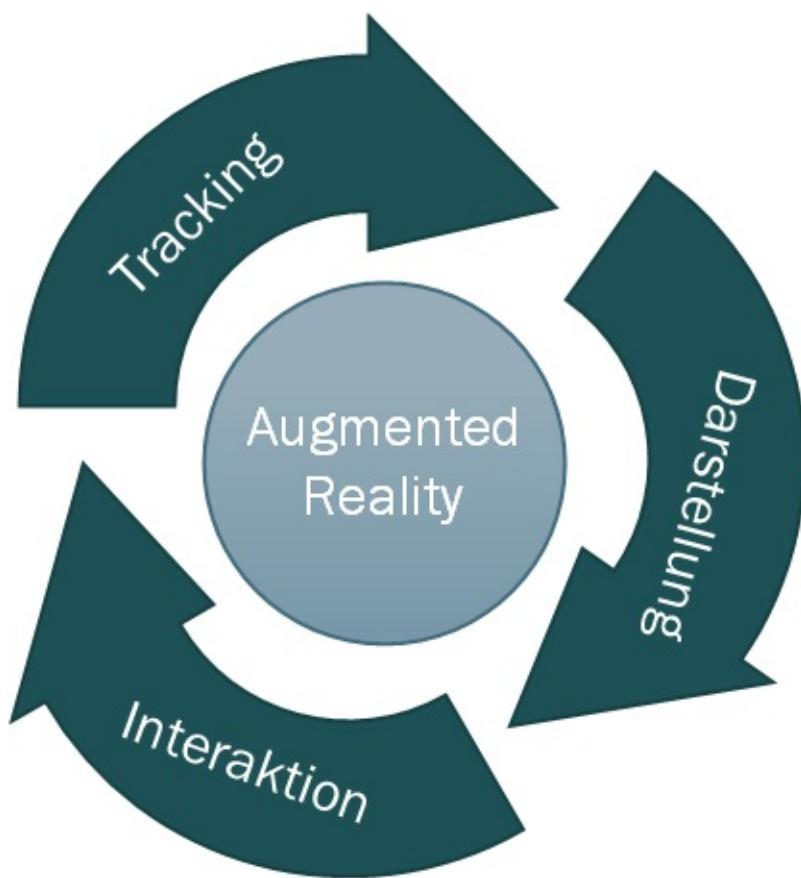


Abbildung 2: AR-Basis-Zyklus

Zur Umsetzung dieses Konzepts stehen frei verfügbare sowie kommerzielle Werkzeuge zur Verfügung. Die folgende Übersicht stellt einige bekannte AR Software Development Kits (SDK) in Relation zu folgenden Aspekten:

- Tracking: siehe Abschnitt 3.2
- Darstellung: siehe Abschnitt 3.3

- Interaktion: siehe Abschnitt 3.4
- Lizenz: stellt dar, ob eine freie oder kommerzielle Lizenz zur Verfügung steht
- Dokumentation: Bewertung der Ausführlichkeit der Dokumentation

Für jeden Aspekt wird ein Symbol vergeben, welches darstellt wie gut das SDK den Entwickler im entsprechenden Aspekt unterstützt:

- „+“: vorhanden
- „-“: eingeschränkt vorhanden
- „x“: nicht vorhanden
- „?“: keine Informationen vorhanden

SDK Name	Tracking			Darstellung	Interaktion	Lizenz		Dokumentation
	Marker	3D Tracking	GPS			frei	kommerziell	
Metaio SDK	+	+	+	+	+	x	x	+
ARToolkit	+	x	x	-	-	+	x	-
Vuforia	+	-	+	+	+	-	+	+
Wikitude	+	+	+	+	+	-	+	+
DroidAR	+	+	+	?	+	x	+	?
D'Fusion	+	x	+	+	+	x	+	+

Tabelle 1: Übersicht AR-SDK's

Modell getriebene Systementwicklung

Projektleiter:

Projektteam: Jonathan Jansen, Oliver Nagel

Github-Repo:

Einführung

Die aktuelle Marktentwicklung stellt Unternehmen vor immer größere Herausforderungen, da zunehmend qualitativ hochwertigere und komplexere Systeme zu geringeren Kosten und kürzeren time-to-market Zeiten gefordert werden. Zusätzlich steigt die Erwartung des Kunden, dass Systeme kundenspezifisch angepasst werden können (mass customization) und über den Produktlebenszyklus hinweg mit kontinuierlichen Systemverbesserungen versorgt werden.

Diese Anforderungen sorgen besonders in einer dokumentenbasierten Entwicklung für große Probleme. Die Pflege von Dokumenten eines komplexen Systems ist sehr zeitintensiv und fehleranfällig. Eine Änderung am System wird oft nicht konsistent in allen relevanten Dokumenten übernommen, da oft unklar ist, welche Dokumente oder Stellen im Dokument geändert werden müssen. Infolgedessen werden die Dokumente nicht mehr gepflegt und das Wissen bleibt im Kopf der einzelnen Entwickler. Weitere Änderungen sind dann oft risikoreich und zeitaufwendig, da ein ganzheitliches Verständnis des Systems nicht mehr existiert und die Informationen mühsam von verschiedenen Personen zusammengetragen werden müssen. Ein weiteres Problem ist, dass diese Dokumente nicht zentral gepflegt werden, sondern in der Realität oft verteilt auf verschiedenen Netzlaufwerken oder Computern der Entwickler liegen. Es fehlt der Überblick über die relevanten Dokumenten.

Eine weitere Herausforderung stellt das Variantenmanagement für Unternehmen dar. Es fehlt den Unternehmen oft an einer effizienten Methodik, die Komplexität der Varianten zu beherrschen und Varianten innerhalb einer Produktlinie effizient zu entwickeln.

Um den Anforderungen gerecht zu werden, versuchen Unternehmen zunehmend ihre Entwicklung von einer dokumentbasierten auf eine modellbasierte Systementwicklung umzustellen. Die modellbasierte Systementwicklung wird auch als Model-Based-Systems-Engineering (MBSE) bezeichnet. MBSE beschreibt einen interdisziplinären Ansatz des Systems Engineering, welche bewährte Vorgehensweisen des traditionellen Systems Engineering mit Modellierungstechniken verbindet. Der Entwurf, die Spezifikation, sowie die Verifikation und Validierung eines komplexen Systems erfolgt in einem Systemmodell. Dieses Systemmodell begleitet den kompletten Entwicklungsprozess und stellt die Quelle der wesentlichen Entwicklungsartefakte dar.

Der Übergang zu MBSE stellt für viele Unternehmen jedoch eine Herausforderung dar, da die modellbasierte Entwicklung eine bedeutsame Veränderung für die Belegschaft in der Entwicklungsorganisation ist. Die nachfolgenden Kapitel sollen einen Einstiegspunkt in die Thematik liefern und einen Überblick über relevante Themen von MBSE und Variantenmanagement geben. Die ersten beiden Kapitel geben eine Einführung in das

Systems Engineering und MBSE. Im dritten und vierten Kapitel wird die SysML vorgestellt, welche eine standardisierte Notation für die Modellierung von Systemarchitekturen liefert und domainspezifisch angepasst werden kann. Im fünften Kapitel wird eine Methodik zur effizienten Entwicklung von Produktlinien beschrieben. Das letzte Kapitel stellt die Modelltransformation vor. Diese wird benötigt um Modelle einer Disziplin (z.B. Systemarchitektur) konsistent in Modelle einer anderen Disziplin (z.B. Software, Hardware) zu überführen.

Systems Engineering

Das Systems Engineering (SE) ist ein interdisziplinärer Ansatz mit dem Ziel Systeme erfolgreich umzusetzen. Dabei werden alle spezialisierten Teams in ein gemeinsames Team integriert und ein strukturierter Entwicklungsprozess geformt, der die Systemkonzeptionierung bis hin zur Inbetriebnahme umfasst. SE weist der Definition der Kundenbedürfnisse und der geforderten Funktionalität von Beginn an einen hohen Stellenwert zu, um qualitativ hochwertige Produkte zu entwickeln, die im Einklang mit diesen Bedürfnissen stehen. [INC17a](#)

Systems Engineering basiert auf System Thinking. Das System Thinking ist eine spezielle Sichtweise auf die Realität. Sie schärfst die Sicht auf das Ganze und wie die Teile innerhalb des Ganzes zusammenwirken. Im Systems Engineering wird System als das Ganze und die einzelnen Systemelemente als Teile betrachtet. [INC17b](#)

Ein System kann als eine Kombination aus Systemelementen betrachtet werden, die zusammen Ergebnisse liefern, die sie alleine nicht erzielen könnten. Unter Ergebnissen eines Systems werden Eigenschaften, Funktionen, Charakteristiken und Verhalten verstanden, die das System aufweist. Die Elemente können dabei durch Menschen, Hardware, Software, Hilfsmittel, oder ähnliches repräsentiert werden. Alle Dinge, die benötigt werden, damit ein System Ergebnisse liefern kann. [INC17a](#)

Beispielsweise kann ein Systemelement durch das Wartungspersonal repräsentiert werden. Das Wartungspersonal ist Teil eines größeren Systems und wird benötigt, damit ein Systemelement (z.B. eine Anlage) über den Lebenszyklus hinweg eine bestimmte Funktion erfüllen kann.

Der Ansatz des System Thinking hilft ein besseres Verständnis von einem System zu erlangen. Ein System Thinker weiß,

- wie das System sich in einen größeren Kontext einzuordnen ist,
- wie es sich verhält und
- wie mit dem System umzugehen ist [INC17b](#).

Eine wichtige Norm im Bereich des Systems Engineering ist die ISO/IEC 15288 – „System- und Software-Engineering - System-Lebenszyklus-Prozesse“ und bildet den Einstiegspunkt für ein ernsthaftes Systems Engineering.

Die ISO/IEC 15288 ist internationale Norm, welche die Lebenszyklusprozesse eines Systems beschreibt. Sie hat das Ziel diese Prozesse bewertbar zu machen und zu verbessern. Die Lebenszyklusprozesse lassen sich in folgende übergeordnete Kategorien

unterteilen:

Agreement Prozesse

Die *Agreement Prozesse* definieren die notwendigen Aktivitäten für das Erstellen von Verträgen oder Vereinbarungen zwischen zwei Organisationen.

Technical Management Prozesse

Die *Technical Management Prozesse* umfassen die Erstellung, Entwicklung und Ausführung von Plänen, der Abgleich des aktuellen Fortschritts und der Zielerreichung mit diesen Plänen, sowie die Kontrolle der Ausführung, um das Ziel zu erreichen.

Technical Prozesse

Die *Technical Prozesse* umfassen die Definition der Anforderungen, die Umwandlung der Anforderungen in ein konkretes Produkt, die Sicherstellung der Reproduktion eines Produktes, die Benutzung des Produktes, sowie die zuverlässige Gewährleistung der Betriebsbereitschaft und die Entsorgung des Produktes.

Organizational Project-Enabling Prozesse

Die *Organizational Project-Enabling Prozesse* stellen sicher, dass das Unternehmen Produkte liefern kann. Sie stellen die Ressourcen und Infrastruktur bereit, damit Projekte gestartet, unterstützt und kontrolliert werden können.

Die in der Norm beschriebenen Prozesse und Lebenszyklusmodelle müssen an das Unternehmen und den Projekten spezifisch angepasst werden. Diesen Schritt der Anpassung wird auch als *tailoring* bezeichnet. [INC15]

Das Lesen der Norm kann ein zäher Akt sein. Die Incose hat aus diesem Grund das SE-Handbook veröffentlicht, welches eine hohe Konformität zu der Norm aufweist. Das SE-Handbook kann zusätzlich als Schulungsmaterial für die Zertifizierung des „Certified Systems Engineer“ genutzt werden.

Model Based Systems Engineering

Dieses Kapitel beschäftigt sich mit dem Model Based Systems Engineering (MBSE). Anhand von Definitionen und Erläuterungen wichtiger Fachbegriffen, die in Verbindung mit MBSE stehen, soll der Leser ein besseres Verständnis erlangen.

Das Model Based Systems Engineering setzt sich aus den Begriffen „Systems Engineering“ und „Model“ zusammen. Das Modell wird im Kontext von MBSE auch als Systemmodell bezeichnet. Dieses Systemmodell ist eine Abstraktion eines komplexen Systems. Die Abstraktion hilft die Komplexität eines Systems zu reduzieren, sodass Vorgänge und Funktionen einfacher dargestellt, verstanden und gestaltet werden können. [THM17](#)

Das Systemmodell enthält alle relevanten Informationen eines Systems für die verschiedenen Stakeholder (z.B. Projektleiter, Entwicklungsleiter, Testingenieur) und ist die primäre Informationsquelle während der kompletten Entwicklung. Die unterschiedliche Darstellung dieser Informationen wird als Sicht (engl. View) bezeichnet und richtet sich nach den Interessen der Stakeholder. Weiterhin kann das Systemmodell wichtige Informationen für Entscheidungen liefern oder für frühe Analysen (z.B. FMEA, Risikoanalyse) genutzt werden. Ein entscheidender Vorteil, der sich durch die Verwendung eines Systemmodells ergibt, ist jedoch die Wiederverwendung von Elementen (z.B. Komponenten). Diese sind im Modell einmal hinterlegt und können in unterschiedlichen Systemarchitekturen visualisiert werden. Eine Änderung an diesem Element sorgt für eine Aktualisierung in allen Systemarchitekturen, die dieses Element verwenden.

Ein Systemmodell zeichnet sich durch folgende Eigenschaften aus:

- Das Modell darf sich aus mehreren Repositories zusammensetzen, solange diese in sich konsistent sind und sich nach außen wie ein einzelnes Modell repräsentieren.
- Das Modell erlaubt unterschiedliche Sichten auf die Informationen.
- Das Modell ist maschinell auswertbar und liegt in einer abstrakten Syntax vor, die explizit MBSE-Konzepte wie Anforderungen oder Systemarchitekturen unterstützt.

[THM17](#)

Die nachfolgenden Abbildung zeigt mögliche beteiligte Rollen in einem Systems Engineering Ansatz, die an einem gemeinsamen Systemmodell arbeiten. Die Rollenbezeichnungen und Prozesse können von Unternehmen zu Unternehmen variieren.

Der Produktmanager oder Kunde auf der linken Seite der Abbildung übergibt dem Systemanalytiker die Anforderungen in einem Lastenheft oder über ein ReqIF-File. ReqIF steht für Requirements Interchange Format und ist ein standardisiertes XML-Dateiformat von der Object Management Group, mit welchem Anforderungen inkl. ihrer Metadaten

zwischen verschiedenen Requirements Management Tools ausgetauscht werden können. Der Systemanalytiker hat die Aufgabe diese Anforderungen zu analysieren, abzustimmen und zu verfeinern. Zusätzlich muss er die Anforderungen von weiteren Stakeholder (z.B. Gesetzgebung) berücksichtigen. Die Analyse der Anforderungen erfolgt durch die Modellierung von Use Cases bis hin zu einem umfangreichen Domänenmodell. Die textuellen Anforderungen in dem Requirements Management Tool können dazu als grafische Elemente in ein Modellierungstool importiert werden. Für die Modellierung von Systemen eignet sich die Notation SysML der Object Management Group, welche Teile der UML-Notationselemente übernimmt und zusätzlich um domänenspezifische Sprachelement erweitert.

Auf Basis der verfeinerten Anforderungen kann der Testingenieur seine Testfälle erstellen und der Systemarchitekt das System genauer spezifizieren. Sowohl die Beschreibung der Testfälle als auch die Spezifizierung des Systems erfolgt erneut in SysML. Die einzelnen Komponenten, die während der Spezifizierung vom Systemarchitekt identifiziert wurden, werden anschließend jeweils an die Experten-Teams für Software, Hardware und Mechanik / Konstruktion weitergegeben.

Die Teams haben die Aufgabe diese Komponenten mit ihren Werkzeugen weiter zu detaillieren. Damit die Teams mit ihren Werkzeugen an einer Komponente weiterarbeiten können, kann es notwendig sein, dass das SysML-Modell durch Modelltransformation in ein anderes Modell übersetzt werden muss.

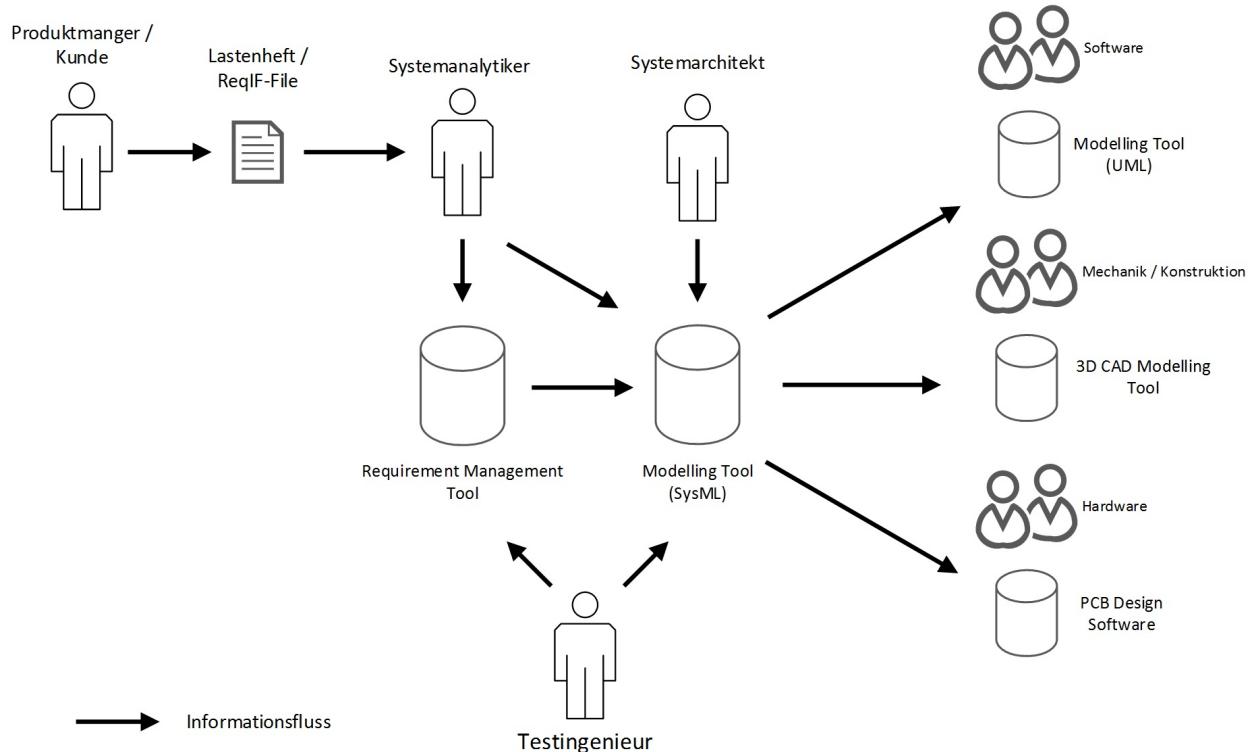


Abbildung 1: Vereinfachte Darstellung der beteiligten Rollen

Obwohl die vorgestellte Vorgehensweise stark an das Wasserfallmodell erinnert, handelt es sich in der Realität um einen stark iterativen Entwicklungsprozess mit vielen Feedbackzyklen. Sobald z.B. die Anforderungen in ausreichender Qualität vorliegen, können erste Entwürfe der Systemarchitektur entstehen und Lösungsideen mit Fachexperten abgestimmt werden. In den Experten-Teams kann ein agiles Vorgehen angestrebt werden.

Systems Modeling Language

Die Systems Modeling Language (SysML) ist eine allgemeine grafische Modellierungssprache um komplexe Systeme zu spezifizieren, analysieren, designen und verifizieren. Die erste Version von SysML wurde von der Object Management Group 2007 angekündigt und liegt bis dato in der Version 1.5 vor. Komplexe Systeme umfassen in der Regel Hardware, Software, Informationen, Personal, Abläufe und technische Hilfsmittel. Die SysML liefert eine grafische Repräsentation mit einer Semantik um Systemanforderungen, Verhalten, Struktur und Zusicherungen eines Systems zu modellieren. Sie enthält eine Teilmenge der UML-Elemente und zusätzlich Erweiterungen, um den Anforderungen des System Engineering gerecht zu werden (siehe Abbildung 2).

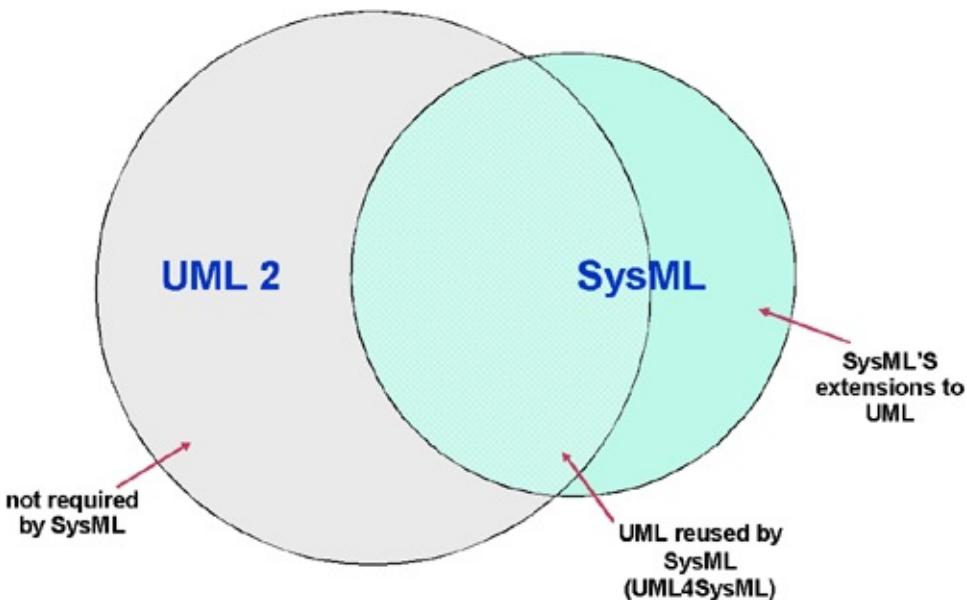
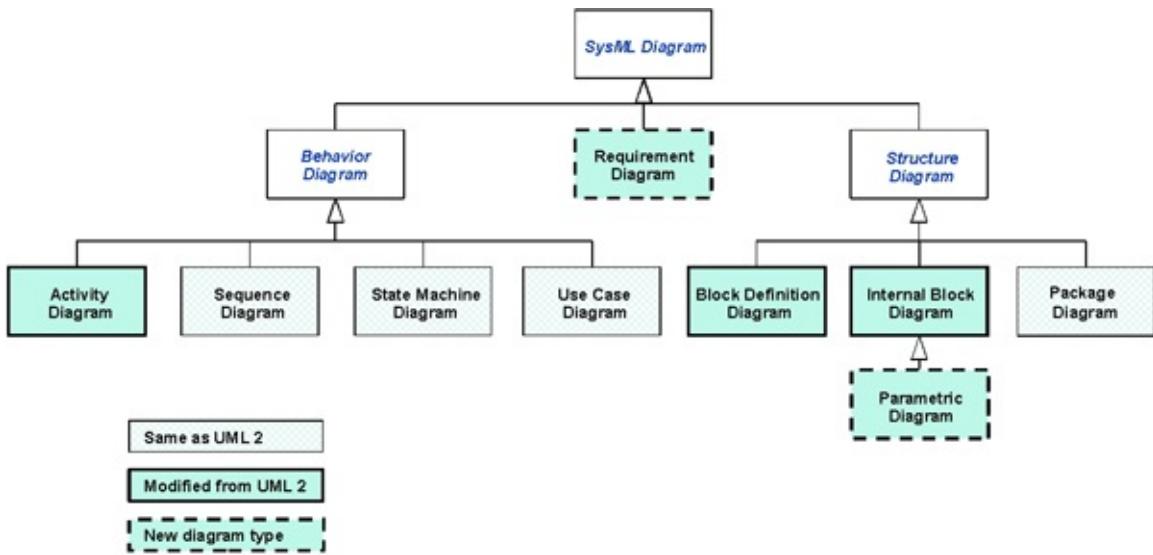


Abbildung 2: SysML und UML Teilmengen [OMG17](#)

SysML-Diagramme

Die SysML umfasst insgesamt neun Diagramme. Das Use Case-, Sequence-, State Machine- und Package Diagramm wurden von der UML übernommen. Das Activity-, Block Definition- und Internal Block Diagramm wurden auch von der UML übernommen, enthalten jedoch Modifikationen. Das Requirement Diagramm und Parametric Diagramm sind neu in der SysML. Abbildung 3 zeigt die Hierarchie der Diagramme.

Abbildung 3: SysML Diagrammtypen [OMG17](#)

Requirement Diagramm

Das Requirement Diagramm wird zur Visualisierung von text-basierten Anforderungen und deren Beziehung zu anderen Modelementen verwendet. Es können Beziehungen zwischen den Elementen hergestellt werden, um z.B. zu zeigen, dass eine Anforderung durch ein anderes Element (z.B. ein Systemelement) erfüllt wird oder das ein Testfall eine Anforderung verifiziert. Zusätzlich können Anforderungshierarchien oder Anforderungsableitungen dargestellt werden. [OMG17](#)

Block Definition Diagramm

Das Block Definition Diagramm wird verwendet um Systemhierarchien zu definieren. Das Basiselement ist der Block, welcher als Repräsentant von Hardware, Software, Hilfsmittel, Personal oder anderen Systemelementen dient. [OMG17](#)

Internal Block Diagramm

Das Internal Block Diagramm beschreibt die interne Struktur eines Blocks mit seinen Teilen (parts), Schnittstellen(ports) und den Verbindungen (connectors) in einem bestimmten Kontext. [OMG17](#)

Package Diagramm

Das Package Diagramm dient zur Darstellung der Organisation / Struktur eines Models. Es eignet sich sehr gut, um eine Navigation durch das Modell herzustellen. [OMG17](#)

Parametric Diagramm

Das Parametric Diagramm beschreibt die Beschränkungen (engl. constraints) der Systemeigenschaften wie z.B. Performanz, Zuverlässigkeit oder zulässige Eigenschaften und ist ein Hilfsmittel für die Analyse eines Systems. [OMG17](#)

Use Case Diagramm

Das Use-Case Diagramm liefert eine High-Level Beschreibung der Funktionalität, die das System seiner Umwelt bereitstellt.[OMG17](#)

Activity Diagramm

Das Activity Diagramm zeigt den Daten- und Kontrollfluss von verschiedenen Abläufen im System.[OMG17](#)

Sequence Diagramm

Das Sequence Diagramm zeigt die Interaktion zwischen dem System und seiner Umwelt oder zwischen verschiedenen Systemelementen.[OMG17](#)

State Machine Diagram

Das State Machine Diagramm beschreibt Zustandsübergänge und Aktionen, die ein System oder Systemelement eine Antwort auf Events durchführt. [OMG17](#)

Cross-cutting

Im Bereich des Systems Engineering wird oft der Begriff „Allokation“ verwendet. Dieser meint die Zuweisung von Funktion zu Komponenten, logische zu physikalische Komponenten oder Software zu Hardware. Die SysML stellt dafür eine spezielle Relationship „allocate“ bereit. [OMG17](#)

Metamodeling in SysML / UML

Die Modellierungssprachen wie UML, SysML oder die BPMN, welche durch die Object Management Group (OMG) definiert wurden, basieren auf dem Konzept von sprachbasierten Metamodellen. Ein sprachbasiertes Meta-Modell stellt die Elemente einer Modellierungssprache und ihre Beziehungen in einem Modell dar. [WIKI17c](#)

Den Meta-Modellen der OMG liegt die Meta Object Facility (MOF) zu Grunde. Die MOF beschreibt eine Metadaten-Architektur, dessen Kernbestandteil das Meta-Meta-Modell ist. Sie wurde mit dem Ziel definiert, eine gemeinsame Grundlage für verschiedene Meta-Modelle zu definieren. Dieser Ansatz bietet einige Vorteile, da verschiedene Meta-Modelle dadurch gleichzeitig persistiert und verarbeitet werden können. Die Modelltransformation ist eine Art der Verarbeitung von Modellen. [WIKI17d](#)

Abbildung 4 zeigt beispielhaft den Zusammenhang der verschiedenen Ebenen der (Meta-)Modelle anhand der UML.

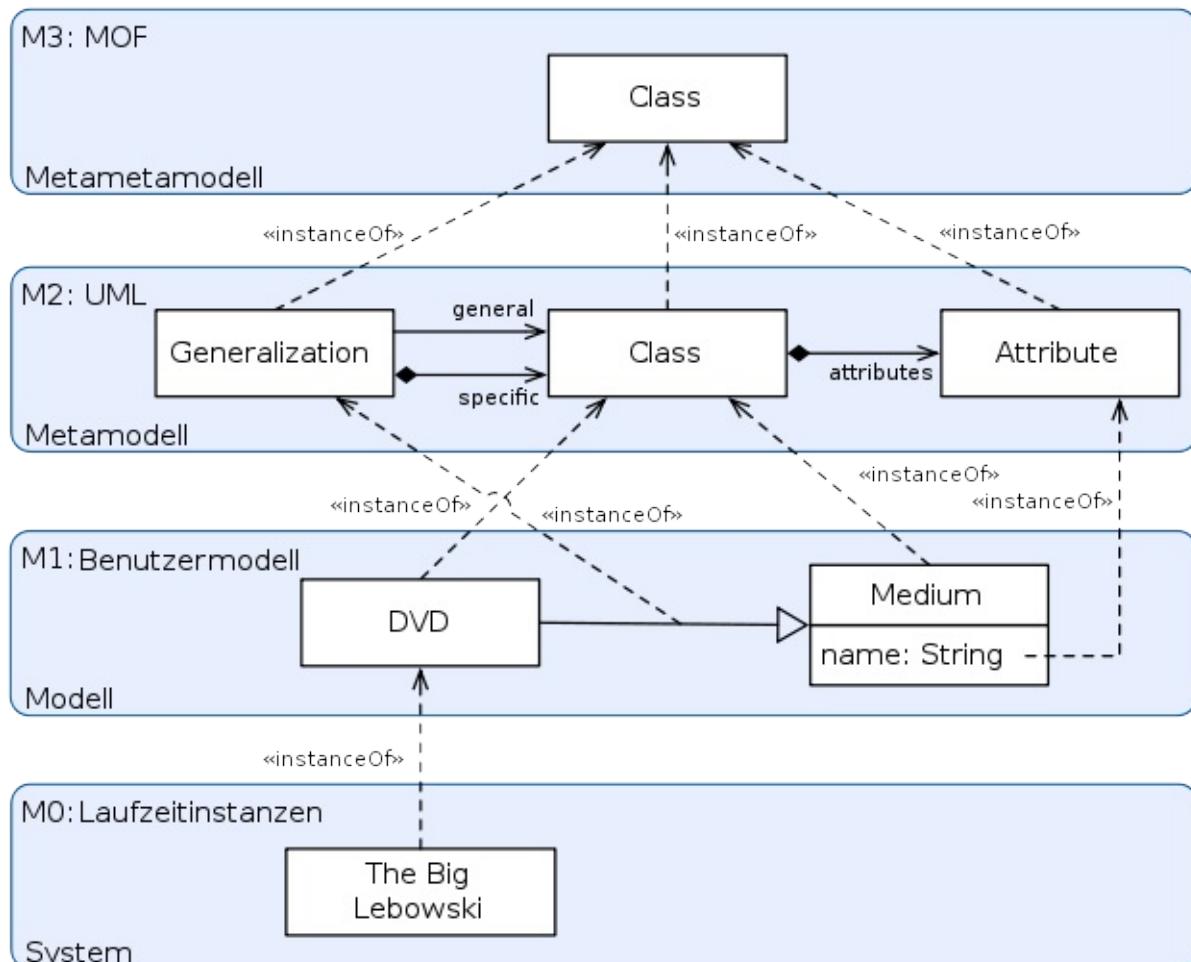


Abbildung 4: Ebenen der Meta-Modelle [WIKI17f](#)

Profil-Mechanismus

Um eine Modellierungssprache wie UML oder SysML um domänenspezifische Sprachelemente zu erweitern, stellt die OMG einen Profil-Mechanismus bereit. Ein Profil ist eine konkrete Erweiterung eines Metamodells. Ein leichtgewichtiger Mechanismus für die Erweiterung eines Metamodells sind Stereotypen. Ein Stereotyp spezialisiert eine Metaklasse für ein spezifisches Einsatzgebiet.

Für jeden Stereotypen können Stereotypeigenschaften definiert werden, in denen domänenspezifische Informationen gespeichert werden können. Der Profil-Mechanismus wird im Folgenden anhand von Beispielen erläutert.

Die folgende Abbildung erweitert die Metaklasse *Actor* um den Stereotypen *externalSystem*. Diese Erweiterung kann genutzt werden, um bei einem Use-Case-Diagramm die Akteure stärker zu differenzieren.

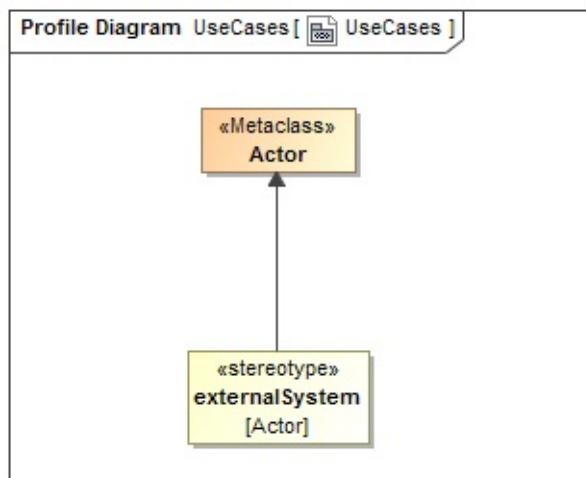


Abbildung 5: Erweiterung der Meta-Klasse Akteur

In Abbildung 6 wird die konkrete Anwendung eines *externalSystems* in einem Use-Case-Diagramm gezeigt. Das Diagramm zeigt Use-Cases für ein Smart Home System. In dem Use-Case „Jalousien mit Smartphone steuern“ ist der Akteur „Bewohner“ und das *externalSystem* „Jalousie“ beteiligt.

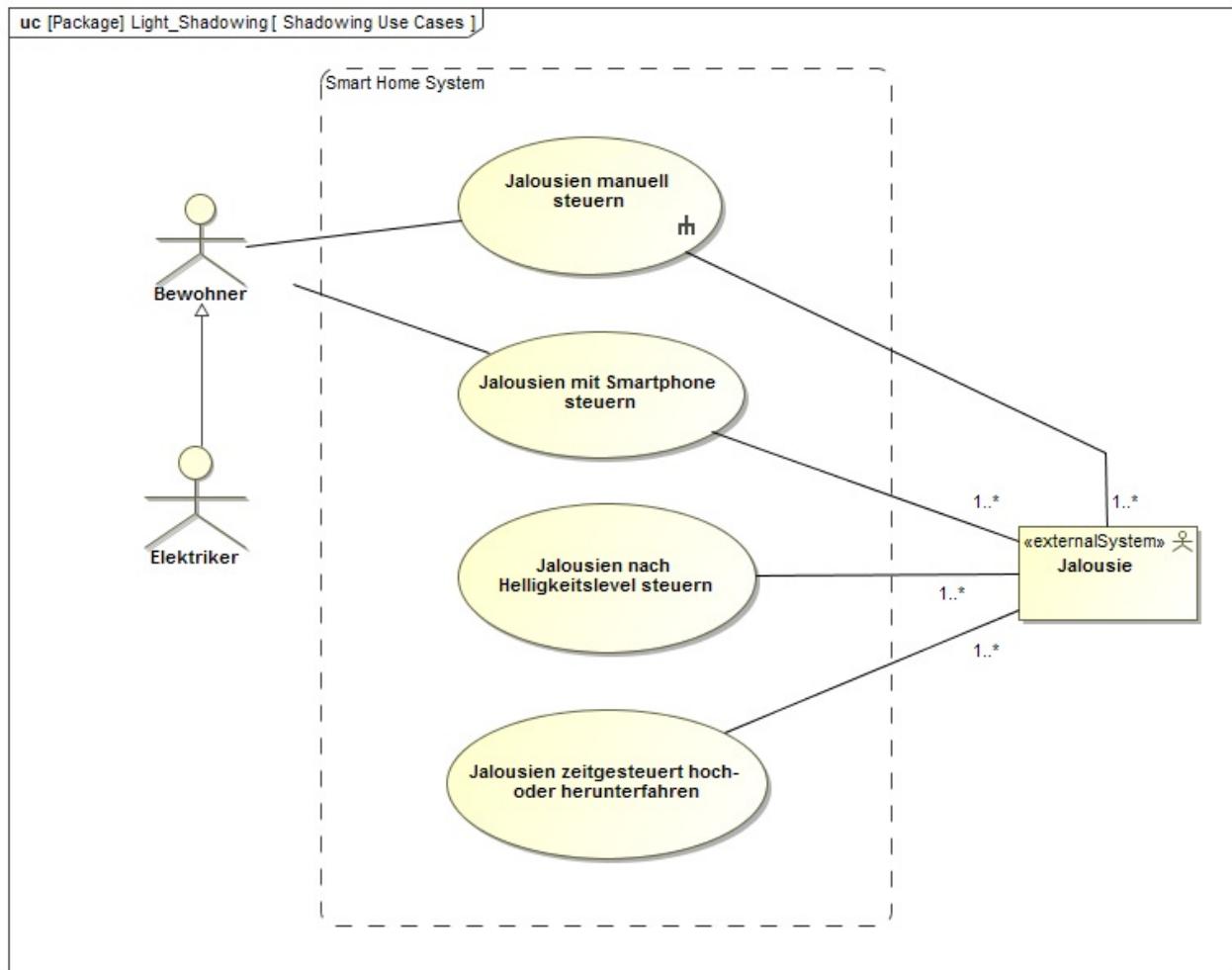


Abbildung 6: Konkrete Anwendung der Aktuer-Erweiterung

Ein Stereotyp kann von anderen Stereotypen mit Hilfe einer „Generalisierung“ spezialisiert werden. In Abbildung 7 ist der Stereotyp „Requirement“ von der Metaklasse *Class* erweitert worden. Der Stereotyp besitzt zusätzliche Eigenschaften wie z.B. *Id* und *Text*. Von „Requirement“ wurde ein spezialisierter Stereotyp „ExtendedRequirement“ erzeugt, der alle Eigenschaften von „Requirement“ besitzt und zusätzlich die Attribute *obligation* und *priority* enthält. Für die bessere Differenzierung von Anforderungen wurden von „ExtendedRequirement“ die drei Anforderungstypen „Business Requirement“, „System Requirement“ und „Design Requirement“ abgeleitet.

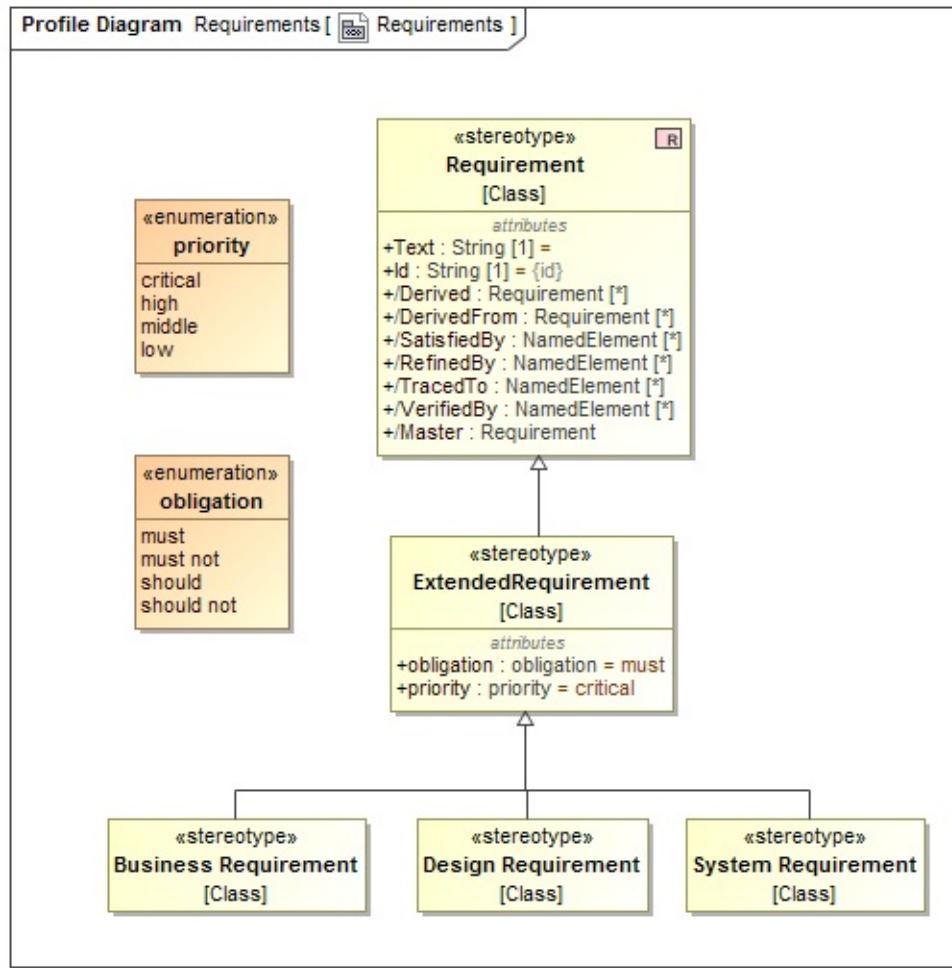


Abbildung 7: Erweiterung einer Anforderung

Abbildung 8 zeigt die Visualisierung des Stereotypen „Business Requirement“ in einem Requirement Diagramm der SysML.

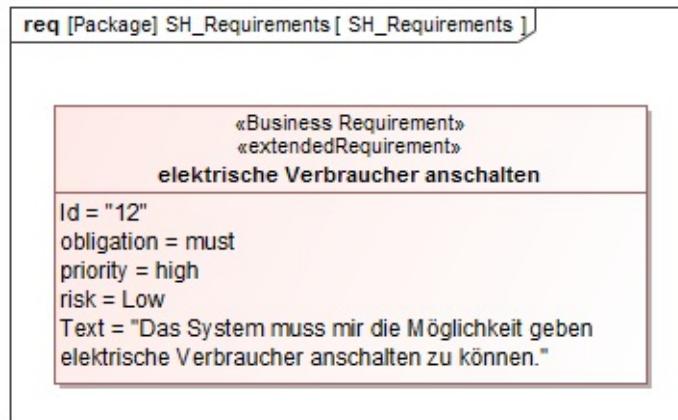


Abbildung 8: Anwendung des Stereotypens "Business Requirement"

Product Line Engineering

Unter dem Begriff Product Line Engineering (PLE) wird die Entwicklung eines Portfolios von verwandten Produkten durch die Verwendung gemeinsamer Entwicklungsartefakten und effizienten Produktionsmitteln verstanden. Anstatt verschiedene Produkte individuell zu entwickeln, steht beim Product Line Engineering die Entwicklung eines gesamten Produktportfolios in den frühen Phasen des Entwicklungsprozesses im Vordergrund. Durch diesen Ansatz können Synergien besser ausgeschöpft und die Wiederverwendung verbessert werden.

Beim Product Line Engineering werden die Produkte aufbauend auf einer Plattform entwickelt, die die Gemeinsamkeiten („Core Assets“) mehrerer Produkte zusammenfasst. Zu diesen Core Assets gehören z.B. Architekturen, Softwarekomponenten, Domain Models, Use-Cases oder Test Cases. Da diese Core-Assets bereitgestellt werden müssen, bevor ein konkretes Produkt entwickelt werden kann, ist die Zeit bis zur Markteinführung des ersten Produktes (time-to-market) und das Entwicklungsrisiko deutlich höher als bei der produktzentrischen Entwicklung. Jedoch können weitere Produkte später durch die Wiederverwendung deutlich schneller entwickelt werden. Die Zeit bis zur Markteinführung eines Produktes in Abhängigkeit der Anzahl der verschiedenen Produkte ist in Abbildung 9 dargestellt.

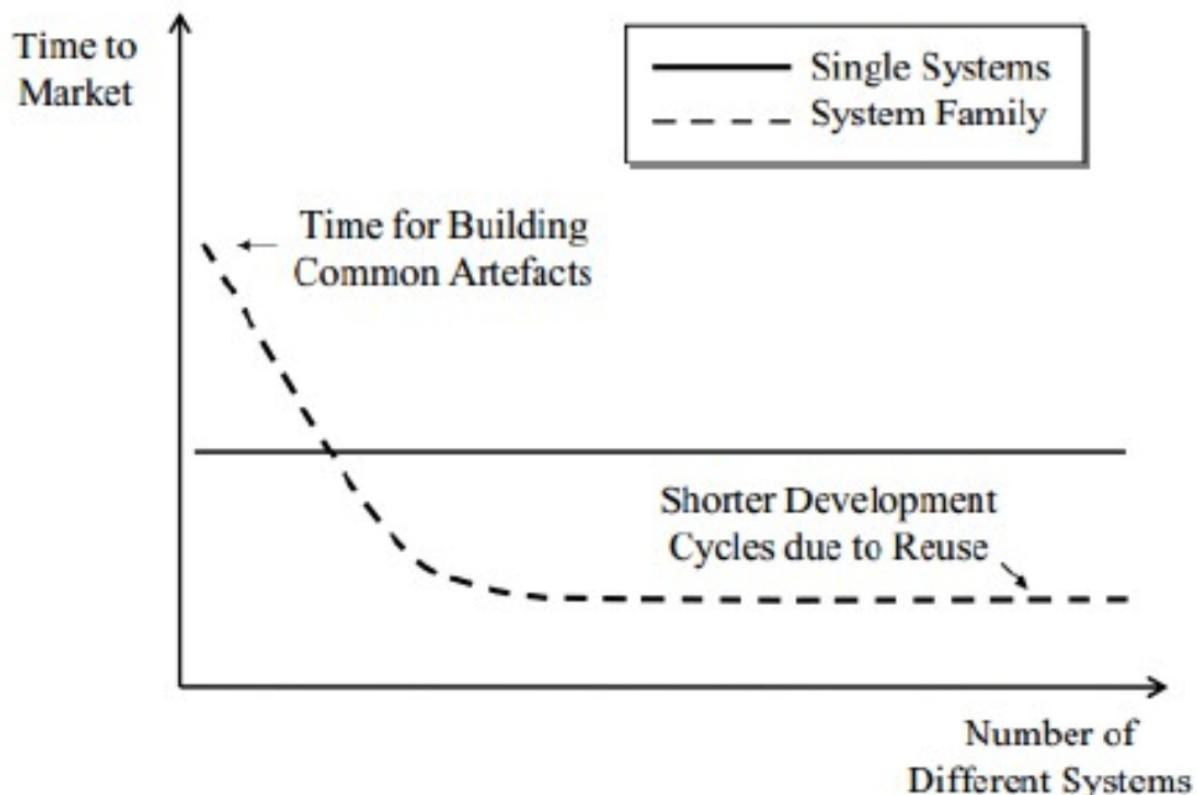


Abbildung 9: Time-to-Market in Abhängigkeit der Anzahl der Produkte KUBE15

Im Product Line Engineering wird zwischen zwei Bereichen unterschieden:

Im **Domain Engineering** werden die „Core-Assets“ herausgearbeitet und auf dieser Basis eine gemeinsame Systemarchitektur entwickelt.

Im **Application Engineering** werden auf Grundlage der Ergebnisse des Domain Engineerings die Produkte der Produktlinie entwickelt.

In Abbildung 10 ist die grundlegende Struktur des *Product Line Engineering* dargestellt. Die beiden Bereiche werden im Folgenden genauer erläutert.

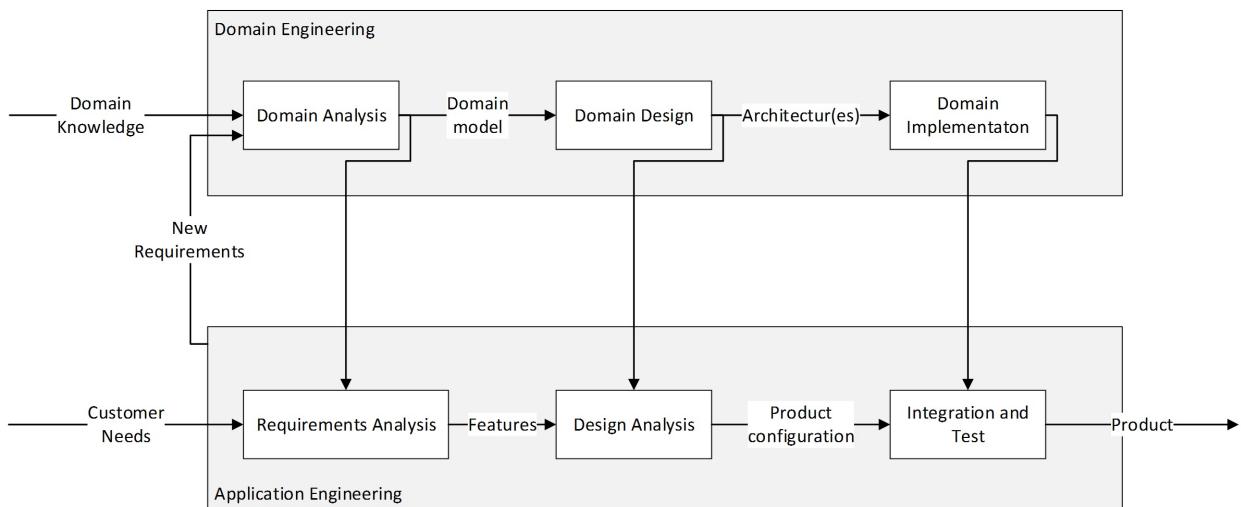


Abbildung 10: Struktur des Product-Line-Engineerings

Domain Engineering

Ziel des *Domain Engineering* ist es, bereits vorhandenes Wissen über eine Domäne zu erfassen und dieses Wissen bei der Entwicklung neuer Produkte in dieser Domäne zu nutzen [CZAR00](#).

„Domain Engineering is the activity of collecting, organizing, and storing past experience in building systems or parts of systems in a particular domain in the form of reusable assets (i.e., reusable work products), as well as providing an adequate means of reusing these assets (i.e., retrieval, qualification, dissemination, adaptation, assembly, and so on) when building new systems.“ [CZAR00](#)

Der *Domain Engineering* Ansatz ist hierbei in die drei Schritte *Domain Analysis*, *Domain Design* und *Domain Implementation* (siehe Abbildung 10) aufgeteilt.

In der **Domain Analysis** wird zu Beginn die „domain of focus“ definiert und anschließend Informationen über diese Domäne gesammelt. Aus diesen Informationen wird das Domain Model (siehe Absatz "Domain Model") erstellt. Als Informationsquelle dienen Domänen-Experten, Anforderungen von dedizierten Produkten der Produktlinie, sowie vorhandene (Mitbewerber-)Produkte. Das Domain Model dient als Eingangsartefakt für den nachfolgenden Schritt „Domain Design“. Die Ergebnisse gehen außerdem in die Anforderungsanalyse (Requirements Analysis) der Produkte der Produktlinie mit ein. Zusätzlich wird in der Domänen-Analyse über Feature-Modeling (siehe Absatz "Feature Modeling") die notwendigen Features der Produkte und deren Abhängigkeiten dargestellt. Die Feinheiten und Methoden des Feature-Modelings werden im späteren Verlauf des Kapitels erläutert.

Im **Domain Design** wird aus dem in der Domain-Analysis erstellten Domain-Model eine generische Systemarchitektur erstellt, die sich in allen Produkten der Produktlinie vereinigen lässt. (Vgl. [SCHR03](#))

In der **Domain Implementation** wird die generische Architektur implementiert. Diese kann in den diversen Produkten der Produktlinie Wiederverwendung finden. (Vgl. [SCHR03](#))

Application Engineering

Das *Application Engineering* beschreibt den Prozess der Entwicklung von Systemen, basierend auf den Ergebnissen des Domain Engineerings. Auch hierbei findet eine Auftrennung in drei Teilschritte statt (*Requirements Analysis*, *Design Analysis* und *Integration and Test* – siehe Abbildung 10).

Domain Model

Das Domain Model enthält alle Aspekte einer Domäne die zur Entwicklung einer Produktlinie innerhalb dieser Domäne von Relevanz sind. Das Domain Model ist hierbei eine abstrakte Beschreibung von Problemstellungen, die durch das System zu lösen sind. Ein einfaches Beispiel hierfür stellt eine Software zur Vertragsverwaltung dar. Das Domain Model enthält hierbei alle Informationen die zur Verwaltung der Verträge relevant sind - zum Beispiel die Beziehungen zwischen Verträgen und Kunden.

Vorgehensmodelle

Im Produktlinienansatz wird zwischen drei unterschiedlichen Vorgehensmodellen unterschieden (Vgl. [APEL10](#)):

Im **proaktiven Vorgehensmodell** wird eine Produktlinie komplett neu entwickelt. Hierbei wird zu Beginn der komplette Domain-Engineering-Prozess durchlaufen. Dies ist mit hohem initialen Aufwand und den damit einhergehenden Kosten verbunden. Der proaktive Ansatz ist sinnvoll, wenn die Anforderungen gut definiert und konstant sind. (Vgl. [APEL10](#))

Das **reaktive Vorgehensmodell** lässt sich mit agilen Vorgehensmodellen in der Software-Entwicklung vergleichen. Zu Beginn erfolgt eine Implementierung einer kleinen Basis. Nachfolgend werden in kleinen Schritten die Domänen-Analyse mit anschließender weiterer Implementierung durchgeführt. Hierdurch erhält man ein zyklisches Arbeiten. Die Vorteile dieses Vorgehensmodell sind wie beim agilen Ansatz die geringen initialen Kosten, wie auch das Ausliefern schneller erster Ergebnisse. Es können allerdings auch Probleme entstehen, die zu einer teuren Umstrukturierung der Arbeitsweise im späteren Verlauf des Projektes führen. Das reaktive Vorgehensmodell ist vor allem geeignet, wenn die Anforderungen zu Beginn noch sehr unklar sind.

Beim **extraktiven Vorgehensmodell** dient eine bereits entwickelte Produktlinie als Grundlage. Aus dieser werden Informationen extrahiert, die in die neue zu entwickelnde Produktlinie einfließen. Da eine existente und funktionsfähige Produktlinie als Basis dient, halten sich die Risiken und Kosten in Grenzen. Allerdings steigt auch der Anspruch im Domain Engineering, da Informationen aus einem System extrahiert werden, das nicht im Produktlinienansatz entwickelt wurde.

Das extractive Vorgehensmodell ist geeignet, wenn man einen schnellen Wechsel vom traditionellen zum Produktlinien-Ansatz realisieren möchte.

Feature Modeling

Im Folgenden wird gezeigt wie mit Feature Modeling die Features bzw. Varianten eines Produktes dargestellt werden können. Die hierbei verwendete Sprache ist die Feature Modeling Notation, die nicht in der UML bzw. SysML enthalten ist.

Ein *Feature* wird immer aus Anwendersicht betrachtet:

„A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems“ [FODA90](#)

Zu Beginn erfolgt eine kurze Einleitung in die verwendete Notation, die abschließend in einem Beispiel angewendet wird.

Mit der **Konjunktion** (UND-Verknüpfung) wird beschrieben, dass alle verbundenen Elemente (Features) im Produkt umgesetzt werden müssen. In Abbildung 11 müssten also alle Elemente (A, B und C) vom Produkt implementiert werden:

$$A \wedge B \wedge C$$

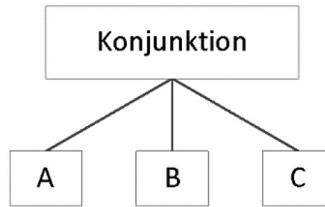


Abbildung 11: Konjunktion

Mit der **Disjunktion** (ODER-Verknüpfung) wird beschrieben, dass mindestens eines der verbundenen Elemente umgesetzt werden muss. Es besteht keine Abhängigkeit zwischen den Elementen. Es können also beliebige Kombinationen der Features umgesetzt werden. Die Disjunktion wird durch einen ausgefüllten Halbkreis am ausgehenden Feature-Strang markiert.

Im Beispiel dürften also beliebige Kombinationen aus den Features A, B und C umgesetzt werden. Jedoch mindestens eines von Ihnen:

$$A \vee B \vee C$$

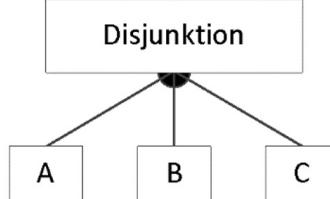


Abbildung 12: Disjunktion

Die **Kontravalenz** (Exklusiv-ODER-Verknüpfung) beschreibt, dass genau eines der verbundenen Elemente umgesetzt werden muss. Es darf kein weiteres Element umgesetzt werden. Die Kontravalenz wird durch einen Halbkreis am ausgehenden Feature-Strang markiert. Im Folgenden müsste also entweder Feature A, B oder C umgesetzt werden:

$$A \oplus B \oplus C$$

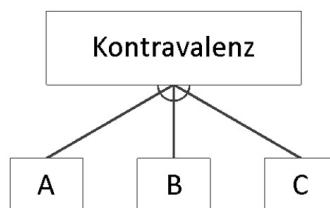


Abbildung 13: Kontravalenz

Mit **Implikationen** können Abhängigkeiten zwischen Features dargestellt werden. Wird ein Feature umgesetzt, kann dies bedeuten, dass auch ein anderes Feature umgesetzt werden muss. In Abbildung 14 können Feature A oder C oder beide umgesetzt werden. Wird C umgesetzt so muss auch B umgesetzt werden. Falls A umgesetzt wird darf B ebenfalls umgesetzt werden, muss es allerdings nicht. Es existiert keine Abhängigkeit zwischen A und B.

$$A \vee (C \wedge B)$$

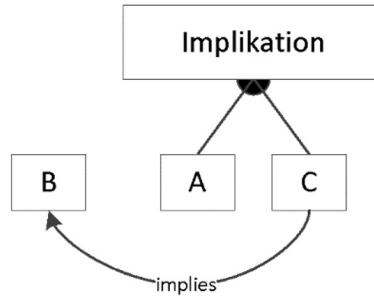


Abbildung 14: Implikation

Obligationen und **Optionen** werden eingesetzt um darzustellen, dass Features umgesetzt werden müssen (Obligation) bzw. umgesetzt werden können (Option). Die Obligation wird dabei durch einen ausgefüllten Kreis am entsprechenden Feature dargestellt, die Option durch einen nicht ausgefüllten. Im Beispiel müssen sowohl Feature A als auch C umgesetzt werden. Feature B kann optional umgesetzt werden:

$$(A \wedge \bar{B} \wedge C) \vee (A \wedge B \wedge C) = A \wedge C$$

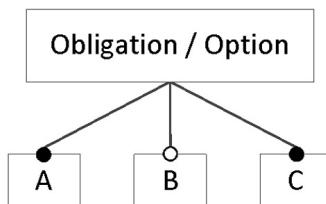


Abbildung 15: Obligation / Option

Im Folgenden wird eine exemplarisches Feature-Model eines Smartphones gezeigt. Dieses Model zeigt die Features, die ein Kunde bei der Auswahl seines Gerätes direkt konfigurieren kann.

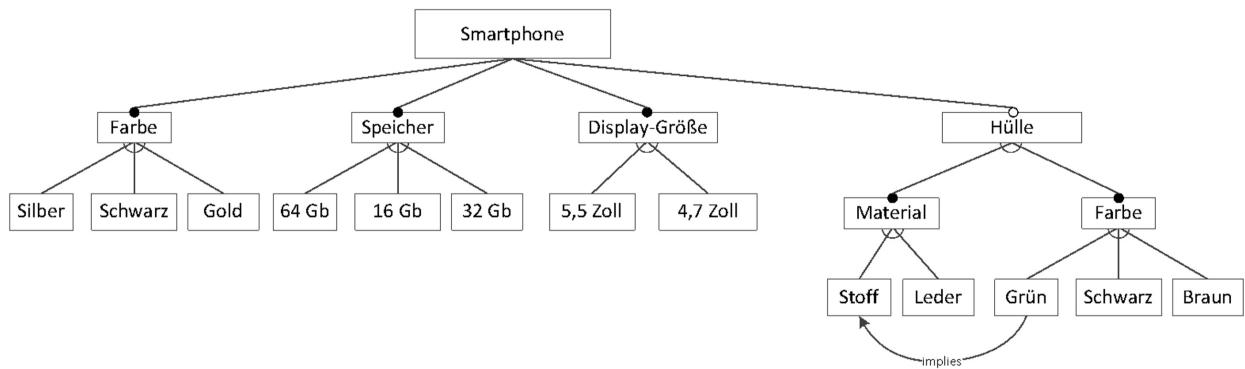


Abbildung 16: Smartphone-Feature-Modell

Durch die zuvor erläuterten Notationselemente ist zu erkennen, dass der Kunde in jedem Fall eine Auswahl der Features *Farbe*, *Speicher* und *Display-Größe* treffen muss. Des Weiteren hat er die Möglichkeit eine *Hülle* zum Gerät hinzuzufügen. Hierbei hat er die Auswahl zwischen zwei *Materialien* (*Leder* und *Stoff*) und verschiedenen *Farben* (*Grün*, *Schwarz* und *Braun*). Entscheidet sich der Kunde für die Farbe *Grün* ist er gezwungen ebenfalls das Material *Stoff* zu nehmen, da diese Farbe für das Material *Leder* nicht zur Verfügung steht.

Aus diesem relativ übersichtlichen Feature-Modell lassen sich bereits 108 unterschiedliche Varianten bilden (Berechnung siehe [Anhang](#)). Es ist also zu erkennen, dass auch eine geringe Variabilität von Features in einem Produkt zu einem schnellen Anstieg der Anzahl der Varianten führen kann.

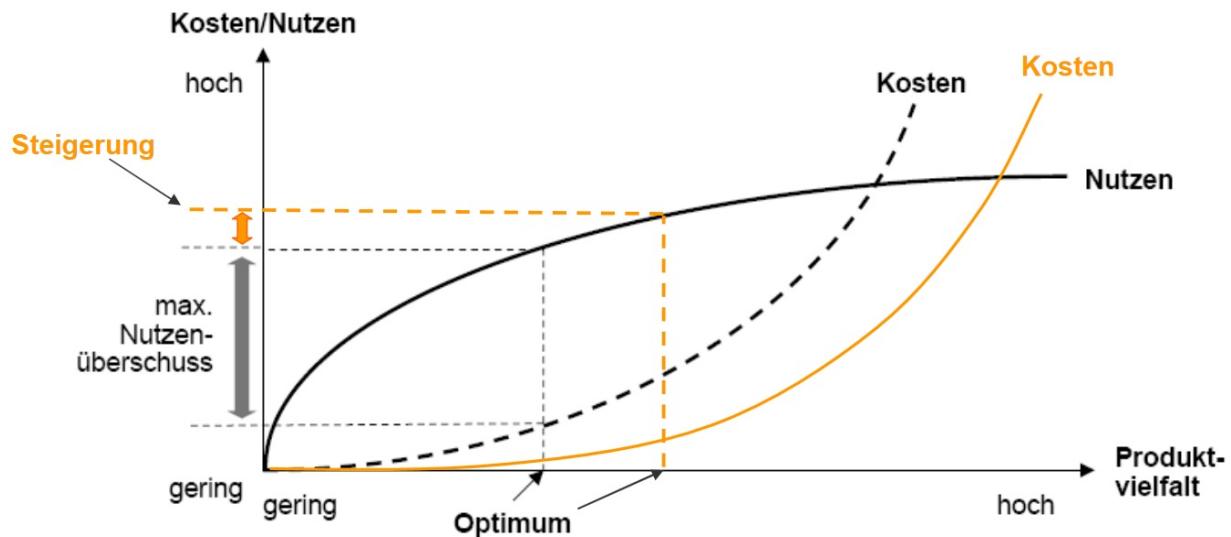


Abbildung 17: Produktvielfalt im Verhältnis zum Kundennutzen [ROCK09](#)

Die Varianten und die Anzahl der Features sind in einem durchschnittlichen Software-System um ein Vielfaches höher, als bei dem hier dargestellten Produkt. Umso wichtiger ist es die Variabilität so gering wie möglich zu gestalten, um auch die Komplexität auf einem

kontrollierbaren Level zu halten. Die Variantenbildung ist oft eine Gratwanderung zwischen Kundennutzen und Komplexität. Dieser Sachverhalt wird in Abbildung 17 gezeigt.

Model Transformation

Eine zentrale Technik bei der modellbasierten Entwicklung ist die Modelltransformation. Die Modelltransformation wird benötigt, um Informationen eines Datenmodells in ein anderes Datenmodell, welches eine andere Struktur besitzt, zu überführen. [OSTF17](#)

Die Bandbreite der Anwendungsgebiete reicht von der Dokumentengenerierung über die Variantengenerierung bis hin zur Erstellung neuer Modelle und der Codegenerierung. Dabei ist die Vorgehensweise der Transformation immer gleich. Es werden Informationen aus dem Quellmodell abgefragt und diese nach definierten Regeln in eine andere Form gebracht. Diese Regeln werden auch als Transformationsregeln bezeichnet und werden auf der Ebene der Metamodelle definiert.

Bei der Modelltransformation wird zwischen Modell-zu-Modell-Transformation (M2M-Transformation) und Model-zu-Text-Transformation (M2T-Transformation) unterschieden. [WIKI17b](#)

Modell-zu-Modell-Transformation

Innerhalb der M2M-Transformation kann erneut in den zwei Varianten Inplace-Transformation und Outplace-Transformation unterschieden werden. Bei der Inplace-Transformation wird durch die Transformation kein neues Modell erzeugt, sondern das bestehende Modell modifiziert. Die Outplace-Transformation generiert dagegen ein neues Modell und modifiziert das Quellmodell nicht. [WIKI17b](#)

Im Folgenden sollen verschiedene Transformationssysteme im Überblick dargestellt werden.

Query View Transformation (QVT)

Die Query View Transformation (QVT) ist eine formale Sprache für M2M-Transformation und wurde von der OMG standardisiert. Sie ist Teil der Meta Object Facility (MOF). Unter einer Query werden Anfragen auf ein MOF-Modell verstanden, um bestimmte Modellelemente zu finden, die transformiert werden sollen. Unter einer View versteht die OMG ein Modell, welches aus einem Quellmodell abgeleitet wurde. Die Transformation ist die Überführung von ein oder mehrere Quellelementen in ein oder mehrere Zielelemente. Für die Transformation stehen innerhalb der QVT zwei Sprachen zur Verfügung:

- die QVT Operational Mappings Language, welche eine imperitative Sprache ist.
- die QVT Relations Language, die eine deklarative Sprache ist und neben einer textuellen auch eine grafische Syntax definiert. [ITWI17a](#)

Atlas Transformation Language (ATL)

Die Atlas Transformation Language ist eine Sprache zur Modelltransformation und wurde auf Grund einer Ausschreibung der OMG für eine allgemeine Programmiersprache zur Modell- und Codetransformation entwickelt. Die ATL ist eine hybride Sprache. Dies bedeutet, dass sie Konzepte imperativer und deklarativer Programmierung vereint. [ITWI17b](#)

MOF Model to Text Transformation Language (MOFM2T)

Die MOF Model to Text Transformation Language ist ein weiterer Standard der OMG zur metamodeltbasierten Modell-zu-Text-Transformation. Sie kann genutzt werden, um z.B. Code auf einem UML oder SysML-Modell zu generieren. [WIKI17e](#)

[INC17a] INCOSE, What is Systems Engineering?, aufgerufen am 28.05.2017,

URL: <http://www.incose.org/AboutSE/WhatIsSE>

[INC17b] INCOSE, About Systems Engineering, aufgerufen am 28.05.2017,

URL: <http://www.incose.org/AboutSE>

[INC15] INCOSE, Systems Engineering Handbook,

A Guide for System Life Cycle Processes and Activities, Fourth Edition, 2015

[THM17] Technische Hochschule Mittelhessen (THM), Systemmodell, aufgerufen am 28.05.2017,

URL: <https://wiki.thm.de/Systemmodell>

[WIKI17a] Wikipedia, Repository, aufgerufen am 29.05.2017,

URL: <https://de.wikipedia.org/wiki/Repository>

[OMG17] ObjectManagement Group, What is SysML, aufgerufen am 18.06.2017,

URL: <http://www.omg.sysml.org/what-is-sysml.htm>

[WIKI17c] Wikipedia, Metamodell, aufgerufen am 18.06.2017,

URL: <https://de.wikipedia.org/wiki/Metamodell>

[WIKI17d] Wikipedia, Meta Object Facility, aufgerufen am 18.06.2017,

URL: https://de.wikipedia.org/wiki/Meta_Object_Facility

[ITWI17a] IT Wissen, QVT (query view transformation), aufgerufen am 18.06.2017,

URL: <http://www.itwissen.info/QVT-query-view-transformation.html>

[ITWI17b] ITWissen, ATL (ATLAS transformation language), aufgerufen am 18.06.2017,

URL: <http://www.itwissen.info/ATL-ATLAS-transformation-language.html>

[WIKI17e] Wikipedia, MOF Model to Text Transformation Language, aufgerufen am 18.06.2017,

URL: https://de.wikipedia.org/wiki/MOF_Model_to_Text_Transformation_Language

[WIKI17f] Wikipedia, Unified Modeling Language, aufgerufen am 18.06.2017,

URL: https://de.wikipedia.org/wiki/Unified_Modeling_Language

[WIKI17b] Wikipedia, Modelltransformation, aufgerufen am 29.05.2017,

URL: <https://de.wikipedia.org/wiki/Modelltransformation>

[OSTF17] Ostfalia,Hochschule für angewandte Wissenschaften, Modelltransformation,
aufgerufen am 02.06.2017,

URL: www.ostfalia.de/cms/de/ivs/ease/Merapi/Modelltransformation.html

[APEL10] Apel, Sven; Christian, Kästner; Saake, Gunter; Softwareproduktlinien,
Entwicklungsprozess und

Variabilitätsmodellierung, Universität Marburg, 2010

[FODA90] Kyo C. Kang; Sholom G. Cohen; James A.Hess; William E. Novak; A. Spencer
Peterson;

Feature-Oriented Domain Analysis(FODA) Feasibility Study; Software Engineering Institute,
Carnegie Mellon University, Pittsburgh 1990

[CZAR00] K. Czarnecki and U. W.Eisenecker; Generative Programming|Methods, Tools, and
Applications; Addison-Wesley, 2000. ISBN 0-201-30977-7

[SCHR03] Prof. Dr.-Ing. Wolfgang Schröder-Preikschat, Dr.-Ing. Olaf Spinczyk;
Vorlesung Betriebssystemtechnik (OSE) – Domain Engineering, 2003

[ROCK09] Dr. Georg Rock; Variantenmanagement –Forschung und industrieller Einsatz; TU
Darmstadt, 2009

[KUBE15] Kuberczyk, Christoph; Product Line Engineering (PLE) - Produktlinienentwicklung;
Universität Hamburg, 2015

Anhang

Berechnung der Anzahl der Varianten:

$$V = n_{Farbe} * n_{Speicher} * n_{DisplayGröße} * (1 + (n_{Material} * (n_{HüllenFarbe} - 1)) + 1)$$

Mit:

V := Anzahl aller möglichen Varianten

$n_{Farbe} = 3$:= Anzahl der möglichen Farbvarianten des Geräts

$n_{Speicher} = 3$:= Anzahl der möglichen Speichervarianten des Geräts

$n_{DisplayGröße} = 2$:= Anzahl der möglichen Displayvarianten des Geräts

$n_{HüllenFarbe} = 3$:= Anzahl der möglichen Farbvarianten der Hülle

$n_{Material} = 2$:= Anzahl der möglichen Materialvarianten der Hülle

Continuous Software Engineering

Ursprünglich verstand man unter dem Begriff der kontinuierlichen Softwareentwicklung die Pflege und Weiterentwicklung von bestehenden Softwarekomponenten. Im Zuge der zunehmenden Komplexität von Software zählen heute zur kontinuierlichen Softwareentwicklung Methodiken und Prozesse, die bereits während der Entwicklung zum Einsatz kommen. Dazu gehören agile Vorgehensmodelle wie Scrum und Kanban, mit denen umfangreichen Produktanforderungen iterativ erarbeitet werden. Zur Erreichung von Qualitätszielen sind unterschiedliche Workflow-Ansätze wie DevOps, Continuous Integration und Test Driven Development Bestandteile der Vorgehensmodelle. Zur technischen Umsetzung solcher Workflows bedarf es zudem Hardware- und Softwarekomponenten, die entwicklungsbegleitend und automatisiert bei der Anforderungsumsetzung und Qualitätssicherung unterstützen. [10](#)

Agile Vorgehensmodelle

Agile Vorgehensmodelle sind eine Methode des Projekt Managements. Sie zeichnen sich üblicherweise durch iteratives Vorgehen und kurze Feedback-Zyklen aus. Sie sollen nach häufiger Auffassung schwergewichtige Vorgehensmodelle ablösen, die mit viel Management-Aufwand verbunden sind, ablösen.

Robert Martin beschreibt hingegen, dass in den Anfangszeiten der Software-Entwicklung in den 60er Jahren bereits agil gearbeitet wurde. Er begründet das damit, dass damals hauptsächlich disziplinierte, professionelle Mathematiker und Fachanwender als Softwareentwickler tätig waren. Erst das auftauchen großer Mengen junger, undisziplinierter Entwickler machte es nötig, ihnen feste Strukturen vorzugeben. (vgl. [6], ab ca. 48m)

Bekannte, aktuelle Vertreter agiler Vorgehensmodelle sind Scrum, Kanban und Extreme Programming (XP). Die ersten beiden sollen nachfolgend kurz erläutert werden.

Scrum

Das zur Zeit bekannteste agile Modell ist Scrum, es soll hier nur in seinen Grundzügen erläutert werden. Es stammt zwar aus der Softwareentwicklung, ist aber prinzipiell unabhängig davon.

Einer der Hauptaspekte von Scrum ist die Einteilung der Projektlaufzeit in sogenannte Sprints. Diese dauern üblicherweise 2-4 Wochen und am Ende jedes Sprints sollte eine lauffähige, inkrementell verbesserte Software bereit stehen. Während eines Sprints werden einzelne Aufgaben in Form von Backlog-Items abgearbeitet. Die abzuarbeitenden Items werden jeweils vor Beginn eines Sprints festgelegt und dem Product-Backlog entnommen. Die Backlog-Items, die während eines Sprints abgearbeitet werden sollen, werden im Sprint-Backlog abgelegt.

Der Product-Backlog beinhaltet alle noch zu erledigenden Backlog-Items und wird vom Product-Owner gefüllt. Das geschieht initial beim Projektbeginn, sowie während der Projektlaufzeit (z.B. Änderungswünsche nach einem Sprint).

Neben dem Product Owner gibt es außerdem noch das Entwicklerteam als beteiligte, welche die eigentliche Entwicklungsarbeit erledigen. Außerdem gibt es den Scrum Master, der die Durchführung überwacht. Er stellt eine "dienende Führungskraft" dar, gibt also keine Arbeitsanweisungen, sondern steht beratend zur Seite.

Kanban

Kanban ist ein weiteres agiles Vorgehensmodell, sein zentraler Punkt ist das sogenannte Kanban-Board. Der Begriff "Kanban" stammt aus dem japanischen und bedeutet "Signalkarte", dort wurde es als Produktionsprozess ursprünglich von Toyota entwickelt. In der IT wurde es zuerst 2007 von David J. Anderson vorgestellt.

Pull-Prinzip

Kanban arbeitet nach dem sogenannten Pull-Prinzip. Das bedeutet, dass keine Arbeit auf Vorrat erledigt wird, sondern immer erst dann ausgeführt wird, wenn sie wirklich benötigt wird. Das führt in der Anwendung in Produktionsumgebungen zu deutlich geringeren Lagerkapazitäten und einer Reduktion von Abfall, falls auf Vorrat produziertes am Ende nicht mehr gebraucht werden kann. In der Software-Entwicklung ist Lagergröße zwar kein entscheidender Faktor, aber die Reduktion von Abfall, also Code der am Ende nicht gebraucht wird, ist mindestens genau so wichtig, wenn nicht wichtiger, da die Arbeitszeit der Entwickler die hauptsächlich begrenzende Ressource ist.

Kanban-Board

Im Kanban-Board werden alle einzelnen Aufgaben in Form von Zetteln aufgeklebt. Dabei gibt es eine Unterteilung in Spalten, die jeweils mit dem Zustand der darin enthaltenen Karten beschriftet sind (z.B. "To Do", "In Arbeit", "Testing", "Erlledigt").



Abbildung: Beispielhaftes Kanbanboard, Quelle: [Wikipedia](#), Jeff.lasovski

Das Board darf natürlich nicht nach belieben mit Aufgaben versehen werden. Es unterliegt einigen Regeln und Gesetzmäßigkeiten, diese werden in den sechs Kernpraktiken genauer beschrieben.

Kernpraktiken

Die sechs Kernpraktiken von Kanban lauten im einzelnen[7]:

1. Visualisiere den Fluss der Arbeit

Auf dem Kanban Board finden sich alle Aufgaben, die noch ausstehen, gerade erledigt werden, oder schon abgeschlossen sind. Das bringt eine gute Übersicht über den aktuellen Status einzelner Arbeitseinheiten sowie den Fortschritt des Projekts.

2. Begrenze die Menge angefangener Arbeit

Es darf jeweils nur eine begrenzte Menge an Karten in den Spalten vorhanden sein. Es kann allerdings sinnvoll sein, nicht für jede Spalte ein Limit zu setzen. So kann zum Beispiel eine Spalte "Fertig" alle bereits erledigten Karten enthalten. Diese zu beschränken würde schlussendlich dazu führen, dass keine Arbeit mehr verrichtet werden darf. Dadurch wird effektiv die angefangene Arbeit auf die Anzahl der Karten limitiert. Das Ziel ist es dabei, dass einzelne Aufgaben schneller erledigt werden, wenn nicht zwischendurch andere angefangen werden. Dabei wird die benötigte Zeit für die einzelnen Aufgaben zwar nicht signifigant gesenkt (höchstens durch weniger Wechsel zwischen Aufgaben), aber der Fortschritt ist schneller sichtbar. Außerdem können eventuell abhängige Aufgaben so eher von anderen Entwicklern begonnen werden.

3. Miss und steuere den Fluss

Typische Werte, die in einem Kanbanprozess gemessen werden, sind die Wartezeit, Zykluszeit und der Durchsatz. Diese erlauben es, Rückschlüsse auf die Organisation der Arbeit zu ziehen und sie zu verbessern.

4. Mache die Regeln für den Prozess explizit

Um unnötige Unsicherheiten und Ablaufprobleme zu vermeiden, sollen alle Beteiligten die genauen Regeln des Prozesses kennen. Dazu gehören unter anderem die exakte Bedeutung der Kanban-Board-Spalten, was genau fertig heißt (Definition of Done), und der genaue Ablauf, wer wann welche Karten auf dem Kanban-Board verschiebt.

5. Implementiere Feedback-Mechanismen

Um den Prozess kontinuierlich zu verbessern, ist es wichtig, Schwächen aufzudecken und Verbesserungsmöglichkeiten für diese zu finden. Damit diese kontinuierliche Verbesserung funktioniert, muss sie auf allen Ebenen der Organisation stattfinden. Dazu gehört, dass auch Entwickler konkrete Verbesserungsvorschläge einbringen.

Darüber hinaus kann aber auch Feedback von Außenstehenden wertvoll sein, da es neue Blickwinkel ermöglicht. Nützliche Mittel, um regelmäßiges Feedback zu ermöglichen, sind Daily-Standups und insbesondere (z.B. monatliche) Retrospektiven.

6. Führe Verbesserungen auf Basis von Modellen durch

Schon lange gibt es wissenschaftliche Modelle, die sich mit Engpässen und Problemen in Betriebsabläufen beschäftigen. Es ist sinnvoll, diese für die Verbesserung des Prozesses einzusetzen. Die Auswahl der Modelle ist nicht weiter eingeschränkt, damit die Modelle verwendet werden können, die für den konkreten Einzelfall am geeignesten sind.

Kanban und andere Vorgehensmodelle

Da Kanban im Gegensatz zu anderen Vorgehensmodellen einige Dinge nicht festlegt, ist es prinzipiell Möglich, Kanban gleichzeitig mit anderen Modellen einzusetzen. So ist zum Beispiel keine zeitliche Einteilung in Sprints gegeben, wie es in Scrum der Fall ist. Außerdem gibt es keine feste Rollenbeschreibung der Beteiligten. Kanban und Scrum können so zum Beispiel in einer Kombination durchgeführt werden. So können die Vorteile des Kanban-Boards und dessen Regeln mit den Rollen- und Zeiteinteilung von Scrum vereint werden.

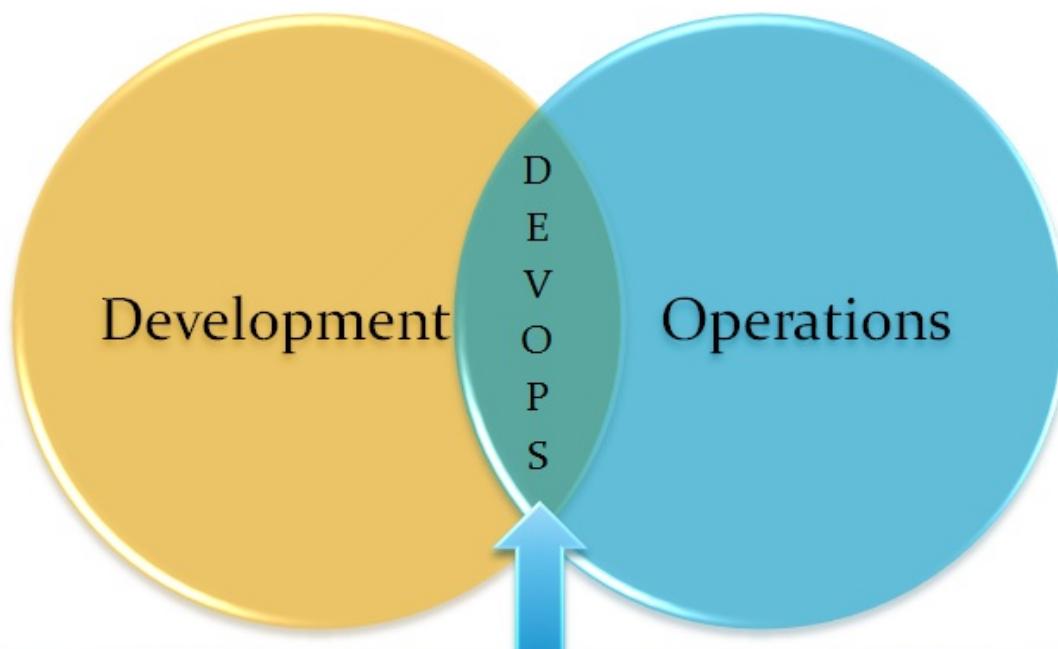
Es ist auch möglich, Kanban mit nicht agilen Vorgehensmodellen zusammen zu benutzen. Fall es in irgendeiner Art Kollisionen gibt, ist auch ein "Kanban-Light" vorstellbar, dass je nach Anwendungsfall eine oder mehrere Kernpraktiken ändert oder auslässt. In jedem Fall kann dabei der Vorteil des Kanban-Boards und dessen dokumentierender bzw. visualisierender Funktion genutzt werden.

Unterstützende Workflows

DevOps

Wichtig für das Verständnis ist, dass DevOps kein Framework, Tool oder Technologie ist. Bei DevOps geht es um die Kultur eines Unternehmens und um das Vorgehen, wie die Menschen in einem Unternehmen arbeiten. Unterstützende Faktoren sind definierte Prozesse und angepasste Tools um durch automatisierte Abläufe eine erhöhte Effektivität zu erreichen.

Durch die Anpassungen und Veränderungen in den Bereichen Kultur, Prozesse und Technologie soll die Kommunikation, sowie die Zusammenarbeit der Entwicklung (development) und "IT operations" verbessert werden [20](#).



People + Transformation of existing Processes + Technology
Patterns + Reusable Components + Automation Tools
Effective Communication and Collaboration
Standardized Practices across Teams

Dies hat Auswirkungen auf die Effektivität des "application life cycles". Durch diese Auswirkungen sind die Vorteile, welche durch den Einsatz von DevOps erzielt werden können, sehr vielseitig.



Die Kultur DevOps kann als innovatives Bündel gesehen werden um "Dev" und "Ops" Teams effektiv zu integrieren. Dies beinhaltet Themen wie continuous build integration, continuous testing, continuous delivery, continuous improvement, etc., mit dem Ziel die Auslieferung von Software zu beschleunigen.

Die Schwierigkeit liegt darin, dass DevOps häufig nicht verstanden wird. Hier geht es nicht um einzelne Bereiche, wie bereits aufgezählt bspw. continuous integration. Es geht um die Kultur! Und eine Kultur zu wandeln, dauert sehr lange. Ein Beispiel soll das Missverständnis rund um DevOps verdeutlichen. Hier geht es um fünf blinde Männer und einen Elefanten. Jeder der Männer berührt einen spezifischen Teil des Elefanten und jeder nimmt an das dies der Elephant sei [20](#).

Selbstverständlich bezieht sich der Kulturwandel nicht nur auf das "development" und "operations" Team. An diesem Wandeln müssen sich alle Teams (testing team, cloud team, etc.) anschließen.

Für ein tieferes Verständnis werden wir im Folgenden auf zwei Fragestellungen eingehen: 1) Warum DevOps? 2) Wie kann eine DevOps Kultur entstehen?

Warum DevOps?

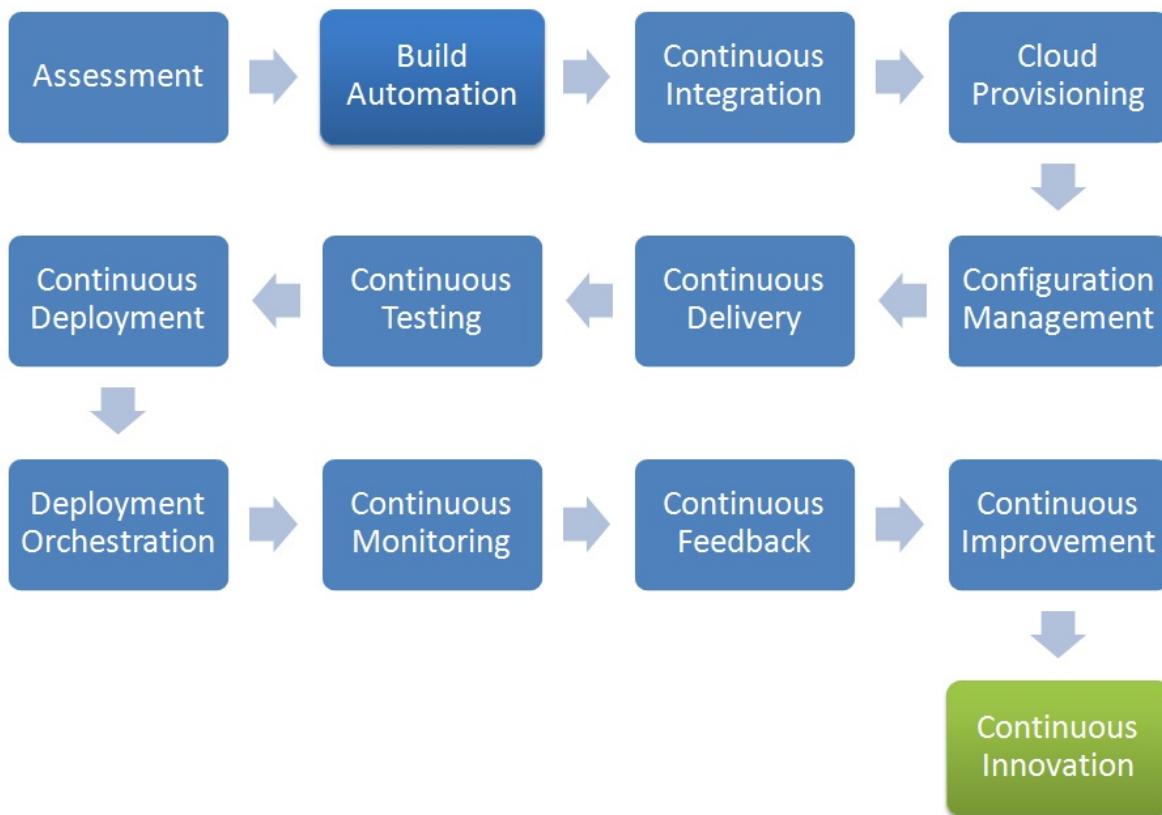
Veränderung ist eine essentieller Bestandteil des Lebens und auch von Organisationen. Schaut man lediglich auf bestehende Methodiken, Kulturen und Praktiken wird man zukünftige "best practices" nie erreichen. Vor allem in der sich schnell ändernden IT-Welt hat die Veränderung eine enorme Bedeutung.

Progress is impossible without change, and those who cannot change their minds cannot change anything. [20](#)

Jede Änderung muss abgewogen werden nach dem Nutzen und den durch die Änderung entstehenden Schwierigkeiten.

Wie kann eine DevOps Kultur entstehen?

Häufig führen Probleme und Ineffizienz zu Veränderungen. Hierdurch hat sich aus einem Wasserfallmodell die Agile Methodik entwickelt. Eine Kultur zu verändern geht nicht über Nacht und benötigt sehr viel Zeit. Hieraus resultiert, dass DevOps als Kultur schrittweise eingeführt und akzeptiert werden muss. Jeder Schritt ist unabhängig von einander zu implementieren und sollte keine Abhängigkeiten zu anderen Schritten haben [20](#).



Bspw. kann "Continuous Testing" ohne jede andere DevOps Praktik implementiert werden. Hierdurch entsteht bereits ein Mehrwert, welcher dazu beiträgt Akzeptanz zu schaffen und die Kultur der Menschen zu verändern [20](#).

Continuous Integration

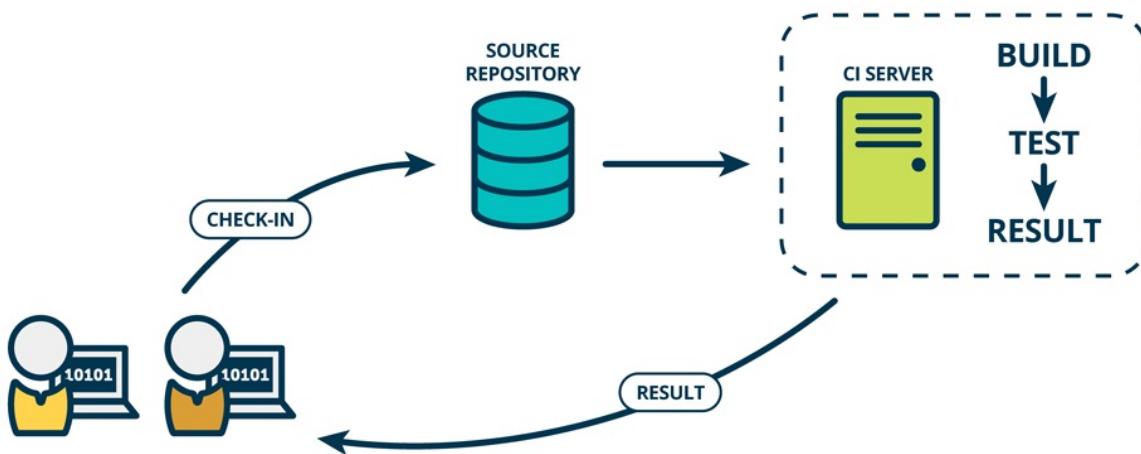
Traditionell wird bei der Softwareentwicklung die Integration am Ende eines Projektes stattfinden. Je nach Projektgröße und Komplexität liegt die Dauer der Integrationsphase im Bereich Wochen oder Monate.

Continuous Integration (CI) bedeutet, dass zwei Entwickler unabhängig von einander Software schreiben können, jedoch für das selbe Produkt. Dies geschieht durch ständige und regelmäßige Integration in ein "source repository".

In dem "source repository" werden die einzelnen Incremente mehrere Entwickler zusammengeführt und können durch einen CI Server integriert werden.

Grundvoraussetzung hierfür, ist das jeder Entwickler neben seinem Produktiv-Code auch Tests entwickelt. Diese Tests werden durch den CI Server ausgeführt um sicher zu stellen, dass die einzelnen Incremente funktionsfähig bleiben. Das Ergebnis des CI Servers wird an die jeweiligen Entwickler zurück gegeben. Hierdurch können Fehler sofort behoben werden.

Das folgende Bild veranschaulicht den geschilderten Prozess.



Der integrierte Code, ist durch Continuous Integration nicht bereit in Produktion zu gehen, da zwar die Komponenten mit einander funktionieren, allerdings das Produkt (die Software) noch nicht in einer "production-like environment" getestet und verifiziert wurde.

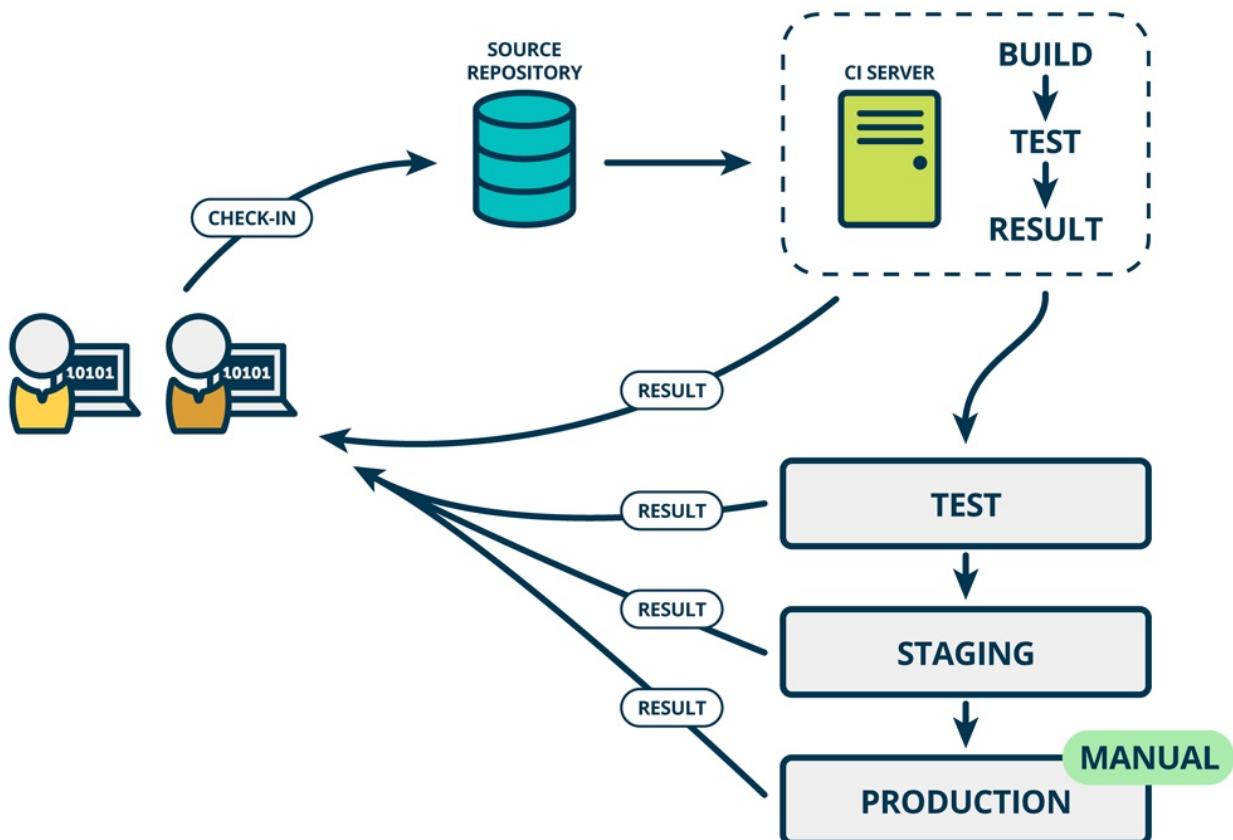
Der Vorteil von CI ist, dass die Integration zum täglichen Geschäft wird. Eine "Big-Bang" Integration am Ende einer Entwicklung wird vermieden.

CI ist zwingend notwendig um Continuous Delivery durchführen zu können [4](#).

Continuous Delivery

Unter Continuous Delivery versteht man, neben allen beschriebenen Schritten von Continuous Integration, das der integrierte Quelltext automatisiert auf verschiedenen Umgebungen getestet wird. Typischerweise sind diese Umgebungen sehr ähnlich zu einer potentiellen Produktiv-Umgebung.

In diesem Zusammenhang wird das deployen und testen auf den verschiedenen Umgebungen als "deployment pipeline" bezeichnet. Je nach Projekt, Team oder Organisation können unterschiedlich viele Umgebungen existieren. Das folgende Bild veranschaulicht diesen Prozess:

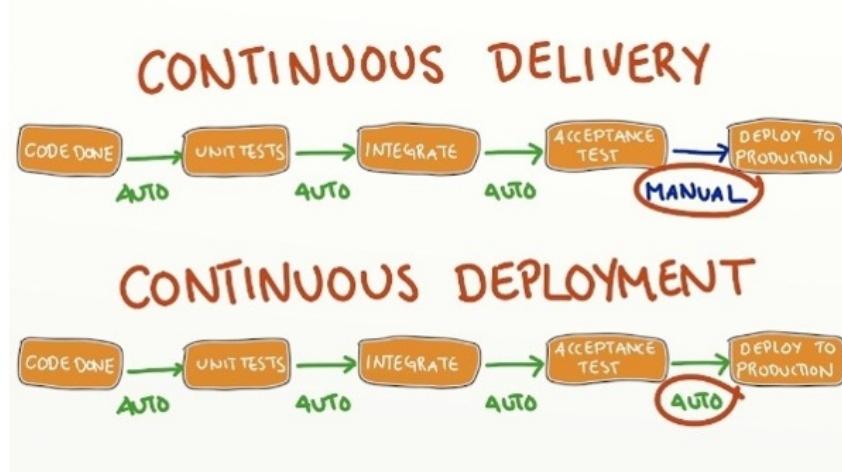


In diesem Beispiel gibt es drei Umgebungen (Development, Test, Staging), welche durchlaufen werden, bevor es in die Produktiv-Umgebung geht.

In jeder Umgebung wird der Quelltext unterschiedlich getestet. Es ergibt sich der Vorteil, dass mit jeder erfolgreich durchlaufenen Umgebung die Wahrscheinlichkeit wächst, dass der Quelltext auch in der Produktiv-Umgebung lauffähig sein wird [4](#).

Continuous Deployment

Continuous Deployment geht noch einen Schritt weiter und automatisiert ebenfalls die Übergabe in die Produktion.



5

Jeder einzelne Commit eines Entwickler kann potentiell automatisiert in der Produktion ankommen. Voraussetzung hierfür ist der erfolgreiche Durchlauf aller bisher aufgeführten Schritte aus den Kapiteln Continuous Integration und Delivery. Zusammengefasst muss der Commit, bzw. der Mehrwert oder das Increment erfolgreich integriert worden sein und alle Tests auf allen implementierten Umgebungen bestehen. Anschließend muss nicht mehr entschieden werden, ob der Mehrwert in der Produktion verwendet werden kann. Dies geschieht mit Continuous Deployment ebenfalls automatisiert [4](#).

TDD, BDD, ATDD

Bei dem Test-driven development (TDD) werden bestehenden Anforderungen zu Beginn in diverse Testfälle gegossen. Anschließend wird die bestehende Software angepasst, sodass die bestehenden Testfälle erfüllt werden. Durch ein solches Vorgehen soll die Implementierung von überflüssigem Quelltext reduziert werden.

Im Kontext des Test-driven developments wird häufig von sehr kurzen Zyklen gesprochen. Diese bestehen aus dem Hinzufügen von Tests, der Prüfung ob die neuen Tests fehlschlagen, dem Implementieren, der erneuten Testausführung und einer abschließenden Phase der Quelltextrestrukturierung.

Grundlegende Prinzipien, welche sich der Entwickler gewahr sein sollte sind bspw. „keep it simple, stupid“ (KISS) oder „You aren't gonna need it“ (YAGNI). Unter Beachtung solcher Prinzipien entsteht überschaubarer, „sauberer“ Quelltext.

Nachteile sind unter anderen das zeitintensive Testen und die Tatsache, dass in der Regel die Tests und der Quelltext von demselben Entwickler verfasst werden. Da dieses Verfahren nur so gut ist wie die Qualität der erstellten Tests, liegt hierin das größte Gefahrenpotential.

Darüber hinaus stoßen test-first Tests dort an ihre Grenzen, wo bspw. komplett Funktionstests implementiert werden müssen (User Interfaces). Diese können nicht implementiert werden bevor die Funktion fertig gestellt wurde [1](#).

Acceptance test-driven development (ATDD) beruht auf demselben Prinzip, wie das Test-driven development. Ein entscheidender Unterschied ist, dass bei diesem Verfahren ebenfalls nach dem test-first Prinzip Akzeptanztests geschrieben werden. Diese Art von Tests sollen die drei Gruppen Entwickler, Tester und den nicht/weniger technischen Part (Kunde, Produktmanager, o.ä.) näher zusammenbringen.

Der Anlass hierfür sind die, sehr häufig drastisch, verschiedenen Verständnisse einer Sachlage und die daraus resultierenden Fehlentwicklungen, zusätzlichen Kosten, etc. Es wird hierbei versucht eine Sprache zu sprechen, indem niedergeschriebene Anforderungen in Akzeptanztests umgewandelt werden. Diese werden anschließend implementiert und nur diese Testfälle im Quelltext implementiert und behoben. Akzeptanztests, welche anschließend hinzugefügt werden, sind neue Anforderungen.

Solche Tests können/sollten von allen Beteiligten gelesen/verstanden werden können. Dies ist beim TDD nicht gegeben [2](#).

Eine Erweiterung des Test-driven developments ist das Behavior-driven development. Die Methode verwendet eine eigene Sprache (domain-specific scripting language = DSL) um Testfälle zu beschreiben. Die verwendete Sprache ist in der Regel strukturiert aufgebaut und sollte auch für „Nicht-Programmierer“ verständlich sein. Hierdurch soll ein noch stärkerer Bezug zu den Akzeptanzkriterien hergestellt werden.

```
Story: Returns go to stock

In order to keep track of stock
As a store owner
I want to add items back to stock when they're returned.

Scenario 1: Refunded items should be returned to stock
Given that a customer previously bought a black sweater from me
And I have three black sweaters in stock.
When he returns the black sweater for a refund
Then I should have four black sweaters in stock.

Scenario 2: Replaced items should be returned to stock
Given that a customer previously bought a blue garment from me
And I have two blue garments in stock
And three black garments in stock.
When he returns the blue garment for a replacement in black
Then I should have three blue garments in stock
And two black garments in stock.
```

Eine Unit wird durch ihr Verhalten beschrieben. Dieses Verhalten sind die Akzeptanzkriterien, aus welchen durch test-first Tests erstellt werden. Diese Tests gehen direkt aus den gegebenen User Stories hervor und dessen Akzeptanzkriterien. Die obige

Abbildung zeigt ein Beispiel für einen solchen Behavior-Test 3.

Unterstützende Tools

Die in den vorangegangenen Kapiteln beschriebenen agilen Vorgehensmodelle und deren Workflows für die Entwicklung komplexer Softwareprodukte benötigen für die Sicherstellung der Softwarequalität kontinuierliche Tests. Dedizierte Server führen die Prozessschritte der vorgestellten Workflows (Continuous Integration, Deployment und Delivery) in sogenannten Pipelines durch. Dazu gehören neben den Modul- und Integrationstests, die der Entwickler auch in seiner lokalen Entwicklungsumgebung durchführen kann, umfangreichere Systemtests bzw. Ende-zu-Ende-Tests. Diese Systemtests benötigen definierte Produktivumgebungen, die sich mit Virtualisierungstechnologien oder neuerdings auch Containerisierungs-Technologien etablieren lassen.

Virtualisierung versus Containerisierung

Bis Anfang der 2010er fand fast ausschließlich die Virtualisierung von Betriebssystemen auf Personal Computern und Servern Anwendung; so auch im Kontext der CI- und CD-Workflows. In den letzten Jahren gewinnt die Technologie der Containerisierung immer mehr an Bedeutung und hat das Potenzial Virtualisierungslösungen teilweise abzulösen. In diesem Kapitel wird zuerst der Bedarf zur Virtualisierung im Kontext von Continuous Software Engineering hergeleitet. Danach wird die jeweilige Funktionsweise der konkurrierenden Technologien grundlegend erklärt, um sie daraufhin vergleichend gegenüberzustellen. Abschließend kommt ein Überblick über etablierte Frameworks.

Bedarf zur Virtualisierung

Im Kapitel zu unterstützenden Workflows wurde der Bedarf von Tests auf allen Integrationsebenen einer Software zur Qualitätssicherung hergeleitet. Bei diesen Tests stehen am Ende die Systemtests, die eine Produktivumgebung für die Applikation voraussetzen. Eine solche Umgebung kann auf dem Hauptbetriebssystem eines Servers konfiguriert werden. Dieses Vorgehen beansprucht einen Server in der Regel nur wenig, da eine Applikation typischerweise ein System nicht voll auslastet. Er ergibt sich somit ein hohes Maß an ungenutzten Ressourcen, was sich negativ auf die Wirtschaftlichkeit solcher Server auswirkt.

Das Konzept der Virtualisierung bietet an dieser Stelle die Möglichkeit die Ressourcen eines Servers deutlich besser auszunutzen, indem entsprechend der Leistungsfähigkeit eines Servers mehrere virtuelle Systeminstanzen erzeugt und parallel ausgeführt werden. Ein weiterer Vorteil der Virtualisierung ergibt sich, wenn eine Applikation auf verschiedenen

Betriebssystemen lauffähig sein soll. Dazu kann für alle zu unterstützenden Betriebssysteme eine separate virtuelle Produktivumgebung bereitgestellt werden. Für verschiedene und parallele Tests genau einer Produktivumgebung kann zudem ein virtuelles Systemabbild beliebig vervielfacht und betrieben werden.

Arten der Virtualisierung

Eine triviale und einfache Form, eine relativ isolierte Umgebung zu schaffen, ist das Partitionieren eines Systems. Diese Vorgehen ermöglicht aber kein wirklich paralleles Ausführen von Applikationen, da es stets zu Kollisionen bei der Verwendung von Hardware-Ressourcen kommen kann.

Tatsächlich brauchbare Möglichkeiten sind die vollständige Virtualisierung, die Paravirtualisierung und die Hardware-Virtualisierung. Wird in der Literatur von Virtualisierung gesprochen ist typischerweise implizit die vollständige Virtualisierung gemeint, da diese Variante meist zur Anwendung kommt. Es beschreibt die Bereitstellung, respektive Simulation eines gesamten Betriebssystems mit Zugriff auf alle Hardware-Ressourcen. Auf die anderen beiden Varianten soll nicht weiter eingegangen werden, da sie aufgrund von bestimmten Nachteilen gegenüber der vollständigen Virtualisierung in der Praxis kaum Anwendungsfälle haben.

Virtuelle Maschinen

Das Konzept der virtuellen Maschine ist die geläufige Technologie, um virtuelle Umgebungen auf einem Hardwaresystem zu betreiben. Die Basis ist das reale Hardwaresystem (Personal Computer oder Server) mit Prozessor, Hauptspeicher, Massenspeicher und diverser weiterer Peripherie. Je nach Zugriffsrechten einer Applikation können von dieser in bestimmten Adressräumen Daten ausgetauscht und Ereignisnachrichten übermittelt werden.

Auf dem Hardwaresystem wird das Gastgeber- bzw. Hauptbetriebssystem ausgeführt, das direkten Zugriff auf die Hardware hat. Im Hauptbetriebssystem wird eine virtualisierende Software als Benutzerprogramm ausgeführt, die man als Virtual Machine Monitor (VMM) oder Hypervisor bezeichnet. Bei einem solchen Hypervisor handelt es sich um einen Typ-2-Hypervisor bzw. Hosted Hypervisor, da er auf dem Hauptbetriebssystem aufsetzt und über dessen Gerätetreiber auf die Hardware-Ressourcen zugreift; das Gegenstück ist der Typ-1-Hypervisor bzw. Native Hypervisor, der als Metabetriebssystem mit eigenen Gerätetreibern direkt auf der Hardware aufsetzt und diese virtuell dupliziert, um sie mehreren Gastbetriebssystemen zur Verfügung zu stellen (vgl. Abbildung). [11](#)

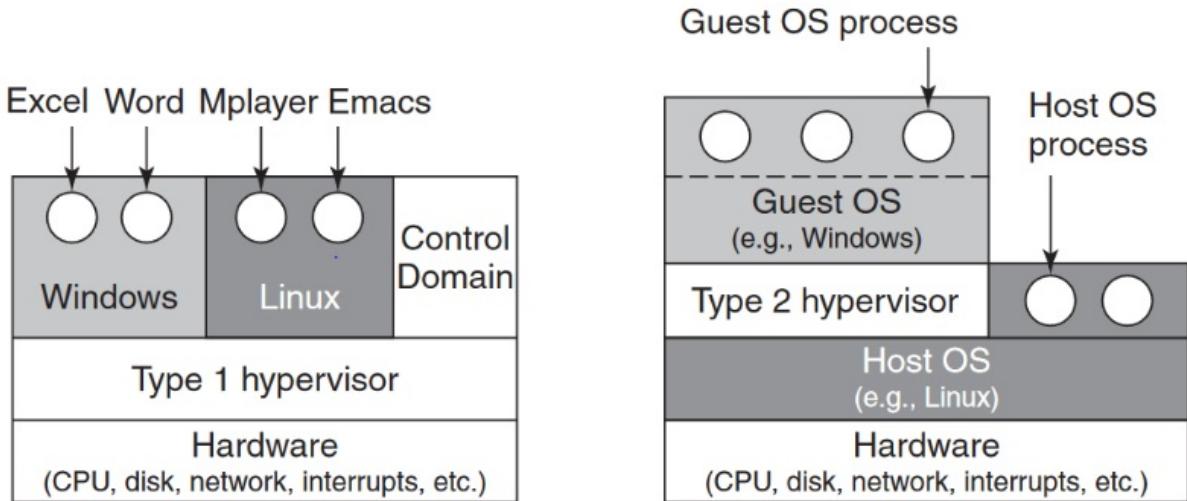


Abbildung: Unterschied zwischen einem Typ-1- und einem Typ-2-Hypervisor [11](#)

Gebräuchlich ist die Virtualisierung mittels Typ-2-Hypervisor (im Folgenden Hypervisor), da dieser betriebssystemspezifisch ist und dadurch eine Virtualisierung auf verschiedenen Hardware-Systemen möglich ist, sofern das Hauptbetriebssystem über die Gerätetreiber verfügt (was bei etablierten Betriebssystemen gegeben ist).

Durch den Hypervisor wird ein Gastbetriebssystem als Benutzerprogramm gestartet, für das ein vollständiges Hardwaresystem simuliert wird. Ein zu lösendes Problem dabei ist das Ausführen von sensitiven Befehlen durch das Gastbetriebssystem, was letzteres als Benutzerprogramm prinzipiell nicht kann. Zur Lösung des Problems verfügen moderne Prozessoren jedoch über Virtualisierungstechnologien mit speziellen Berechtigungsmodi (Intel Virtualization Technology und AMD Secure Virtual Machine). Neben dem Ring-Konzept als Berechtigungskonzept (Ring-0 für Kernmodule mit vollen Zugriffsrechten, Ring-3 für Benutzerprogramme mit eingeschränkten Privilegien) implementieren die Virtualisierungstechnologien das "trap-and-emulate"-Verfahren, das die zwei Modi "Root Operation" und "Non Root Operation" definiert. Der Hypervisor wird im Modus "Root Operation" und das Gastbetriebssystem im Modus "Non Root Operation" ausgeführt. Mit beiden Modi können Ring-3- als auch Ring-0-Befehle ausgeführt werden. Führt jedoch das Gastbetriebssystem Ring-0-Befehle im "Non Root Operation"-Modus aus, können diese durch den Hypervisor im "Root Operation"-Modus abgefangen und verarbeitet werden, was den sensiblen Bereich schützt. [12](#)

Auf einem Continuous Integration Server können auf diese Art mehrere virtuelle Maschinen ausgeführt werden. Auf dem Hauptbetriebssystem läuft die Server-Applikation als Master, während die Gastbetriebssysteme eine Slave-Applikation ausführen und Kontakt zum Master halten. Der Master kann so Bau- und Testaufgaben auslastungsregelnd an seine Slaves delegieren.

Containerisierung

Eine andere Art der Virtualisierung ist die sogenannte Virtualisierung auf Betriebssystemebene bzw. Containerisierung. Bei dieser Virtualisierungsart wird das Prinzip der Speicheraufteilung in Kernel und User Space genutzt, worüber moderne Betriebssysteme für Personal Computer und Server die Zugriffsverwaltung regeln.

Bei dieser Virtualisierungsart werden mehrere Instanzen des User Space erzeugt, die man Container nennt. Die Trennung ergibt sich durch die Verwendung einzigartiger Namespaces für jeden Container. Innerhalb eines Containers wird eine Applikation zusammen mit allen benötigten Bibliotheken und Binaries ausgeführt. Durch die verschiedenen Namespaces sind die Container voneinander unabhängig und isoliert, wodurch es wiederum für die jeweilig eingebetteten Applikationen den Schein eines eigenen Systems macht. Verwaltet werden die Container im Hauptbetriebssystem durch ein Container-Subsystem.

Gemeinsame Systemgrundlage der Container ist der Kernel des Hauptbetriebssystems. Das bringt die Einschränkung mit sich, dass bei Containerisierung kein anderes Betriebssystem als das des Hosts simuliert werden kann. Unter Linux können jedoch verschiedene Distributionen in Containern auf einem Hardwaresystem ausgeführt werden, sofern diese auf der gleichen Kernelversion beruhen.

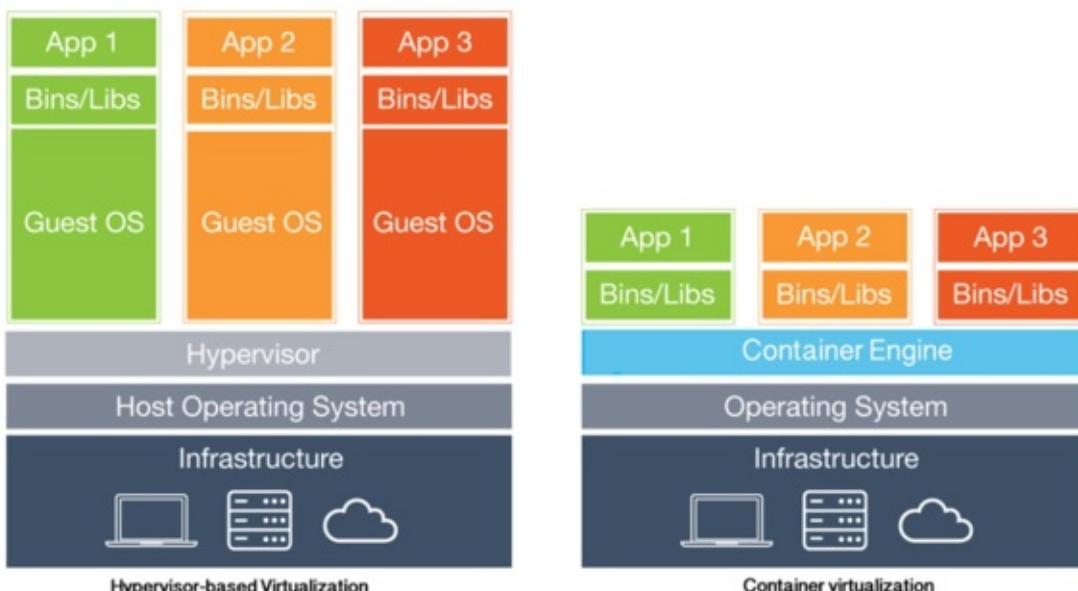


Abbildung: Architektonischer Unterschied zwischen Virtualisierung und Containerisierung [13](#)

Anders als bei einer virtuellen Maschine wird neben dem Hauptbetriebssystem kein weiteres Betriebssystem ausgeführt, weshalb es keinen Hypervisor braucht (vgl. Abbildung). Die verschiedenen Größen der einzelnen Komponenten in der Abbildung verdeutlichen den unglaublich großen Ressourcenbedarf zwischen Virtualisierung- und Container-Frameworks.

Gegenüberstellung

Bei der Bewertung der beiden Technologien sind Konfigurierbarkeit, Simulationsfähigkeiten und Ressourcenbedarf maßgebend. Die Konfigurationsmechanismen bei der Containerisierung sind sehr dynamisch und flexibel durch dedizierte Orchestrierung verschiedener Container. Es können einzelne gekapselte Container entfernt und andere hinzugefügt werden, was am Ende ein neues lauffähiges System ergibt. Virtuelle Maschinen müssen hingegen stets als Ganzes rekonfiguriert und auch wieder verteilt und installiert werden. Hier ist die Containerisierung sehr attraktiv, was der Hauptgrund der wachsenden Anwendung in den letzten Jahren ist.

Hinsichtlich Simulationsfähigkeiten können mit virtuellen Maschinen verschiedene Gastbetriebssysteme beliebigen Typs auf einem Hauptbetriebssystem ausgeführt werden. Container sind hingegen an die systemische Umgebung des Hauptbetriebssystems und dessen Gerätetreiber und Kernelmodule gebunden. Andere Betriebssysteme oder andere Versionen des Hauptbetriebssystems können in Containern nicht simuliert werden.

Beim Ressourcenbedarf liegt der Vorteil bei der Containerisierung, da Container nativ Systembefehle ausführen können. Es braucht kein separates Betriebssystem, was einen Hypervisor erübriggt und einen effizienteren Umgang mit den verfügbaren Ressourcen mit sich bringt. Hinzu kommt, dass virtuelle Maschinen immer ein gesamtes Betriebssystem beinhalten und mindestens mehrere Hundert Megabyte groß sind, während Container teilweise nur wenige Megabyte umfassen. Das macht zusammen mit der dynamischen Konfigurierbarkeit die Verteilung besonders schnell und einfach.

Solange also nicht der Bedarf eines anderen Kernels als der des Hauptbetriebssystems benötigt wird, ist die Containerisierung im Vergleich zur klassischen Virtualisierung im Vorteil. Letztere ist bei der Simulation von verschiedenen Systemen auf einer Hardware weiterhin die einzige Möglichkeit.

Implementierungen und Standards

Im kommerziellen Umfeld sind die Virtualisierungsprodukte der Firma VMware mit leistungsfähigen Hypervisors für virtuelle Maschinen auf Personal Computern und Servern marktführend [14](#). Als einfache und zuverlässige Open Source-Alternative für Personal Computer erfreut sich der Hypervisor VirtualBox der Firma Oracle großer Beliebtheit [15](#).

Das erste Werkzeug, mit dem eine Art Containerisierung möglich war, ist das Linux-Programm chroot (change root), das durch das Ändern des Stammverzeichnis' eines Prozess' die Kapselung einer Applikation ermöglicht [16](#). Und obwohl andere Frameworks

wie Solaris Containers oder FreeBSD jail bereits seit Anfang der 2000er Jahre verfügbar sind [17](#) [18](#), wurde die Containerisierung erst 2013 mit dem Release des Frameworks Docker durch die gleichnamige Firma Docker Inc. populär.

Docker setzte sich im Laufe der letzten Jahre sowohl im Open Source als auch im industriellen bzw. kommerziellen Umfeld durch und ist mittlerweile der defacto-Standard (vgl. Abbildung) [19](#).

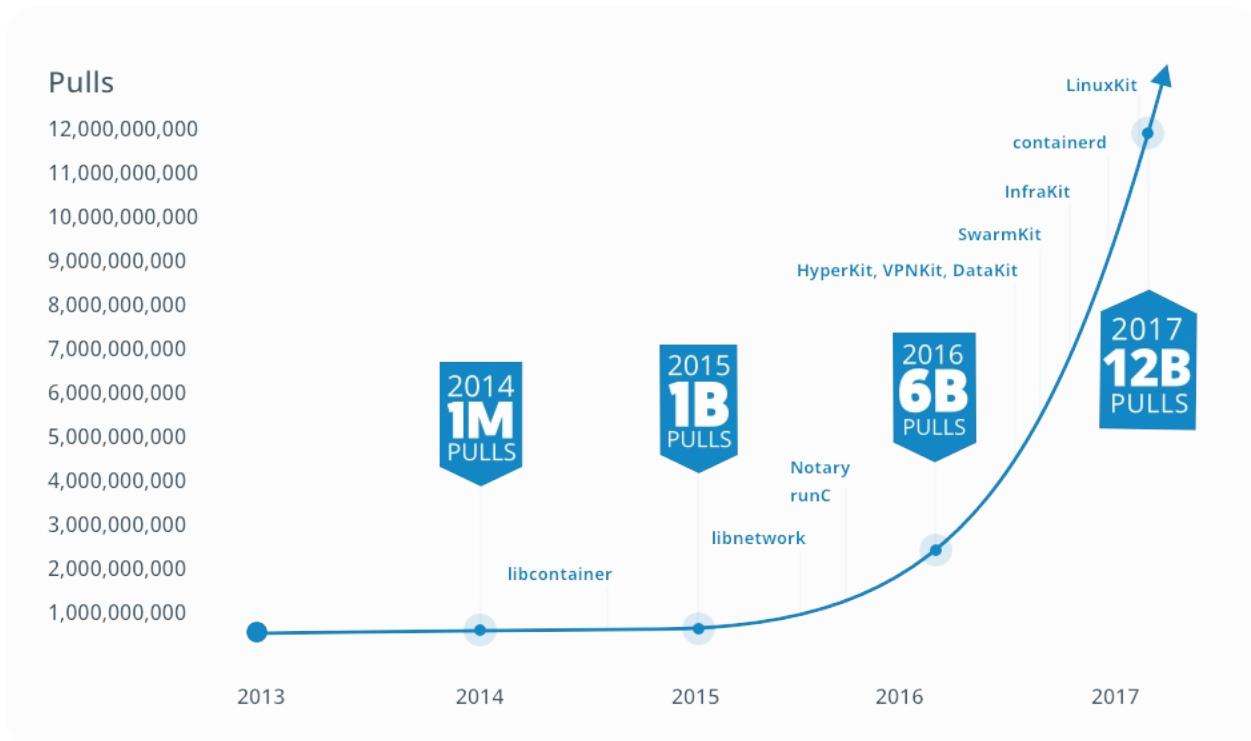
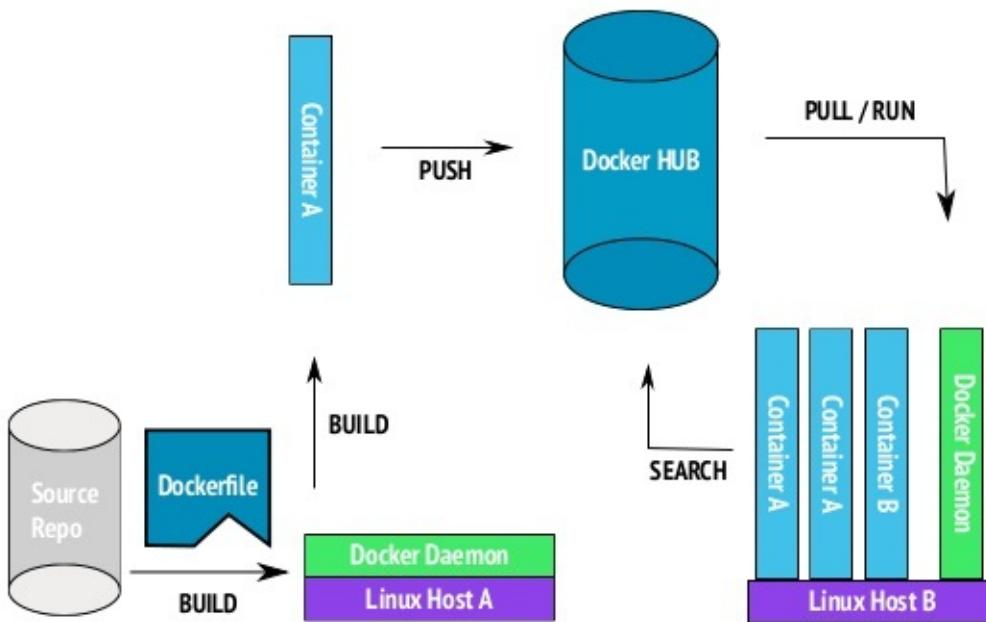


Abbildung: Veranschaulichung des Wachstums des Docker-Ökosystems anhand der steigenden Anzahl an Pulls auf DockHub [19](#)

Das Docker-Framework besitzt mit der Docker-Engine sein eigenes Container-Subsystem zur Ausführung von Containern. Die Implementierung basiert auf der Container-Runtime `containerd` [20](#). Die Stärke des Docker-Frameworks liegt in der Verfügbarkeit von etlichen standardisierten Images aller gebräuchlich Systemkomponenten (JRE, .NET, Apache Server, SQL Datenbank u.v.m.), die dynamisch konfiguriert und miteinander verknüpft werden können. Bereitgestellt werden alle Images über die Online-Plattform DockHub, von der die lokale Docker-Applikation bei Bedarf auch automatisch benötigte Images bezieht, um einen Container zusammenzubauen. Neue standardisierte Images können von jedermann implementiert und über DockHub angeboten werden. Die Konfiguration eines Containers erfolgt entweder aus der Konsole heraus über den parametrisierten Startbefehl oder mit einem Konfigurationsskript in YAML über das separate Werkzeug Docker-Compose (vgl. Abbildung). Letzteres macht die Verwendung im Kontext von CI-Servern sehr attraktiv.

Docker: Workflow



Milano - 2015 / 03 / 11

Abbildung: Erzeugung eines konfigurierten Containers mit Docker-Compose 21

Build Server

von Lukas Taake

Build Server sind ein zentraler Bestandteil im Continuous Software Engineering. Sie stellen einen zentralen Ort dar, an dem der aktuelle Stand der Software, z.B. in Form von Testergebnissen und kompilierten Artefakten, einsehbar ist.

Ein großer Vorteil ist, dass alle Prozesse innerhalb eines Build Server vollautomatisch ablaufen und somit Fehler durch falsche, ausgelassene oder vertauschte Prozess-Schritte ausgeschlossen werden können. Darüber hinaus stellt ein zentraler Build Server eine isolierte Umgebung dar, sodass es nicht dazu kommen kann, dass die Software aufgrund unterschiedlicher Konfiguration auf einem Entwickler-Rechner funktioniert (bzw. Tests erfolgreich sind) und auf einem anderen nicht. Wenn also der Build-Server erfolgreich ist und ein Entwickler-Rechner nicht, muss es sich um eine Fehlkonfiguration des Entwicklers handeln, da die des Build-Servers im Optimalfall möglichst nah an der eines Produktivsystems liegt.

Über das Kompilieren und Testen hinaus werden Build-Server auch zur Durchführung von Continuous Delivery und Deployment verwendet. Oftmals existieren bereits Build-Server-Plugins zur Unterstützung verbreiteter Deployment-Ziele.

Die konkrete Funktionsweise soll Beispielhaft am Beispiel von Jenkins und Travis CI dargestellt werden, wobei Jenkins ein Open Source Projekt ist und weitreichende Konfigurationsmöglichkeiten bietet, während es sich bei Travis um eine SaaS handelt.

Jenkins

Jenkins ist eins der populärsten CI-Tools, vor allem im Java-Umfeld. Üblicherweise wird es selbst gehostet und ist tiefgreifend konfigurierbar. Es unterstützt verschiedene Build-Tools, wie Ant, Maven, Gradle oder MSBuild, außerdem Versionskontrollsysteme, wie Git, Mercurial oder SVN. Darüber hinaus gibt es eine Vielzahl an Plugin, über die es um neue Funktionalität erweitert werden kann. (vgl. [9])

Gewöhnliche Build-Jobs können zwar aus mehreren auszuführenden Schritten bestehen, liefern als Ergebnis allerdings nur das Gesamtergebnis (und evtl. einen Test-Report) zurück. Im Gegensatz dazu gibt es auch Pipeline-Jobs, bei denen ein Job explizit in mehrere Stufen (Stages) unterteilt wird (z.B. Compile, Unit Test, Integration Test). Dann wird der Status jeder einzelnen Stufe ausgegeben, wenn also ein Integrationstest fehlschlägt, ist das sofort ersichtlich (vgl. Abbildung).

Stage View

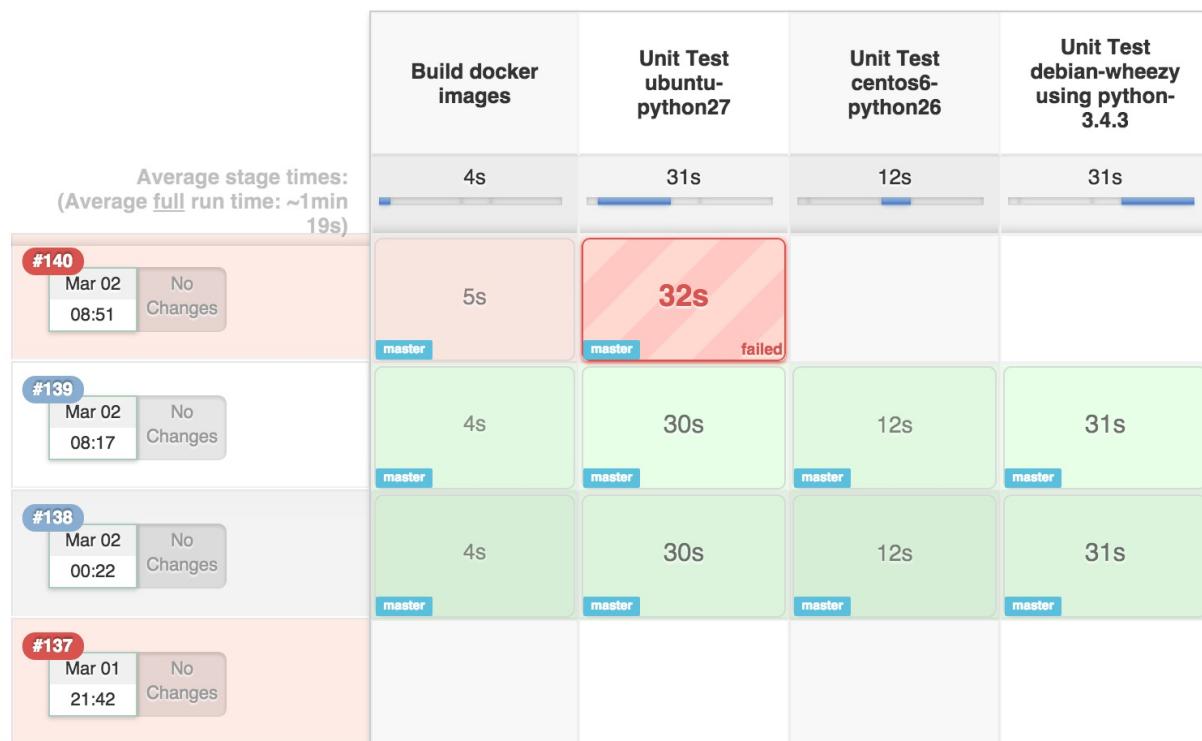


Abbildung: Übersicht der Build-Historie mit einzelnen Stages. Quelle: [Jenkins Wiki, Pipeline Stage View Plugin](#)

In komplexen Setups, oder wenn viele Projekte zentral verwaltet werden, ist es möglich, einzelne Build-Jobs auf Slave-Knoten zu verteilen. In Pipeline-Jobs ist es sogar möglich, verschiedene Stages auf verschiedenen Slaves ausführen zu lassen, um die Geschwindigkeit zu steigern.

Travis CI

Travis CI verfolgt einen anderen Ansatz als Jenkins, indem es ausschließlich als Software as a Service angeboten wird und als Versionskontrollsystem ausschließlich Github-Repositories akzeptiert. Dafür ist als Ausgleich zur dieser Einschränkung die Integration zu Github sehr komfortabel. So können Build Jobs für neue Repositories mit einem Maus-Klick erstellt werden und die Konfiguration geschieht über eine YAML-Datei (siehe Listing).

```
language: java

jdk: oraclejdk8

services:
- mysql

before_install:
- mysql -e 'CREATE DATABASE newsboard'
- mysql newsboard < src/main/resources/sql/create_script.sql
- mysql newsboard < src/main/resources/sql/example_data.sql

install: mvn install -DskipTests=true -Dmaven.javadoc.skip=true
script: mvn verify -Dspring.profiles.active=travis
```

Listing: Beispielhafte Travis CI Build Definition

Jede Ausführung eines Build-Jobs geschieht auf einer neu erstellten virtuellen Maschine um absolute Isolation sicherzustellen. Über die Angabe der Programmiersprache in der Konfiguration wird bestimmt, wie die Maschine vorkonfiguriert ist. Darüber hinaus können mit `services` auch Abhängigkeiten, z.B. Datenbanken, installiert werden und mit `before_install` initialisiert werden. Mit `install` und `script` werden die Befehle zum Kompilieren und Ausführen der Tests angegeben.

Auf den ersten Blick ähnlich dem Konzept der Pipeline bei Jenkins sind die Build Stages bei Travis. Die Verwendung ist allerdings im Gegensatz zu Jenkins Pipelines ausschließlich sinnvoll, um unterschiedliche zeitaufwändige Teststufen parallel ablaufen zu lassen. Es wird nämlich für jede einzelne Stage eine eigene virtuelle Maschine erstellt und die Software dort kompiliert. Um also z.B. Unit- und Integrationstests zu parallelisieren, sind sie nicht unbedingt geeignet. Darüber hinaus ist es nicht möglich, ohne zuhilfenahme externer Dienste, Daten (z.B. Test Reports) zwischen den einzelnen Stages auszutauschen. Eine Gesamt-Coverage unter Berücksichtigung aller Teststufen wäre so nicht mehr möglich. (vgl. [8])

Quellen

- [1] Test-driven development; https://en.wikipedia.org/wiki/Test-driven_development
- [2] Acceptance test-driven development;
https://en.wikipedia.org/wiki/Acceptance_test%20%93driven_development
- [3] Behavior-driven development; https://en.wikipedia.org/wiki/Behavior-driven_development
- [4] The Product Managers' Guide to Continuous Delivery and DevOps;
<http://www.mindtheproduct.com/2016/02/what-the-hell-are-ci-cd-and-devops-a-cheatsheet-for-the-rest-of-us/>
- [5] What is the difference between Continuous Integration, Continuous Deployment & Continuous Delivery?; <https://codeship.com/continuous-integration-essentials>
- [6] The Future of Programming, Robert C. Martin; <https://youtu.be/eclWPzGEbFc>
- [7] Kanban (Software Entwicklung);
[https://de.wikipedia.org/wiki/Kanban_\(Softwareentwicklung\)](https://de.wikipedia.org/wiki/Kanban_(Softwareentwicklung))
- [8] Build Stages - Travis CI; <https://docs.travis-ci.com/user/build-stages/>
- [9] Jenkins (Software); [https://de.wikipedia.org/wiki/Jenkins_\(Software\)](https://de.wikipedia.org/wiki/Jenkins_(Software)) [10] Jan Bosch, "Continuous Software Engineering", Springer International Publishing, Switzerland, 2014, Link: <http://www.springer.com/us/book/9783319112824> (Zugriff am 24.06.2017) [11] Andrew S. Tanenbaum, "Modern operating systems", Boston, 2015 [12] Gerald J. Popek, "Formal Requirements for Virtualizable Third Generation Architectures", University of California, Los Angeles, 1974 [13] Chenxi Wang, InfoWorld (IDG Communications, Inc.), 2017, Link: <http://www.infoworld.com/article/3204171/linux/what-is-docker-linux-containers-explained.html> (Zugriff am 26.06.2017) [14] Contel Bradford, "Virtualization Wars: VMware vs. Hyper V: Which is Right For Your Virtual Environment?", StorageCraft Technology Corporation, 2015, Link: <http://www.storagecraft.com/blog/virtualization-wars-vmware-vs-hyper-v-which-is-right-for-your-virtual-environment/> (Zugriff am 02.17.2017) [15] Heise Medien GmbH & Co. KG, 2017, Link: <https://www.heise.de/download/product/virtualbox-40385> (Zugriff am 02.07.2017) [16] Scott Hogg, "Software Containers: Used More Frequently than Most Realize", Network World (IDG Communications, Inc.), 2014, Link: <http://www.networkworld.com/article/2226996/cisco-subnet/software-containers--used-more-frequently-than-most-realize.html> (Zugriff am 28.06.2017) [17] Mike DeGraw-Bertsch, "FreeBSD jails", BSD DecCenter (O'Reilly Media, Inc.), 2003, Link: <http://www.onlamp.com/pub/a/bsd/2003/09/04/jails.html> (Zugriff am 28.06.2017) [18] Tucker Andrew G. et al., "Global visibility controls for operating system partitions", European Patent Office, 2005, Link: https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20050127&DB=&locale=en_EP&CC=US&NR=2005021788A1&KC=A1&ND=1 (Zugriff am 28.06.2017) [19] "What is a Container", Docker Inc., 2017, Link: <https://www.docker.com/what-container> (Zugriff am 01.07.2017) [20] "About Containerd",

Docker Inc., 2016, Link: <https://containerd.io> (Zugriff am 02.07.2017) [21] "Docker",
f2informatica, Milano, 2015, Link: <https://www.slideshare.net/Pokerone/docker-45628208>
(Zugriff am 01.07.2017)

AI

Teilbereich(e): Data Mining

Projektleiter:

Projektteam: Daniel Beneker, Sven Schirmer, Yannick Kloss

Github-Repo: [Twitter Miner](#)

Übersicht Data Mining Algorithmen

Die Datamining Algorithmen lassen sich grob in vier Kategorien untergliedern (vgl. Runkler S. 3):

- Klassifikation
- Clusteranalyse
- Korellationsanalyse
- Regressionsanalyse

Klassifikation

Klassifikation setzt man grundsätzlich ein, wenn man einen gelabelten Datensatz zur Verfügung hat. Gelabelt heißt in diesem Kontext, dass jedem Tupel eine Klasse zugeordnet wurde und das Modell anhand dieser Information trainieren kann. In der englischsprachigen Literatur ist dies unter supervised learning bekannt. Die Klassifikation bietet den großen Vorteil, dass eine Überprüfung des Algorithmus anhand der gelabelten Testdaten stattfinden kann. Hierfür wird der Quelldatensatz aufgeteilt; zum einen in einen Lerndatensatz für das Training des Modells und zum anderen in einen Testdatensatz für die spätere Evaluierung des Algorithmus.

Einige bekannte Algorithmen: (teilweise entnommen aus Tan)

- Naive Bayes
- Decision Tree
- Decision Forest
- Rule-Based
- Support vector machine
- Neural network

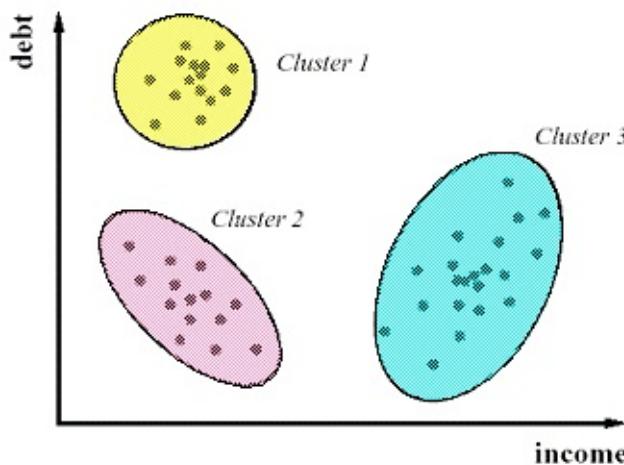
Die Algorithmen Naive Bayes, Decisiontree und Support Vector Machine sind in den folgenden Kapiteln detailliert beschrieben. In unserem Softwareprojekt der Twitter-Text-Analyse ist ein gelabelter Datensatz vorhanden, sodass das Projektteam die Klassifikation mit den drei genannten Algorithmen vornehmen kann.

Clusteranalyse

Die Clusteranalyse ist ein unsupervised learner. Beim unsupervised learning besitzen die Quelldaten kein Label. In den Quelldaten werden Ähnlichkeiten und Muster zwischen den Tupeln gesucht und die Tupel in Cluster aufgeteilt. Die Tupel sollen in ihrem Cluster

möglichst ähnlich sein und verschieden zu den Tupeln in den anderen Clustern. Somit ist auch eine Evaluierung möglich: denn ein gutes Clustering hat eine hohe Separation zwischen den einzelnen Clustern. (vgl. Cichosz S. 15)

Eine Beispielvisualisierung mit den Attributen *debt* und *income*.



Entnommen aus <https://www.analyticsvidhya.com/wp-content/uploads/2013/11/Clust1.gif>.

Einige bekannte Algorithmen:

- k-means
- k-medoids
- fuzzy c-Mean

Korrelationsanalyse

Die Korrelationsanalyse untersucht die Korrelation (Abhängigkeit) zwischen den Attributen über alle Daten. Die Stärke wird durch den Korrelationskoeffizienten ausgedrückt. Das Ergebnis der Korrelationsanalyse ist meistens eine Korrelationsmatrix, welche alle Korrelationskoeffizienten der Attribute zueinander enthält.

Attributes	Passen...	Name	Sex	Age	No of Si...	No of P...	Ticket N...	Passen...	Cabin	Port of ...	Life Boat	Survived
Passenger Class	1	0.898	0.125	-0.408	0.061	0.018	0.884	-0.559	0.546	0.038	0.431	0.312
Name	0.898	1	0.125	-0.344	0.066	0.004	0.989	-0.481	0.732	0.054	0.400	0.294
Sex	0.125	0.125	1	0.064	-0.110	-0.213	0.122	-0.186	0.060	-0.122	0.055	0.529
Age	-0.408	-0.344	0.064	1	-0.244	-0.151	-0.336	0.179	-0.145	0.049	-0.120	0.056
No of Siblings or Spouses on Board	0.061	0.066	-0.110	-0.244	1	0.374	0.073	0.160	-0.065	-0.074	-0.088	0.028
No of Parents or Children on Board	0.018	0.004	-0.213	-0.151	0.374	1	0.002	0.222	-0.002	-0.096	-0.112	-0.083
Ticket Number	0.884	0.989	0.122	-0.336	0.073	0.002	1	-0.497	0.638	0.059	0.410	0.291
Passenger Fare	-0.559	-0.481	-0.186	0.179	0.160	0.222	-0.497	1	-0.224	0.062	-0.370	-0.244
Cabin	0.546	0.732	0.060	-0.145	-0.065	-0.002	0.638	-0.224	1	0.031	0.090	0.095
Port of Embarkation	0.038	0.054	-0.122	0.049	-0.074	-0.096	0.059	0.062	0.031	1	0.170	-0.100
Life Boat	0.431	0.400	0.055	-0.120	-0.088	-0.112	0.410	-0.370	0.090	0.170	1	0.013
Survived	0.312	0.294	0.529	0.056	0.028	-0.083	0.291	-0.244	0.095	-0.100	0.013	1

Diese Korrelationsmatrix wurde mithilfe von Rapidminer erstellt und zeigt die Koeffizienten des "Titanic-Datensatzes". Der Datensatz ist zu finden unter <https://www.kaggle.com/c/titanic/data>. Auffällig ist die Korrelation zwischen "Sex" und "Survived".

Regressionsanalyse

Die Regressionsanalyse lässt sich am ehesten mit der Klassifizierung vergleichen. Doch hier wird keine diskrete Klasse vorhergesagt, sondern ein numerischer Wert. Dieser Wert wird durch die Regression bestimmt. (vgl. Cichosz S. 14)

Quellen

- Thomas A. Runkler: Data Mining - Modelle und Algorithmen intelligenter Datenanalyse; 2. Auflage; Springer; 2015
 - Tan, Steinbach, Kumar: Data Mining - Classification: Basic Concepts, Decision Trees, and Model Evaluation; 2004; https://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap4_basic_classification.pdf Aufgerufen am 01.07.2017.
 - Pawet Cichosz: Data Mining Algorithms - Explained Using R; Wiley; 2015; <http://pdf.th7.cn/down/files/1502/Data%20Mining%20Algorithms.pdf> Aufgerufen am 01.07.2017.
-

Author: Sven Schirmer

Knowledge Discovery in Databases (KDD)

Das Volumen der jährlich generierten digitalen Datenmenge ist in den letzten Jahren massiv angewachsen. Doch nur eine geringe Menge dieser Daten wird für Analysen genutzt.^[5] *Knowledge Discovery in Databases* beschreibt das generelle Vorgehen, um genau diese Daten zu nutzen, dass heißt Wissen zu generieren.

Definition

Der Begriff *Knowledge Discovery in Databases* wird in der Literatur unterschiedlich verwendet. Es kommt vor, dass es als Synonym zu dem Begriff *Data Mining* genutzt.^[4] Üblicherweise beschreibt *Knowledge Discovery in Databases* aber einen Prozess in dem *Data Mining* lediglich einen Teilschritt darstellt.^[2]

Das *Gabler Wirtschaftslexikon* definiert KDD wie folgt:

Knowledge-Discovery-in-Databases (KDD)-Prozess; umfassender Datenanalyseprozess, in dessen Kern Verfahren des Data Mining zur Anwendung kommen. Der-Knowledge-Discovery-in-Databases (KDD)-Prozess umfasst folgende Phasen:[...]^[3]

Hung (2009) beschreibt KDD als einen:

[...] nichttrivialer mehrstufiger Prozess der Wissensfindung aus vorhandenen Informationen. KDD-Prozess umfasst alle Schritte, von woher die Daten abgeholt werden, über Vorverarbeitung und eigentliche Verarbeitung zur Informationsgewinnung (Data-Mining-Schritt), bis hin wie die Endinformation interpretiert und dargestellt wird.^[2]

KKD-Prozess

Fayyad et al. beschrieben 1996 erstmals die einzelnen Schritte, die bei der *Knowledge Discovery in Databases* genutzt werden. Dazu entwickelten sie ein Modell, welches heutzutage oft unter den Begriffen "KDD Prozess" oder "Schritte des KDD" zu finden sind.

Nach Fayyad et al. besteht KDD aus den folgenden neun Schritten:^[1]

1. Problemabgrenzung
2. Auswahl der Daten
3. Datenvorverarbeitung
4. Datenreduktion und Kodierung
5. Auswahl der Data Mining Methode

6. Auswahl des Data Mining Algorithmus
7. Data Mining
8. Interpretation der Ergebnisse
9. Anwendung des gefundenen Wissens

Das *Fayyad Modell des KDD Prozesses* ist in Abbildung 1 dargestellt. Es handelt sich im ein iteratives Modell, dass bedeutet, die Schritte können mehrfach durchlaufen werden. Nach der Evaluation der Ergebnisse müssen eventuell einige Schritte neu durchlaufen werden, um das Ergebnisse in der Evaluation zu verbessern. So kann es beispielsweise sein, dass in der Evaluation auffällt, dass bei der Vorverarbeitung nicht alle unerwünschten Daten herausgefiltert wurden. [1][2]

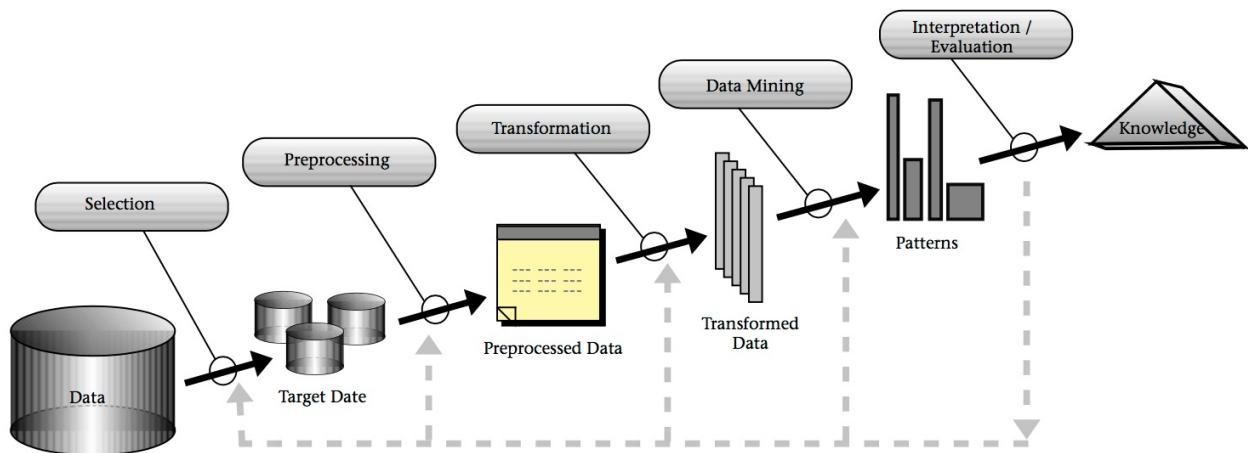


Abbildung 1: Fayyad Modell des KDD Prozesses [1]

Die Schritte im Detail

1. Problemabgrenzung: Im ersten Schritt muss zunächst das Ziel aus der Sicht des Kunden identifiziert werden. Außerdem ist es wichtig sich mit dem jeweiligen Fachbereich vertraut zu machen. [1]

2. Auswahl der Daten: Im zweiten Schritt wird ein Datensatz ausgewählt. Das Ziel aus Schritt eins sollte dabei beachtet werden, denn das erreichen des Ziels hängt von der Qualität des Datensatzes ab. [1]

3. Datenvorverarbeitung: Im dritten Schritt werden die Daten vorverarbeitet. Unerwünschte Informationen, die die Ergebnisse des Data Minings negativ beeinflussen würden, sollten herausgefiltert werden. In diesem Schritt sollte auch entschieden werden, wie mit fehlenden/unvollständigen Datenfeldern umgegangen wird. [1]

4. Datenreduktion und Kodierung: Im vierten Schritt werden die Daten in eine andere Form überführt, damit Data Mining Algorithmen sie verstehen und verarbeiten können. Dieses Verfahren wird oft auch "Kodierung", "Datenprojektion" oder "Datentransformation"

genannt. Bei der Datentransformation wird i.d.R. auch eine Datenreduktion durchgeführt, bei der Daten mit geringer Aussagekraft entfernt werden. [1]

5. Auswahl der Data Mining Methode: Im fünften Schritt wird mit Hilfe des Ziels aus Schritt 1 eine bestimmte Data Mining Methode ausgewählt. Bekannte Methoden sind beispielsweise die Klassifikation oder die Clusteranalyse. [1]

6. Auswahl des Data Mining Algorithmus: Im sechsten Schritt wird innerhalb der zuvor gewählten Data Mining Methode ein Algorithmus ausgewählt. Zudem wird entschieden mit welchen Parametern dieser Algorithmus am besten aufgerufen wird. [1]

7. Data Mining: Der siebte Schritt ist das eigentliche Data Mining. [1]

8. Interpretation der Ergebnisse: Im achten Schritt werden die Ergebnisse des Data Minings interpretiert. Dies kann auch eine Visualisierung umfassen. [1]

9. Anwendung des gefundenen Wissens: Im neunten Schritt werden die Ergebnisse angewendet. Das bedeutet beispielsweise das Wissen in ein anderes System weiter zu geben, oder auch einfach eine Dokumentation der Ergebnisse anzufertigen. Weiter sollte auf Konflikte zu dem bisherigen Wissensstand geprüft werden. [1]

Einordnung dieses Projektes in den KDD Prozess

Dieses Projekt "Twitter-Miner" durchläuft alle Schritte des KDD Prozesses. In diesem Kapitel soll ein kurzer Überblick gegeben werden, was in diesem Projekt in den jeweiligen Schritten gemacht wurde und wo weitere Informationen zu finden sind.

1. Problemabgrenzung: Die Problemabgrenzung und Definition des Ziels wurde u.a. im Pflichtenheft festgelegt. Ziel ist die Stimmungsanalyse von Tweets (=Sentiment Analysis).

2. Auswahl der Daten: Die Datenbasis dieses Projektes bilden Tweets. Zu einem bestimmten Hashtag werden über eine Schnittstelle Tweets geladen.

3. Datenvorverarbeitung: Tweets können viele unerwünschte Informationen enthalten wie beispielsweise Links, die in diesem Schritt entfernt werden. Weiteres zum Thema Datenvorverarbeitung ist [hier](#) zu finden.

4. Datenreduktion und Kodierung: Um die Tweets als Eingabe für unsere Data-Mining-Algorithmen zu nutzen wurden sie in entsprechende Zahlenrepräsentationen umgewandelt. Innerhalb dieses Projektes wurden dafür sogenannte *tf-idf Vektoren* genutzt. Dieses Thema wird ebenfalls in [diesem Kapitel](#) beschrieben.

5. Auswahl der Data Mining Methode: Das Ziel aus Schritt 1 (Sentimentanalyse) lässt sich mit der Data Mining Methode "Klassifikation" erreichen. Klassifikationsalgorithmen können bestimmen, ob ein Merkmal zu einer bestimmten Klasse gehört oder nicht. Das heißt bei der

Sentimentanalyse gibt der Klassifikationsalgorithmus aus, ob es sich um einen positiven Tweet oder einen negativen Tweet handelt.

Eine generelle Übersicht der Data Mining Methoden ist [hier](#) aufgelistet.

6. Auswahl des Data Mining Algorithmus: Innerhalb dieses Projektes werden die drei folgenden Klassifikationsalgorithmen genutzt.

- Bayes
- Support Vector Machine
- Decision Tree

7. Data Mining: In diesem Schritt wird ein Datensatz auf mehrere Tweets mit den Algorithmen aus Schritt 6 klassifiziert.

8. Interpretation der Ergebnisse: Die Ausgabe der Algorithmen wird in diesem Schritt interpretiert. Ist die Ausgabe beispielsweise [0,1], so ist der entsprechende Tweet positiv, ist die Ausgabe [1,0], so ist er negativ.

9. Anwendung des gefundenen Wissens: Die Anwendung des gefundenen Wissens erfolgt in einer Präsentation durch eine Weboberfläche.

1. Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3), 37. [↔↔↔](#)
 2. Hung, P. T. (2009). Data-Mining und Knowledge Discovery in Databases (KDD) Ein Überblick. Dresden. Retrieved from https://www.inf.tu-dresden.de/content/institutes/iai/tis-neu/lehre/archiv/folien.ws_2008/Vortrag_Hung.pdf [↔↔↔](#)
 3. Springer Gabler Verlag (Herausgeber), Gabler Wirtschaftslexikon, Stichwort: Knowledge Discovery in Databases (KDD), online im Internet:
<http://wirtschaftslexikon.gabler.de/Archiv/75635/knowledge-discovery-in-databases-v10.html>
↔
 4. Alpaydın, E. (2014). Introduction to machine learning. Methods in Molecular Biology (Second Edi, Vol. 1107). The MIT Press. [↔](#)
 5. Gantz, John und David Reinsel (2012). IDC IVIEW: THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East. Techn. Ber. URL: <http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf> [↔](#)
-

Datenvorverarbeitung

In den meisten Fällen (abhängig von der Datenquelle) sind rohe Textdaten nicht gut formatiert, nicht standardisiert und nicht strukturiert. Mithilfe von diversen Techniken aus dem Bereich des *text-preprocessing* sollen die Daten in definierte Sequenzen aus linguistischen Komponenten umgewandelt werden, welche eine standardisierte Struktur und Notation haben. [1]

Methodenübersicht

Nachfolgend werden einige wohlbekannte Methoden mit einer kurzen Beschreibung aufgelistet. Die Verwendung dieser Methoden, sowie deren Reihenfolge, ist von der Anwendung bzw. der Domäne abhängig: [1]

Text Bereinigung: Oftmals beinhalten rohe Textdaten unbekannte und/oder unnötige Token und Zeichen, welche für weitere Verarbeitungen entfernt werden sollten. Beispielsweise sind HTML-Tags oder bestimmte Daten von XML- oder JSON-Strukturen nicht Notwendig.

Text Tokenisierung: Üblicherweise werden Textdaten für weitere Verarbeitungsschritte in Token (Einheiten der Wortebene, z. B. Wörter, Sätze oder Absätze) extrahiert. Beispielsweise ist eine sehr einfache Variante einen Text auf Basis der Leerzeichen zu Tokenisieren. In Abhängigkeit von der Domäne kann dieser Prozess komplexer sein.

Entfernung von Speziellen Zeichen: Einige Zeichen sind für weitere Verarbeitungsschritte hinderlich. Beispielsweise sind Satzzeichen oder Sonderzeichen in vielen Anwendungen zu entfernen. Die Domäne muss berücksichtigt werden: Eventuell könnten Smilies, bestehend aus speziellen Zeichen, für die Anwendung von Bedeutung sein.

Kontraktionen erweitern: Mit Kontraktion ist die Zusammenziehung zweier oder mehr Wörter zu einem Wort gemeint (Bspw.: "you will" -> "you'll"). Die Erweiterung von Kontraktionen in ihre Wortbestandteile kann in vielen Anwendung Notwendig sein. Sprachen haben üblicherweise Listen von Kontraktionen, welche angewendet werden können.

Umwandlung von Groß- und Kleinbuchstaben: Die Umwandlung von Zeichen in Groß- oder Kleinbuchstaben macht viele Anwendung einfacher: Beispielsweise das matchen von Wörtern. Die Umwandlung von Textdaten in Groß- oder Kleinbuchstaben ist üblicherweise eines der ersten Schritte in der Datenvorverarbeitung.

Entfernung von Stoppwörtern: Stoppwörter sind Wörter, welche in der Anwendung nur eine kleine oder keine Bedeutung haben. Es gibt keine universelle Liste von Stoppwörtern, jede Sprache und Domäne hat ihre eigenen Listen. Typische Stoppwörter der Englischen

Sprache sind beispielsweise *a*, *the*, *me* und so weiter.

Wortkorrektur: Die Korrektur von Wörtern gehört zu den größten Herausforderungen im Bereich der Datenvorverarbeitung von Textdaten. Zu der Korrektur von Wörtern gehören u. a. das Verbessern von falsch geschriebenen Wörtern und Zeichen Wiederholungen. Oftmals ist das Verbessern kontraproduktiv: In Abhängigkeit von der Domäne können Zeichen Wiederholungen beispielsweise einen intensiven emotionalen Ausdruck unterstreichen (finallllyyyyyy), welche für eine weitere Verarbeitung von Bedeutung sein könnten.

Stammformreduktion: Stammformreduktion (Stemming) ist das Zurückführen von verschiedenen morphologischen Varianten eines Wortes auf den Wortstamm ("having" -> "have"). Stammformreduktionen werden für eine verbesserte Verarbeitung von Textdaten angewendet.

Lemmatisierung: Mithilfe von Lemmatisierungs Verfahren werden Wörter auf ihre Grundform zurückgeführt ("better" -> "good").

Verwendete Methoden für Twitter Datensätze

Die nachfolgenden Methoden wurden für das Projekt angewendet:

1. Text Bereinigung: Tweets können HTML-Tags beinhalten um u. a. Wörter hervorzuheben. Mit der wohlbekannten Python Web-Scraping Library *Beautifulsoup* werden für dieses Projekt sämtliche relevanten Inhalte eingelesen und HTML-Tags ausgelassen.
2. Text Tokenisierung: Tweets beinhalten viele domänenspezifische Zeichenketten, welche als Token eine Bedeutung haben, bspw.: Smilies, @-Mentions, Hashtags, und so weiter. Das *Natural Language Toolkit (NLTK)* beinhaltet u. a. eine Tokenisierungs Funktion für Tweets, welche im Rahmen dieses Projekts verwendet wird.
3. Kontraktionen erweitern: Für die Sentimentanalyse ist die Erweiterung von Kontraktionen von Bedeutung, um u. a. Negierungen als einzelne Tokens hervorzuheben ("won't" -> "will not").
4. Entfernung von Speziellen Zeichen: Mithilfe des Attributs *string.punctuation* werden in der Python Anwendung alle Token entfernt, welche aus einer Interpunktions bestehen.
5. Entfernung von Stopwörtern: Das bereits erwähnte NLTK beinhaltet eine Liste von Stopwörtern der Englischen Sprache. Sämtliche Tokens, welche aus einem dieser Stopwörter bestehen, werden entfernt.
6. Stammformreduktion: Für dieses Projekt wird eine Stammformreduktion anstatt Lemmatisierung verwendet. Der Grund ist, dass die Grundformen von Wörtern die emotionalen Ausdrücke abschwächen könnten. In der Sentiment Analyse ist dies jedoch von Bedeutung.

Der folgende Code-Abschnitt zeigt die Ausgabe der Datenvorverarbeitung eines beispielhaften Texts:

```
normalized_text = preprocess_text(["RT This is a text! Is that great? @-User -> :) $199,  
that's good. You can't win. I'm hoping you're in a good mood :P !!!"])  
print(normalized_text)  
-> ['text great user -> :) 199 good cannot win hope good mood :p']
```

1. Sarkar, D.: Text Analytics with Python. Apress (Herausgeber) (2016) ↪

Author: Yannick Kloss

Decision Tree

In den nachfolgenden Kapiteln werden allgemeine Grundlagen verdeutlicht sowie eine Implementierung vorgestellt.

Author: Yannick Kloss

Grundlagen

Definition

Decision Trees werden im Data Mining im allgemeinen verwendet um den Wert einer Zielvariable, basierend auf diversen Input Variablen, zu bestimmen. Decision Trees dienen der Darstellung von Entscheidungsregeln. Die hierarchische Darstellung in Baumdiagrammen veranschaulichen aufeinanderfolgende Entscheidungen. [1]

Die Erstellung bzw. das Trainieren eines Decision Trees ist typischerweise ein hierarchisches Partitionieren von Trainingsdaten mit Hilfe von Entscheidungen. Die Entscheidung an einem Knoten des Decision Trees wird als *Split-Kriterium* bezeichnet und ist üblicherweise eine Bedingung über ein oder mehrere Input Variablen des Trainings Datensatzes, mit dessen Hilfe der Decision Tree erstellt wird. Das Split-Kriterium teilt den Datensatz in zwei oder mehr Teile. Beispiel: Wenn *Alter* eine Input Variable ist und das Split-Kriterium $Alter \leq 30$, dann enthält der linke Zweig des aktuellen Knotens alle Trainingsdaten für die das Split-Kriterium erfüllt ist, also alle Trainingsdaten in denen die Variable *Alter* den Wert 30 nicht überschreitet. Alle anderen Daten sind im rechten Zweig vertreten. Die letzten Knoten werden als *Blätter* bezeichnet und bestimmen den eigentlichen Wert der Zielvariable. Der Wert ist der dominante Wert von den verbleibenden Trainingsdaten. Das Erstellen von Knoten, basierend auf Splitkriterien, kann mit *Stopp-Kriterien* unterbrochen werden um einen zu großen Decision Tree zu vermeiden. Eine weitere Möglichkeit ist das Beschneiden bzw., *pruning* eines Decision Trees um ein Overfitting zu verhindern. Overfitting bezeichnet im allgemeinen das Anlernen von rauschenden bzw. ausreisenden und fehlerhaften Daten. Dies gilt zu vermeiden. [1]

Mithilfe des Trainings Datensatzes wird die Erstellung des Decision Trees, bzw. das Split-Kriterium, *überwacht*. Somit gehören Decision Trees im Data Mining Bereich zu den *Klassifikationsmethoden*. [1]

Pseudocode

Der nachfolgende Pseudocode zeigt einen generischen Trainings Algorithmus für das Erstellen eines Decision Trees: [1]

Algorithm GenericDecisionTree(Data Set: D)

begin

 Create root node containing D ;

repeat

 Select an eligible node in the tree;

 Split the selected node into two or more nodes

 based on a pre-defined split criterion;

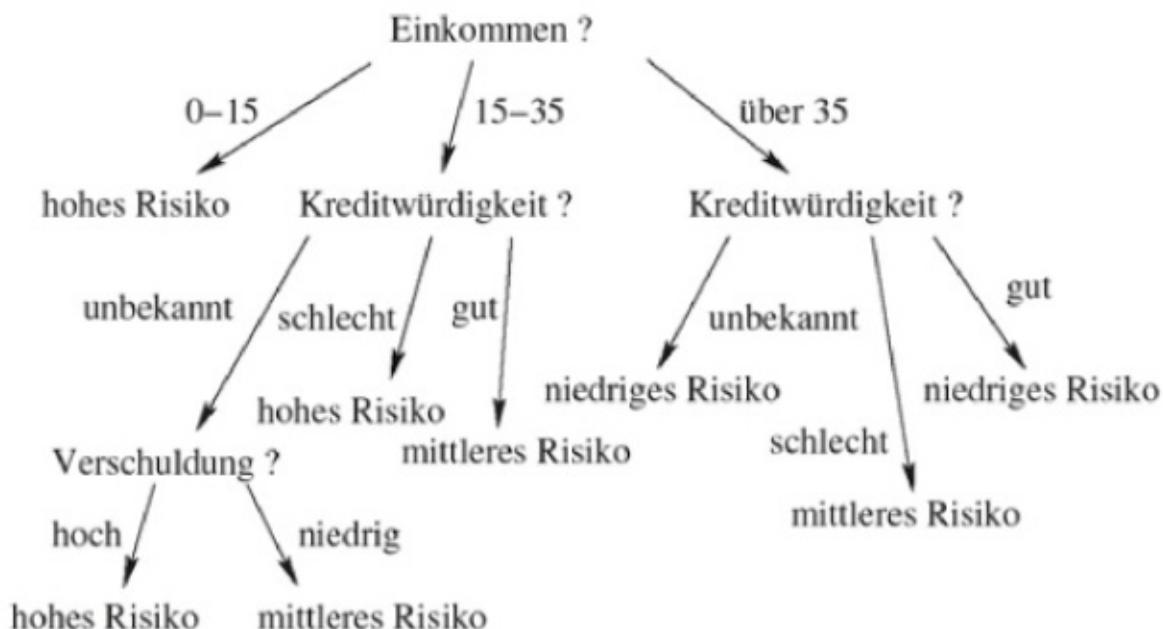
until no more eligible nodes for split;

 Prune overfitting nodes from tree;

 Label each leaf node with its dominant class;

end

Nachdem der Decision Tree erstellt wurde, wird dieser benutzt um eine Zielvariable von einer Input Datenmenge zu bestimmen. Der Decision Tree wird dabei von oben nach unten traversiert bis ein Blattknoten erreicht wird. Die nachfolgende Abbildung illustriert beispielhaft einen Decision Tree.



Decision-Tree für ein Kredit Beispiel. [2]

Splitkriterien

Ein Split-Kriterium sind Bedingungen über ein oder mehrere Input Variablen einer Instanz des Trainings Datensatzes. Splitkriterien werden verwendet um einen Trainingsdatensatz zu partitionieren. Das Ziel ist es, Splitkriterien zu finden, welche den Unterschied zwischen den partitionierten Teilen maximiert. Nachfolgend werden drei typische Splitkriterien näher erläutert. [1]

1. *Fehlerrate*: Wenn p die Menge der einzelnen Instanzen (also einzelne Datenreihen in einem Datensatz) eines Datensatzes ist, welche zum dominierenden Wert der Zielvariable gehören, dann ist die Fehlerrate $1 - p$. Die gesamte Fehlerrate einer Partitionierung bzw. Split eines Trainings Datensatzes mit einem Split-Kriterium errechnet sich mit dem gewichteten Durchschnitt der einzelnen Fehlerraten der einzelnen Partitionen. Das Gewicht einer Fehlerrate ist der Anteil der einzelnen Instanzen zum gesamten Datensatz. Es wird das Split-Kriterium mit der geringsten Fehlerrate gewählt. [1]
1. *Gini Index*: Der Gini Index $G(S)$ von einer Datenmenge S berechnet sich mit der nachfolgenden Gleichung mit der Verteilung der Werte der Zielvariablen $p_1 \dots p_k$ des Trainings Datensatzes. Der gesamte Gini Index für eine Partitionieren bzw. Split errechnet sich mit dem gewichteten Durchschnitt der einzelnen Gini Index Werte. Das Split-Kriterium mit dem geringsten Gini-Index wird gewählt. [1]

$$G(S) = 1 - \sum_{j=1}^k p_j^2$$

1. *Entropie*: Die Entropie $E(S)$ von einer Datenmenge S berechnet sich mit der nachfolgenden Gleichung. Die Verteilung der Werte von der Zielvariablen zum Trainingsdatensatz wird als $p_1 \dots p_k$ bezeichnet. Die gesamte Entropie für eine Partitionieren bzw. Split errechnet sich mit dem gewichteten Durchschnitt der einzelnen Entropie Werte. Das Split-Kriterium mit dem geringsten Entropie Wert wird gewählt. [1]

$$E(S) = - \sum_{j=1}^k p_j \log_2(p_j)$$

Stopp-Kriterien und Pruning

Wenn ein Decision Tree bis zum allerletzten Blatt erstellt wird bis alle Blätter jeweils nur noch Datenmengen eines Wertes der Zielvariablen enthalten, dann weist der Decision Tree eine Genauigkeit von 100% in Bezug auf den Trainingsdatensatz auf. Leider hat sich gezeigt, dass ein solcher Decision Tree sich schlecht auf neue, unvorhergesehene Daten verhält. Im Allgemeinen sind einfachere und weniger komplexe Decision Trees zu bevorzugen um ein Overfitting zu vermeiden. [3]

Mit einem Stopp-Kriterium (z. B. eine festgelegte Fehlerrate) lässt sich das wachsen des Decision Trees aufhalten. Leider gibt es keine Möglichkeit vorherzusagen an welchem Punkt der Decision Tree aufhören soll zu wachsen. Daher ist eine gängige Strategie bestimmte Knoten Teile zu entfernen oder zu konvertieren (Pruning). Eine Pruning-Strategie ist das

Verwenden einer Kostenfunktion, welche beispielsweise die Fehlerrate sowie die Komplexität (Anzahl der Knoten und Informations-theoretische Prinzipien) des Decision Trees berücksichtigen. [1]

1. Aggarwal, C.: Data Mining: The textbook. Springer (Herausgeber) (2015)
 2. Cleve, J., Lämmel, U.: Data Mining. Walter de Gruyter GmbH (Herausgeber) (2016)
 3. Witten, I., Frank, E., Hall, M.: Data Mining. Morgan Kaufmann (Herausgeber) (2011)
-

Author: Yannick Kloss

Implementierung

Das Projekt verwendet für die Implementierung der Klassifizierungs Algorithmen die *Scikit-learn Library*^[1], welche eine wohl bekannte und beliebte Python Library für Data Mining Algorithmen ist. Scikit-learn wird von Google unterstützt und befindet sich unter aktiver Entwicklung.

Decision Tree Beispiel mit Scikit-learn

Für das kommende Beispiel werden die folgenden, beispielhaften, Trainingsdaten verwendet. Die einzelnen Zeilen stehen jeweils für ein Fall bzw. *Sample*. Es existieren zwei Features. Jedes Sample hat für jedes Feature einen Wert und ist einer Klasse zugewiesen.

Samples	Feature 1	Feature 2	Klasse
Sample 1	0	0	0
Sample 2	1	1	1

Der nachfolgende Code-Abschnitt implementiert einen Decision Tree mit dem Gini-Index als Splitkriterium. Der Code-Abschnitt zeigt das Trainieren des Decision Trees mit den bereits aufgeführten Trainingsdaten. Variable x ist die *samples x features* Matrix und y ist ein Array mit den zugewiesenen Klassen. Beispielsweise ist y[0] = 0 die Klasse des Samples x[0] mit den Features x[0][0] = 0 und x[0][1] = 0.

```
from sklearn import tree
x = [[0, 0],
     [1, 1]]
y = [0, 1]
clf = tree.DecisionTreeClassifier(criterion='gini')
clf = clf.fit(x, y)
```

Mit dem nächsten Code-Abschnitt wird für ein drittes Sample die Klasse mit dem trainierten Decision Tree vorhergesagt.

```
result = clf.predict([[1, 1]])
print(result)
-> Ausgabe: [1]
```

Decision Tree für Sentimentanalyse

Im Bezug auf das Projekt wird für die Variable x (*samples x features* Matrix) der Output des Kapitels *Datenvorverarbeitung* vorgestellten Verfahrens verwendet. Variable y (Klassenbezeichnungen) sind die Klassifizierungen der Trainingsdaten.

```
from sklearn import tree
vectorizer =
df = pd.read_csv("training_data.csv", usecols=["ItemID", "Sentiment",
"SentimentSource", "SentimentText"])
y = df.Sentiment
x = vectorizer.fit_transform(df.SentimentText)
clf = tree.DecisionTreeClassifier(criterion='gini')
clf.fit(x, y)
```

1. Scikit-learn Homepage. <http://scikit-learn.org/stable/> (2017)
-

Author: Yannick Kloss

Author: Sven Schirmer

Einleitung

Naive Bayes gehört zu den probabilistischen Klassifikatoren und beruht auf dem Satz von Bayes, welcher die Berechnung der bedingten Wahrscheinlichkeit beschreibt. Naive Bayes gehört zu der Gruppe der Supervised Learner (dt. Überwachtes Lernen). Hierbei wird das Modell anhand der gelabelten Daten trainiert und jeder neuen Instanz eine Klasse zugewiesen.

Author: Sven Schirmer

Mathematische Grundlagen

Grundlage des Naive Bayes Klassifikatoren bildet die Wahrscheinlichkeitstheorie. Folgend werden die allgemeinen Grundlagen beschrieben. Beim einmaligen Werfen eines fairen Würfels sind die möglichen Elementarereignisse des Experiments:

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

Für jedes Element ω aus der Menge Ω gilt die gleiche Wahrscheinlichkeit.

$$\forall \omega \in \Omega : P(\omega) = \frac{1}{6}$$

Demnach ist die Summe der Wahrscheinlichkeiten der Elementarereignisse 1.

(Streng mathematisch genommen ist diese Definition für unendlich große Ω nicht korrekt, aber für den Bayes Klassifikator und für dieses Beispiel ist dies irrelevant, da der Wahrscheinlichkeitsraum diskret ist)

$$\sum_{\omega \in \Omega} P(\omega) = 1$$

$$P(\omega) = \frac{1}{|\Omega|}$$

Die Wahrscheinlichkeit eine 6 zu Würfeln ist zufolge 1/6.

Eine Teilmenge A aus Ω wird als Ereignis bezeichnet. Die Wahrscheinlichkeit des Ereignisses A ist die Summe der Wahrscheinlichkeiten aller Elementarereignisse aus A

$$P(A) = \sum_{\omega \in A} P(\omega)$$

Das Ereignis A könnte z.B. das Würfeln einer geraden Zahl sein.

$A = \{2, 4, 6\}$ Die Wahrscheinlichkeit für das Eintreffen von A beträgt 50%.

$$P(A) = P(\{2\}) + P(\{4\}) + P(\{6\}) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{1}{2}$$

Nun erweitern wir unser Experiment auf das zweifache Werfen eines fairen Würfel.

$$\Omega = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (2, 1), \dots, (6, 6)\}$$

$$\forall \omega \in \Omega : P(\omega) = \frac{1}{36}$$

Wir definieren das Ereignis A neu: die summierte Augenzahl beider Würfel ist 10.

$$A = \{(4, 6), (5, 5), (6, 4)\}$$

$$P(A) = P(\{(4, 6)\}) + P(\{(5, 5)\}) + P(\{(6, 4)\}) = \frac{1}{36} + \frac{1}{36} + \frac{1}{36} = \frac{3}{36}$$

Das Ergebnis B bedeutet, dass der erste Wurf eine 6 ist.

$$B = \{(6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)\}$$

$$P(B) = P(\{(6, 1)\}) + P(\{(6, 2)\}) + P(\{(6, 3)\}) + P(\{(6, 4)\}) + P(\{(6, 5)\}) + P(\{(6, 6)\}) = 6 \cdot \frac{1}{36} = \frac{6}{36}$$

Wenn man die Ereignisse A und B miteinander verknüpft, dann muss der erste Wurf eine 6 sein und die Augenzahl muss 10 betragen.

$$A \cap B = (6, 4)$$

$$P(A \cap B) = P(\{(6, 4)\}) = \frac{1}{36}$$

Bedingte Wahrscheinlichkeit

Die bedingte Wahrscheinlichkeit $P(A|B)$ beschreibt, die Wahrscheinlichkeit, dass das Ergebnis A eintritt, unter der Bedingung, dass das Ergebnis B bereits eingetreten ist. In unserem Beispiel bedeutet dies folgendes. Unter der Bedingung, dass die erste Zahl eine 6 ist (Ergebnis B), wie groß ist dann die Wahrscheinlichkeit, dass die Augensumme 10 beträgt (Ergebnis A)?

B ist gegeben:

$$B = \{(6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)\}$$

In der Menge B gibt es ein Element, welches auch A entspricht.

$A \cap B = (6, 4)$ Somit beträgt die Wahrscheinlichkeit von A innerhalb B:

$$P(A|B) = \frac{P(\{(6, 4)\})}{P(\{(6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6)\})}$$

$$P(A|B) = \frac{\frac{1}{36}}{\frac{1}{6}} = \frac{1}{6}$$

Da für $P(A|B)$ gilt:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

auch bekannt als Bayesesches Theorem.

Komplette Definition des Theorems:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{\frac{P(A \cap B)}{P(A)} \cdot P(A)}{P(B)} = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Author: Sven Schirmer

Bayesesches Theorem als Klassifikator

Dieses Theorem ist als Klassifikator anwendbar und bestimmt die Zugehörigkeit einer Instanz zu einer Klasse anhand der errechneten bedingten Wahrscheinlichkeiten.

So können wir den folgenden Term

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

umrechnen, um die Wahrscheinlichkeit für eine Klassenzugehörigkeit zu errechnen.

$$P(K|I) = \frac{P(I_1|K) \cdot P(I_2|K) \cdot \dots \cdot P(I_n|K) \cdot P(K)}{P(I)}$$

Wobei

I die Instanz,

In die einzelnen Attribute der Instanz und

K das Ereignis bzw. die Klasse der Instanz beschreibt.

Hierbei ist davon auszugehen, dass alle Attribute unabhängig voneinander sind, sodass wir die Attribute einfach aufteilen können. Deswegen wird dieser Bayes Klassifikator auch als naiv oder simpel bezeichnet.

Author: Sven Schirmer

Rechenbeispiel

Als Datengrundlage dient das Dataset auf github: <https://github.com/gr8Adakron/naive-bayes-using-python/blob/master/tennis.csv>

Die Tabelle enthält die normalen Attribute *Outlook*, *Temperatur*, *Humidity* und *Windy*. *Play* ist das Label Attribut, nach dem es zu klassifizieren gilt.

Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Der folgende Datensatz soll klassifiziert werden.

Outlook	Temp.	Humidity	Windy	Play
sunny	hot	high	false	?

Wir suchen also die Wahrscheinlichkeiten für die Klassen *yes* und *no* unter der Bedingung der Instanzen.

Für die erste Instanz sieht dies so aus:

$$P(\text{yes}|I) = \frac{P(I_1 = \text{sunny}|\text{yes}) \cdot P(I_2 = \text{hot}|\text{yes}) \cdot P(I_3 = \text{high}|\text{yes}) \cdot P(I_4 = \text{true}|\text{yes}) \cdot P(\text{yes})}{P(I)}$$

$$P(\text{no}|I) = \frac{P(I_1 = \text{sunny}|\text{no}) \cdot P(I_2 = \text{hot}|\text{no}) \cdot P(I_3 = \text{high}|\text{no}) \cdot P(I_4 = \text{true}|\text{no}) \cdot P(\text{no})}{P(I)}$$

Fangen wir mit der Klasse yes und folgendem Term an.

$$P(I_1 = \text{sunny}|\text{yes})$$

Unter der Bedingung der Klasse yes (Anzahl 9) zählen wir 2 Instanzen mit *sunny*. Somit beträgt die bedingte Wahrscheinlichkeit für den Term:

$$P(I_1 = \text{sunny}|\text{yes}) = \frac{2}{9}$$

Wir verfahren analog mit den anderen Attributen:

$$P(I_2 = \text{hot}|\text{yes}) = \frac{2}{9}$$

$$P(I_3 = \text{high}|\text{yes}) = \frac{3}{9}$$

$$P(I_4 = \text{false}|\text{yes}) = \frac{6}{9}$$

$$P(\text{yes}) = \frac{9}{14}$$

Nun können wir bereits den Likelihood für die Klasse yes ausrechnen bzw. fast den kompletten Term außer $P(I)$.

$$\text{Likelihood}(\text{yes}) = \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{3}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} = \frac{4}{567} = 0,0071$$

Wir verfahren analog zu der Klasse no,

$$P(I_1 = \text{sunny}|\text{no}) = \frac{3}{5}$$

sowie mit den anderen Attributen und erhalten die Formel für den Likelihood.

$$\text{Likelihood}(\text{no}) = \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} \cdot \frac{5}{14} = \frac{36}{875} = 0,0411$$

Der Likelihood für die Klasse no ist höher und die Instanz wird somit als no klassifiziert. Der Term $P(I)$, die Wahrscheinlichkeit für die Instanz, ist uns unbekannt. Dennoch können wir die Wahrscheinlichkeit für die Klasse anhand der bereits errechneten Likelihoods bestimmen.

$$P(\text{yes}|I) = \frac{\text{Likelihood}(\text{yes})}{\text{Likelihood}(\text{yes}) + \text{Likelihood}(\text{no})}$$

$$P(\text{yes}|I) = \frac{0,0071}{0,0071 + 0,0411} = 0,1473$$

$$P(\text{no}|I) = \frac{\text{Likelihood}(\text{no})}{\text{Likelihood}(\text{yes}) + \text{Likelihood}(\text{no})}$$

$$P(\text{no}|I) = \frac{0,0411}{0,0071 + 0,0411} = \mathbf{0,8527}$$

Zu 85% trifft die Klassifizierung zu *no* zu.

Author: Sven Schirmer

Anwendungsgebiete

Der Naive Bayes Algorithmus zeichnet sich durch seine Einfachheit aus. Die Anwendung des Klassifizierers empfiehlt sich, wenn die Attribute eine sehr hohe Unabhängigkeit voneinander aufweisen, da die Attribute nicht gewichtet werden. (vgl. Rish 2014) In der Praxis ist dies oft der Fall und der Naive Bayes Klassifikator liefert eine sehr gute Effizienz bzw. Performance. (vgl. Zhang) Im Speziellen ist Naive Bayes hervorragend für die Textklassifizierung einsetzbar, welches die Praxis wiederspiegelt. (vgl. Felden 2006, S. 27) Die Email-Spam Erkennung wird in der Regel durch Bayes Algorithmen gelöst. (vgl. Metsis) Ferner kann der Algorithmus angepasst, erweitert, sowie optimiert werden, sodass die Fehlerrate bei der Textklassifizierung weiter gesenkt werden kann. (vgl. Rennie)

Author: Sven Schirmer

Referenzen

- Irina Rish: An empirical study of the naive Bayes classifier; T.J. Watson Research Center; 2014;
https://www.researchgate.net/profile/Irina_Rish/publication/228845263_An_Empirical_Study_of_the_Naive_Bayes_Classifier/links/00b7d52dc3cc8d692000000/An-Empirical-Study-of-the-Naive-Bayes-Classifier.pdf Aufgerufen am 01.07.2017
 - Harry Zhang: The Optimality of Naive Bayes; Faculty of Computer Science - University of New Brunswick; <http://www.aaai.org/Papers/FLAIRS/2004/Flairs04-097.pdf>
Aufgerufen am 01.07.2017.
 - Carsten Felden, Heiko Bock, André Gräning, Lana Molotowa, Jan Saat, Rebecca Schäfer, Bernhard Schneider, Jenny Steinborn, Jochen Voecks, Christopher Woerle: Evaluation von Algorithmen zur Textklassifikation; 2006; http://tu-freiberg.de/sites/default/files/media/fakultaet-6-3307/fileadmin/Arbeitspapiere/2006/felden_10_2006.pdf Aufgerufen am 01.07.2017.
 - Vangelis Metsis, Ion Androutsopoulos, Georgios Palioras: Spam Filtering with Naive Bayes – Which Naive Bayes?;
https://classes.soe.ucsc.edu/cmps290c/Spring12/lect/14/CEAS2006_corrected-naiveBayesSpam.pdf Aufgerufen am 01.07.2017.
 - Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, David R. Karger: Tackling the Poor Assumptions of Naive Bayes Text Classifiers;
<http://www.aaai.org/Papers/ICML/2003/ICML03-081.pdf> Aufgerufen am 01.07.2017.
-

Author: Sven Schirmer

Support Vector Machine

Die Support Vector Machine (SVM) gehört zur Familie der Klassifikationsalgorithmen.

Klassifikation bedeutet, eine Menge von Daten einer bestimmten Klasse bzw. Kategorie zuzuordnen. Gegeben sei also eine Menge an Daten \square , wie beispielsweise \square und eine Menge an Klassen \square , wie z.B. \square , wobei die Klassen i.d.R. von Menschen definiert werden. Die Trainingsdaten sind Zuordnungen von Daten zu Klassen, z.B.: \square mit \square . Algorithmen aus diesem Bereich können eine Klassifikator \square erlernen, der alle Daten zu einer Klasse zuordnen kann: \square [1]

Die Anfänge der Support Vector Machine liegen bereits einige Jahre zurück. Bereits im Jahr 1963 stellten Vladimir N. Vapnik und Alexey Ya. Chervonenkis den SVM-Algorithmus vor. [2] Die nächste größere Erfindung war der sogenannte *Kernel Trick*. Dieser wurde 1992 von Boser et. al. vorgestellt. [3] Mit Hilfe des Kernel Tricks ist es möglich die Support Vector Machine auch auf nicht linear separierbare Daten anzuwenden. In den folgenden Jahren wurden nur noch wenige geringe Verbesserungen und Varianten veröffentlicht. Auch heutzutage wird dieser Algorithmus noch oft genutzt und dient ihn vielen wissenschaftlichen Arbeiten als Vergleich zu modernen Ansätzen. [4]

Der weitere Teil dieser Arbeit ist wie folgt gegliedert:

- [Algorithmus im Detail](#)
 - [Ebenendarstellung](#)
 - [Bedingungen](#)
 - [Abstandsberechnung](#)
 - [Optimierungsproblem](#)
 - [Kernel Trick](#)
- [Implementierungen](#)

Im Kapitel [Algorithmus im Detail](#) wird die Support Vector Machine mathematisch beschrieben und hergeleitet. Das Kapitel [Implementierungen](#) enthält Verweise und Beispiele zu verschiedenen Implementierungen des Algorithmus.

1. Manning, C. D., Raghavan, P., & Schütze, H. (2009). An Introduction to Information Retrieval. Online, (c), 569. <https://doi.org/10.1109/LPT.2009.2020494>, S. 256 ↵

2. https://en.wikipedia.org/wiki/Support_vector_machine ↵

3. Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 144–152. <https://doi.org/10.1.1.21.3818> ↵

4. Lu, Y. (2016). Food Image Recognition by Using Convolutional Neural Networks (CNNs). Retrieved from <http://arxiv.org/abs/1612.00983> ↵

Author: Daniel Beneker

Algorithmus im Detail

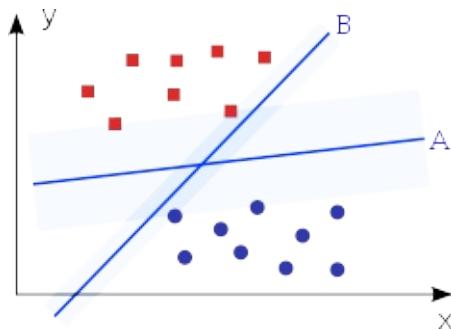


Abbildung 1: Zwei mögliche Hyperebenen^[1]

Eine Support Vector Machine trennt eine Menge von Objekten mit einer Hyperebene in zwei Klassen, sodass der Abstand zwischen den Objekten der Klassen und der Hyperebene maximal wird. Sie wird daher auch als *Large Margin Classifier* bezeichnet. Abbildung 1 zeigt eine Menge von Objekten, die zu zwei verschiedenen Klassen gehören, sowie zwei mögliche Hyperebenen A und B. Beide teilen die Objekte in zwei Klassen auf, doch der Abstand bei Trennebene A ist wesentlich größer als bei Trennebene B. Die Trennebene A, mit dem maximalen Abstand, stellt damit einen besseren Klassifikator dar, weil sie den kleinsten Generalisierungsfehler erzeugt.^[2]

Ebenendarstellung

Gegeben sei ein zweidimensionaler Raum, dann lässt sich die Hyperebene als lineare Gleichung $y = a^*x + b$ darstellen. Sei Vektor $\begin{pmatrix} \square \\ \square \end{pmatrix}$ und $\begin{pmatrix} \square \\ \square \end{pmatrix}$, dann lässt sich die lineare Gleichung darstellen als $\begin{pmatrix} \square \end{pmatrix}$.^[3]

Bedingungen

Gesucht werden zwei Hyperebenen $\begin{pmatrix} \square \end{pmatrix}$ und $\begin{pmatrix} \square \end{pmatrix}$, die eine Menge von Datenpunkten in zwei verschiedene Klassen trennen. Zwischen diesen Hyperebenen sollen sich keine Datenpunkte befinden. Für jeden Datenpunkt x_i muss also gelten:

Formel 1: $\begin{pmatrix} \square \end{pmatrix}$

oder

Formel 2: $\begin{pmatrix} \square \end{pmatrix}$

Sei y_i ist die zugehörige Klasse zum Datenpunkt , dann lassen sich die Bedingungen aus Formal 1 und 2 zu einer einzigen Bedingung zusammenfassen, indem sie mit y_i multipliziert werden.

Formel 3: $\boxed{w^T x_i + b \geq 1}$ für $y_i = 1$ und $\boxed{w^T x_i + b \leq -1}$ für $y_i = -1$

In Abbildung 2 markieren die gestrichelten Linien zwei mögliche Hyperebenen, sodass die Bedingung aus Formel 3 erfüllt ist.^[4]

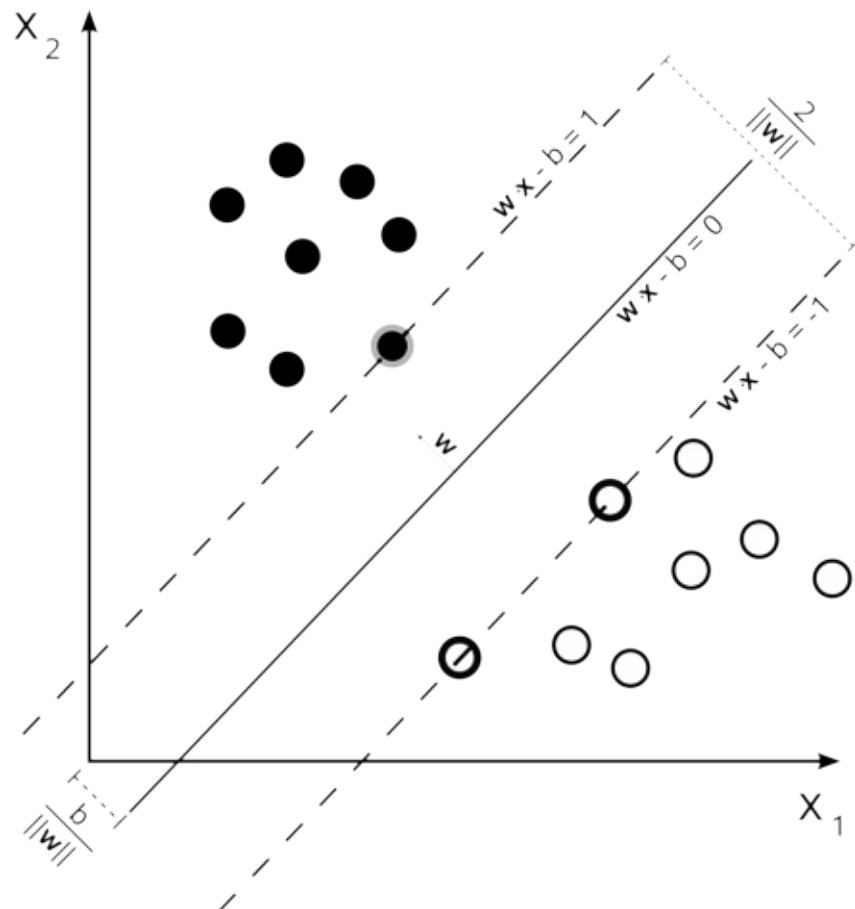


Abbildung 2: Support Vector Machine - Hyperebene mit maximalem Abstand^[1]

Abstandsberechnung

Ziel der Support Vector Machine ist es den Abstand m zwischen den Hyperebenen $\boxed{w^T x + b = 1}$ und $\boxed{w^T x + b = -1}$ zu maximieren.

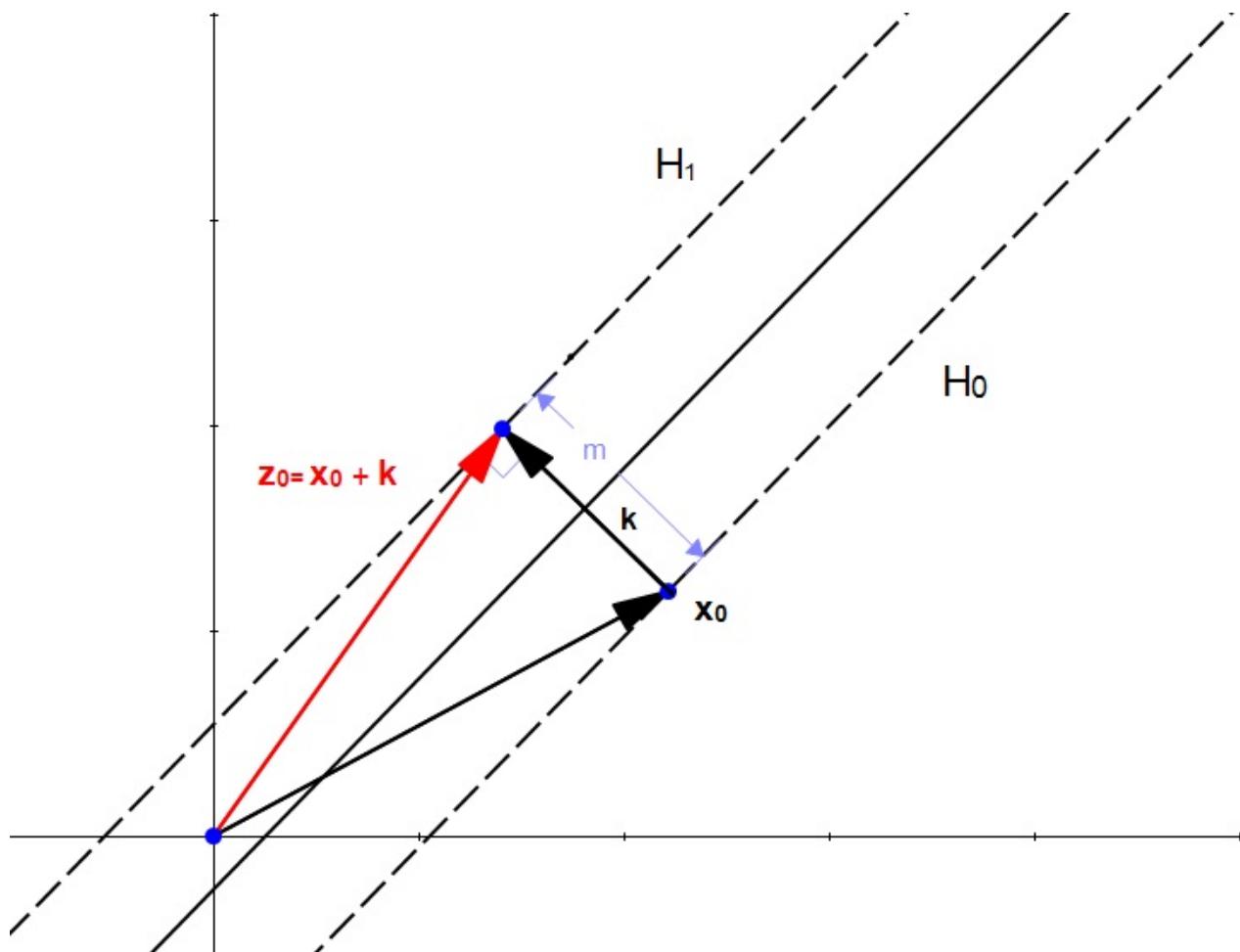


Abbildung 3: Abstandsberechnung [4]

Der Abstand m lässt sich einfach als Vektor \square darstellen mit:

Formel 4: \square

Addiert man zu einem Punkt x_0 in der Hyperebene H_0 den Vektor \square , so erhält man den Punkt z_0 in der Hyperebene H_1 . Setzt man Formel 4 in H_1 ein, so erhält man die Formel 5 um den Abstand zu berechnen. [4]

Formel 5: \square

Die Herleitung der Formel 5 ist [hier](#) aufgelistet. [4]

Optimierungsproblem

Den maximalen Abstand m erhält man demnach, indem \square minimiert wird. Es wird also eine Hyperebene mit dem kleinsten \square gesucht, die die Bedingung aus Formel 3 erfüllt.

Eine Möglichkeit dieses Optimierungsproblems zu lösen ist mittels Gradientenabstieg. Genutzt wurde dazu die Implementierung des *Pegasos*^[5] Algorithmus aus dem *sklearn*-Framework^[6]. Der Algorithmus basiert auf der Fehlerfunktion aus Formel 6. \square bezieht sich

hierbei auf die *hinge loss* Funktion (siehe Formel 7).

Formel 6: $\frac{1}{2} \sum_{i=1}^n \ell_i(\langle \vec{w}, \vec{x} \rangle + b)$

Formel 7: $\}$

Kernel Trick

Die Support Vector Machine, wie sie bisher besprochen wurde, kann nur Daten in zwei Klassen trennen, wenn sie linear separierbar sind. Das wird erreicht, indem eine Hyperebene konstruiert wird. Linear separierbar bedeutet, dass es sich immer um eine gerade Ebene handeln muss. Im zweidimensionalen Raum ist eine Hyperebene eine Gerade. Abbildung 4 stellt das Problem gut dar. Links ist eine Menge von Datenpunkten im zweidimensionalen Raum dargestellt, die zu zwei verschiedenen Klassen gehören. Diese Daten lassen sich beispielsweise mit einem Kreis in zwei Klassen trennen. Eine Support Vector Machine könnte diese Daten nicht klassifizieren, da sich die Daten nicht mit einer Gerade trennen lassen.^[7]

Doch mit dem sogenannten Kernel Trick ist dies trotzdem möglich. Die Idee dabei ist, dem Merkmalsraum eine weitere Dimension hinzuzufügen. Auf dem linken Bild befinden sich die Datenpunkte im zweidimensionalen Raum. Das rechte Bild zeigt die gleichen Datenpunkte im dreidimensionalen Raum. Gut zu erkennen ist, dass sich die Daten nach der Transformation linear separieren lassen. Dadurch lässt sich auch die Support Vector Machine auf dieses Problem anwenden.^[7]

Die Datenpunkte lassen sich auf verschiedene Weisen in eine höhere Dimension projizieren. In dem Beispiel aus Abbildung 4 wurden die drei neuen Dimensionen wie folgt berechnet:^[7]

Formel 8: $\boxed{\quad}$

Ein weiterer Vorteil des Kernel Tricks ist, dass die Berechnung sehr effizient ist. Die Berechnung der Hyperebene kann in einem höherdimensionalen Raum durchgeführt werden, ohne die einzelnen Datenpunkte in diesen Raum zu projizieren.^[7]

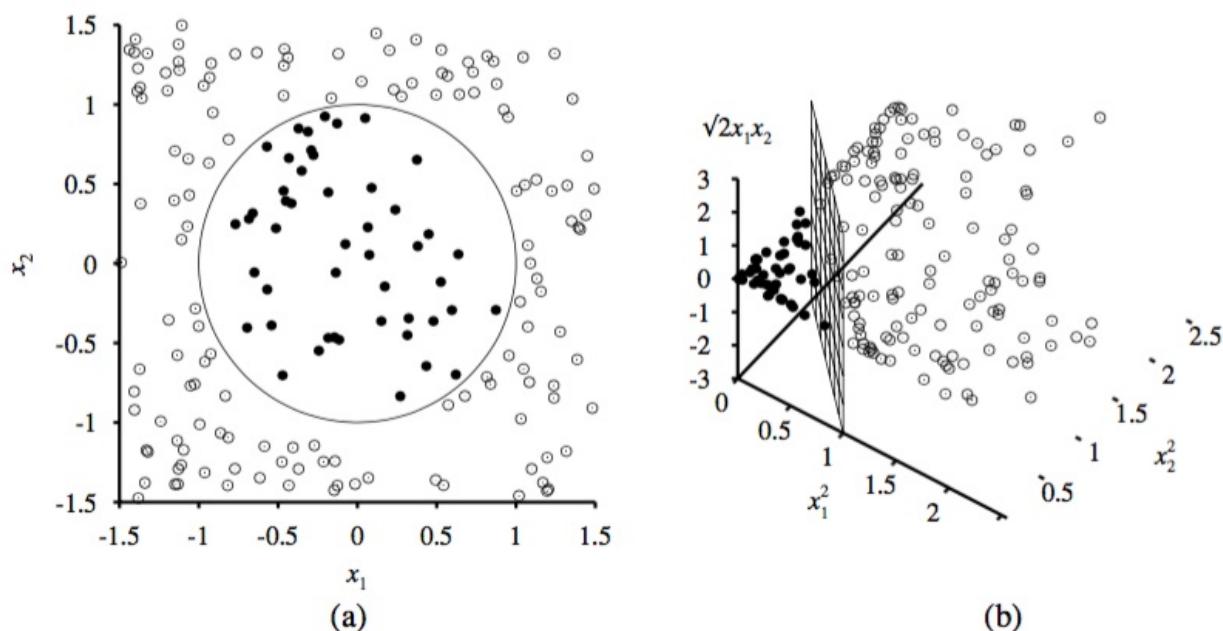


Abbildung 4: Kernel Trick grafisch dargestellt: (a) Daten im zweidimensionalen Raum, (b) Daten projiziert in den dreidimensionalen Raum^[7]

1. https://de.wikipedia.org/wiki/Support_Vector_Machine ↪
 2. Manning, C. D., Raghavan, P., & Schütze, H. Cambridge University Press; 2009. Introduction to Information Retrieval ↪
 3. <http://www.svm-tutorial.com/2014/11/svm-understanding-math-part-2/> ↪
 4. <http://www.svm-tutorial.com/2015/06/svm-understanding-math-part-3/> ↪ ↪ ↪ ↪
 5. Shalev-Shwartz, Shai, et al. "Pegasos: Primal estimated sub-gradient solver for svm." Mathematical programming 127.1 (2011): 3-30. ↪
 6. <http://scikit-learn.org/stable/modules/sgd.html> ↪
 7. Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach. Prentice Hall (Third edit). Pearson. ↪ ↪ ↪ ↪ ↪
-

Author: Daniel Beneker

Implementierungen

Der Algorithmus Support-Vector-Machine ist in vielen verschiedenen Sprachen verfügbar. Viele greifen intern auf die Bibliothek [libSVM](#) zurück und müssen dadurch nur ein Interface bereit stellen.

PHP

In php steht ein Interface zu [libSVM](#) zur Verfügung. Es existiert eine [Dokumentation](#), die die Installation und Nutzung erklärt. Dort wurde auch folgendes Beispiel entnommen:

```
<?php
$data = array(
    array(-1, 1 => 0.43, 3 => 0.12, 9284 => 0.2),
    array(1, 1 => 0.22, 5 => 0.01, 94 => 0.11),
);

$svm = new SVM();
$model = $svm->train($data);

$data = array(1 => 0.43, 3 => 0.12, 9284 => 0.2);
$result = $model->predict($data);
var_dump($result);
$model->save('model.svm');
?>
```

Python (sklearn)

Das bekannte Machine-Learning-Framework *sklearn* nutzt ebenfalls [libSVM](#). Das folgende Beispiel, sowie andere Varianten des SVM-Algorithmus sind zu finden unter der [sklearn SVM Dokumentation](#).

```
from sklearn import svm
X = [[0, 0], [1, 1]]
y = [0, 1]
clf = svm.SVC()
clf.fit(X, y)
clf.predict([[2., 2.]])
```

Java (weka)

In Java steht SVM über das *weka*-Framework zur Verfügung. Die Dokumentation des Algorithmus, der intern ebenfalls *libSVM* nutzt, ist [hier](#)) zu finden.

NodeJS

Für NodeJS lässt sich der Support-Vector-Machine Algorithmus über den Packet Manager *npm* installieren. Die Dokumentation des Paketes und das hier aufgeführte Beispiel ist unter [diesem Link](#) verfügbar. Die Implementierung nutzt intern ebenfalls libSVM.

```
var svm = require('node-svm');
var xor_data = [
  [[0, 0], 0],
  [[0, 1], 1],
  [[1, 0], 1],
  [[1, 1], 0]
];
var clf = new svm.CSVC();
clf.train(xor_data).done(function () {
  // predict things
  xor_data.forEach(function(ex){
    var prediction = clf.predictSync(ex[0]);
    console.log('%d XOR %d => %d', ex[0][0], ex[0][1], prediction);
  });
});
```

MatLab

MatLab ist bei vielen mathematischen Anwendungen ein beliebtes Programm. Besonders zum Visualisieren von Funktionen ist es hervorragend geeignet. Eine Implementierung des SVM-Algorithmus ist ebenfalls enthalten: [fitcsvm](#). Mit MatLab lassen sich nicht nur die Datenpunkte als Grafik plotten, sondern auch die Trennebenen, die der SVM-Algorithmus findet, visualisieren.

Author: Daniel Beneker

Embedded Computing

Teilbereich(e): Smart Home, IoT

Projektleiter: Gamze Söylev Öktem

Projektteam: Nils Kohlmeier, Wladimir Streck, Gamze Söylev Öktem, Jonas Wiese, Justin Jagieniak

Github-Repo: <https://github.com/nkohlmeier/Spezielle-Gebiete-zum-Softwareengineering.git>

Einführung

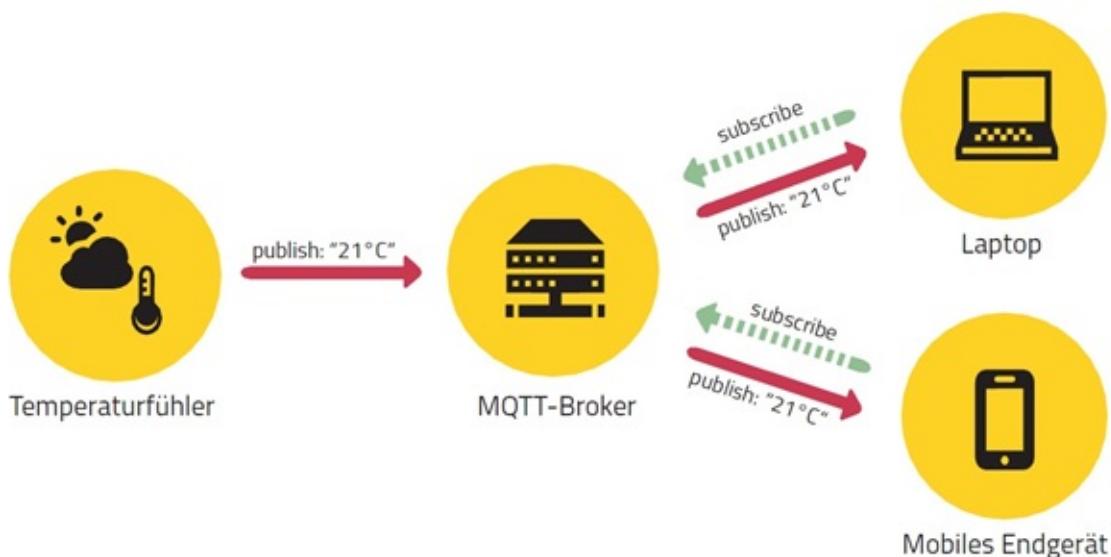
Mit dem Aufkommen des Trends des „Internet of Things“, also normalen Gegenständen wie Lampen oder Steckdosen aber auch aufwändigen Produktionsmaschinen, die über das Internet kommunizieren können, ist es nötig klare Standards für dessen Kommunikation zu definieren. Hierbei muss auf viele Faktoren geachtet werden, so sollten die verwendeten Protokolle zum Beispiel eine vollständig gesicherte Kommunikation gewährleisten können. Da einige dieser Geräte mit Batterien betrieben werden, kann ein aufwändiges Kommunikationsprotokoll erhebliche Abstriche in der Laufzeit des Gerätes bedeuten. Daher ist es ebenso wichtig, dass die Protokolle nicht zu hohe Ansprüche an die Geräte stellen.

Im nachfolgenden Kapitel werden fünf Protokolle vorgestellt, welche für diesen Einsatz geeignet sein könnten.

MQTT

Struktur & Nutzen

Das Message Queue Telemetry Transport Protokoll (MQTT) ist ein schlankes und leichtgewichtiges Kommunikationsprotokoll, dass auf dem Publish-Subscribe Prinzip basiert. Es wurde 1999 als Kommunikationsprotokoll zwischen Maschinen entwickelt. Besonders wichtig war den Entwicklern der geringe Protokoll-Overhead, sodass das Protokoll auch von Low-Energy Geräten ohne viel Prozessorzeit verarbeitet werden kann. Die „Organization for the Advancement of Structured Information Standards“ (OASIS) standardisiert seit 2013 MQTT als das Protokoll des Internets der Dinge [5](#). Zur Kommunikation zwischen zwei Clients ist ein Broker (Server) nötig. Beide Clients melden sich beim Broker an und teilen diesem mit, welche „Topics“ sie abonnieren möchten. Ein Topic wird mit einer Zeichenkette beschrieben und definiert den Kommunikationskanal. Der Broker sorgt dafür, dass die von Client A auf einem Topic gesendeten Daten, an alle Abonnenten dieses Topics zugestellt werden. Hierbei ist besonders zu beachten, dass auch viele Konsumenten und Produzenten das gleiche Topic verwenden können. MQTT eignet sich deshalb besonders als „Many-to-Many“ Kommunikationsprotokoll [6](#). Meistens wird zum Nachrichtenaustausch MQTT über TCP übertragen. Dies ist aber nicht vorgeschrieben, sodass auch andere Protokolle genutzt werden können [4](#). Das untenstehende Bild beschreibt den beschriebenen Ablauf mit einem Beispiel. Ein Temperaturfühler misst die Temperatur und sendet diesen Wert über ein Topic (z.B. „/temperatur“) an den Broker. Der Broker verteilt die Nachricht an alle Clients, die das Topic abonniert haben.

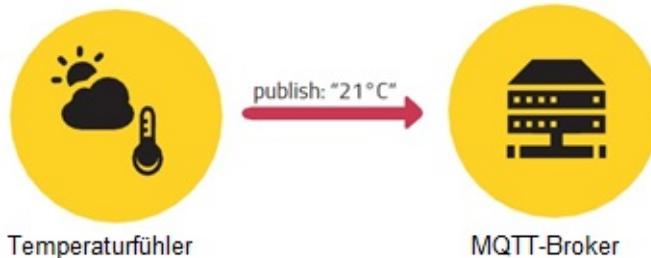


Die Konsumenten und Produzenten sind aber nicht fest an ihre Rollen gebunden, sondern können auch die jeweils andere Rolle übernehmen.

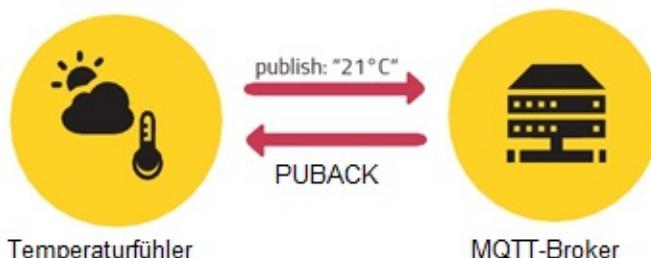
QoS

Da MQTT nicht zwangsläufig über TCP übertragen werden muss, das unterliegende Protokoll also nicht ein Erhalt einer Nachricht garantieren muss, definiert MQTT drei verschiedene Quality of Service (QoS) Klassen.

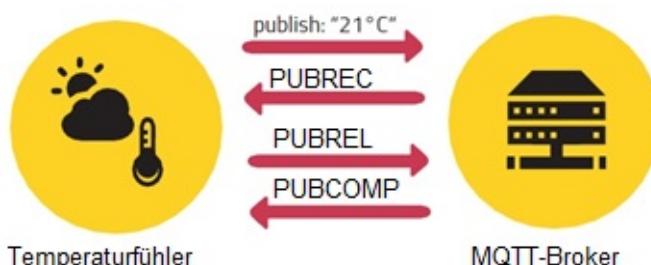
QoS Level 0 ist die niedrigste Qualitätsstufe. Bei dieser wird nicht garantiert, dass die Nachricht überhaupt irgendwo ankommt – „fire'n'forget“ [2](#).



QoS Level 1 garantiert, dass die Nachricht mindestens einmal beim Ziel ankommt. Der Broker muss das Erhalten der Nachricht einmal bestätigen. Bleibt die Bestätigung aus, oder kommt erst nach Ablauf eines Timeouts beim Client an, sendet er die Nachricht erneut. Deshalb kann es bei diesem Level zu mehrfach Sendungen kommen [2](#). ankommt – „fire'n'forget“ [2](#).



QoS Level 2 garantiert, dass die Nachricht genau einmal beim Ziel ankommt. Hierzu müssen vier Nachrichten zwischen Client und Broker ausgetauscht werden. Der Broker bestätigt das erhalten der Publish-Nachricht mit einer „PUBREC“ („Publish received“) Paket. Zu diesem Zeitpunkt ist dem Client klar, dass seine Nachricht erfolgreich beim Broker angekommen ist. Das bestätigt er wiederum mit einer „PUBREL“ („Publish release“) Paket, was der Broker wiederrum mit dem „PUBCOMP“ („Publish complete“) Paket bestätigt [2](#). ankommt – „fire'n'forget“ [2](#).



Diese QoS Klasse muss an zwei Stellen definiert werden. Zum einen muss der Sender jeder seiner Nachrichten ein QoS Level mit geben um zu definieren mit welcher Garantie die Nachricht beim Broker ankommt, zum anderen muss der Abonnent eines Topics definieren mit welcher Wahrscheinlichkeit die Nachrichten bei ihm ankommen sollen. Das gewählte QoS Level hängt von vielen Faktoren ab, die Wichtigkeit der Nachricht, die Zuverlässigkeit des Transportwegs oder die Kosten einer Nachricht können in die Entscheidung einfließen [2](#).

Last Will

Ein Client kann beim Broker festlegen, dass eine bestimmte Nachricht an alle Interessierten geschickt wird, sobald der Broker merkt, dass der entsprechende Client die Verbindung verloren hat. Hiermit kann der Client indirekt allen Abonnenten mitteilen, dass er nicht mehr erreichbar ist und auch keine Nachrichten mehr Senden kann. Auch bei LastWill Nachrichten kann ein QoS festgelegt werden [3](#).

Retained Messages

Ein Produzent kann pro Topic eine Retained Message festlegen, diese wird beim Broker gespeichert. Wenn sich nun ein Konsument dieses Topic abonniert, bekommt er sofort diese Nachricht. Der neue Abonnent bekommt also sofort eine Nachricht, ohne erst auf eine neue Nachricht warten zu müssen. Setzt nun der Producer jede seiner Nachrichten als Retained, bekommt jeder neue Abonnent sofort die letzte Nachricht [1](#).

Sicherheit

MQTT verfügt über einen einfachen Login Mechanismus. Ist dieser aktiviert, muss jeder Client, egal ob Producer oder Consumer einen Username und ein Passwort übermitteln. Zusätzlich kann im Broker konfiguriert werden, welcher User welche Rechte auf welches Topic hat. Ohne weiteren Schutz werden die Logininformationen allerdings im Klartext übertragen. Um die übertragenden Daten zu schützen, können TLS und SSL genutzt werden. Diese Verfahren garantieren eine Verschlüsselung der Daten und die hinterlegten Zertifikate garantieren sowohl Broker als auch Client seine Identität. Zusätzlich können die eigentlichen „Userdaten“ von MQTT, also z.B. die gemessenen Werte noch weiter verschlüsselt werden. Sowohl TLS / SSL als auch allgemeine Verschlüsselung führen aber zu Overhead, sodass die übertragenden Pakete größer werden und mehr Zeit zum verarbeiten benötigt wird [7](#) [8](#).

IoT Verwendung

Das Protokoll eignet sich sehr gut für den Einsatz im IoT. Grade die äußerst einfache Verwendung und der kleine Protokoll Overhead fallen positiv auf.

Rest

Protokollübersicht

Der Begriff des Representational State Transfer (REST) wurde von Roy T. Fielding in seiner Dissertation (2000) eingeführt [9](#). Gemeint ist damit ein Programmierparadigma für verteilte Systeme. Darin wird die Struktur und das Verhalten des World Wide Webs abstrahiert. Mit Hilfe dieser Abstraktion ist es möglich einen Architekturstil zu schaffen, der die Anforderungen des modernen Webs besser darstellt. Strukturell basiert REST auf einer klassischen Client Server Architektur – der Server stellt einen Dienst bereit, der bei Bedarf vom Client aufgerufen werden kann. Die Ressourcen, die der Server bereitstellt, werden vom Client mittels einer URI (Uniform Resource Identifier) adressiert und abgerufen. Das zugrundeliegende Protokoll ist [http 10](#).

REST nutzt vier Operationen des http Protokolls um einem Client Zugriff auf die Ressourcen des Servers zu bieten:

Funktionalität	http-Verb	URI
Ressource anlegen	POST	http://example.org/picture
Ressource aktualisieren	PUT	http://example.org/picture/123
Ressource abfragen	GET	http://example.org/picture/123
Ressource löschen	DELETE	http://example.org/picture/123

Einer der wichtigsten Eigenschaften des REST Protokolls ist die Zustandslosigkeit jeder Interaktion zwischen Client und Server. So speichert der Server keinerlei Zustandsinformationen zu einem Client, dieses hat zur Folge, dass bei jeder Interaktion alle nötigen Informationen zu der Anfrage übergeben werden müssen. Die Anfrage ist also in sich geschlossen. Diese Eigenschaft sorgt dafür, dass ein REST Service gut skalierbar ist, sodass Anfragen bei viel Last auf andere Server verteilt werden können [11](#).

Sicherheit

Ein normaler REST Service ohne weitere Absicherungen kann verwendet werden, wenn keine Authentifikation durch den User nötig ist. Falls diese doch erforderlich ist, kann das REST Protokoll durch die darunterliegenden Protokolle abgesichert werden. So kann zum Beispiel durch die Verwendung von HTTPS die gesamte Kommunikation des REST Protokolls abgesichert werden [12](#).

IoT Verwendung

Da Protokoll kann zwar tendenziell im IoT verwendet werden, allerdings fehlen einige hilfreiche Features wie zum Beispiel das bestehen lassen einer Verbindung.

CoAP

Protokollübersicht

Das “constrained application protocol” (CoAP) basiert auf dem REST Architekturmodell. Es wurde 2014 von der Internet Engineering Task Force (IETF) als RFC eingereicht und ist als Protokoll zur Machine to Machine (M2M) Kommunikation im Internet of Things (IoT) gedacht. Es eignet sich besonders für diese Art der Kommunikation, da es nicht wie das normale Rest Modell auf http und damit TCP setzt, sondern die Daten über UDP ausgetauscht werden. Damit kann der durch das Protokoll entstehende Overhead deutlich reduziert werden. Deshalb ist diese Art der Datenübertragung für das Umfeld von Low Power Geräten besser geeignet. CoAP wurde zusätzlich so gestaltet, dass eine Nachricht ohne großen Aufwand auf eine normale REST Schnittstelle umgeleitet werden kann. So wurden die vier Grundfunktionen von REST (Get, Post, Put, Delete) als Aktionen in CoAP übernommen. Auch werden Ressourcen grundsätzlich mit einer URI identifiziert [13](#). Zusätzlich wurden aber weitere Funktionen, die grade im IoT Umfeld nützlich sind, hinzugefügt.

Observe

Anders als bei REST, können die vier Standardoperationen auch zu einer langanhaltenden, zustandsbehafteten Kommunikation genutzt werden. Wird das Observe Flag im Header einer CoAP Nachricht gesetzt, wird zuerst die gewünschte Ressource übertragen. Anschließend kann der Server aber bei Änderungen der Ressource diese sofort wieder an den Client übertragen, ohne dass dieser erneut danach fragt. Beide Kommunikationspartner können gleichermaßen die „Beobachtung“ beenden [6](#).

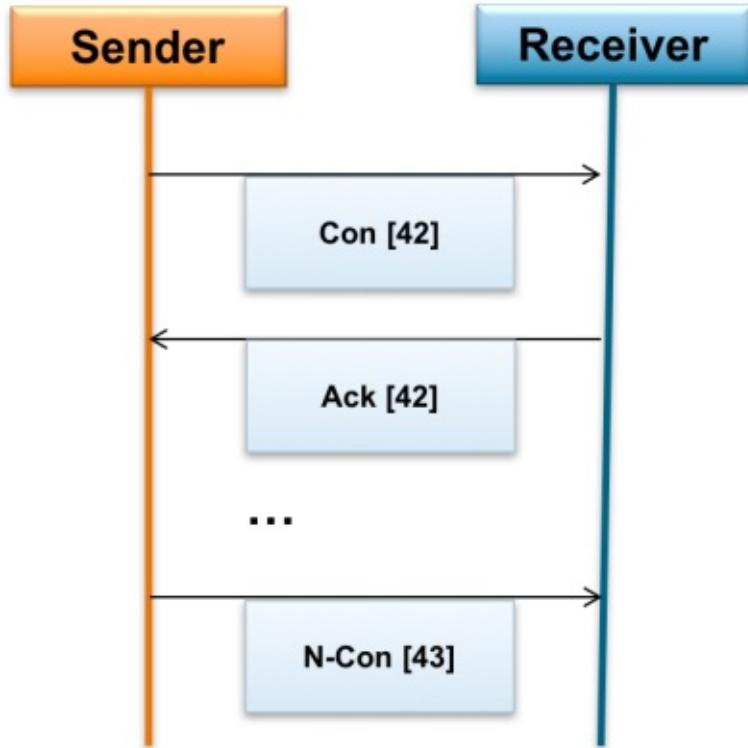
Resource Discovery

Server stellen an einem bestimmten Pfad („./well-known/core“) eine Liste der verfügbaren Ressourcen zur Verfügung. Diese Liste kann von Clients genutzt werden, um nach Ressourcen zu suchen [14](#).

Confirmable und Non-Confirmable Nachrichten

Bei REST wird durch das TCP Protokoll jede Nachricht zu nahezu 100% zugestellt. Hierzu werden die Nachrichten durch ein „ACK“ von Empfänger bestätigt. Obwohl CoAP auf UDP setzt, eine solche Funktion also nicht im darunterliegenden Protokoll verfügbar ist, können wichtige Nachrichten als solche gekennzeichnet werden, sodass eine Empfangsbestätigung

vom Empfänger gesendet werden muss. Allerdings lässt sich dieses Feature für jede Nachricht getrennt definieren, sodass es auch, für „unwichtige“ Daten den Nachrichtentyp „Non-Confirmable“ gibt – „fire'n'forget“ 9.



Sicherheit

Da CoAP nicht auf TCP basiert, sind SSL und TLS nicht einfach zu nutzen. Um trotzdem eine Verschlüsselung zu bieten nutzt CoAP „Datagram Transport Layer Security“ (DTLS) [15](#). Dieses basiert weitestgehend auf TLS und bietet den Clients eine RSA oder AES Verschlüsselung [16](#).

IoT Verwendung

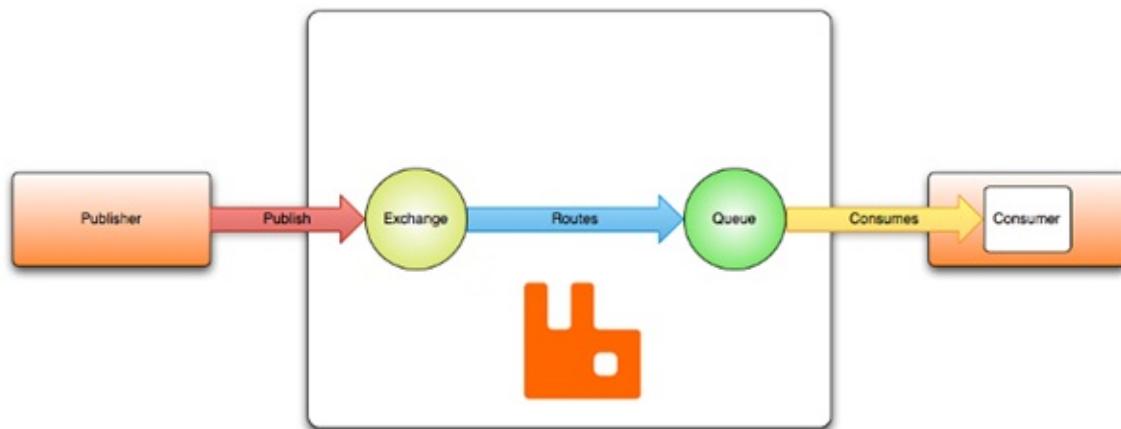
Das Prokotoll eignet sich sehr gut für den Einsatz im IoT. Besonders die gute Kombination mit REST fallen positiv auf.

AMQP

Protokollübersicht

Das Advanced Message Queuing Protocol (AMQP) wurde 2010 als binäres Netzwerkprotokoll auf Anwendungsebene entwickelt [17](#). Es deckt alle Funktionen des „Java Message Services“ (JMS) ab und erweitert diese. Außerdem ist es, anders als JMS Programmiersprachen unabhängig [18](#). Ähnlich wie MQTT und JMS nutzt das Protokoll einen Message Broker (Server) welcher die Nachrichten der Producer an die Consumer weiterleitet. Um die Nachrichten zu transportieren, wird das TCP Protokoll verwendet. Ein Broker verwaltet intern mehrere Exchanges, welche die Nachrichten der Producer annehmen und mithilfe von definierten Regeln (routes) an eine oder mehrere Message Queues übergeben. Von hieraus können die Nachrichten entweder direkt an den Consumer übergeben werden, oder solange zwischen gespeichert werden, bis der Consumer sie abruft. Das untenstehende Bild zeigt den groben Ablauf der Kommunikation [19](#).

"Hello, world" example routing



Exchanges und Routing

Das Erstellen und Konfigurieren der Routing Regeln wird von den Anwendungen selbst übernommen und muss nicht von den Administratoren des Brokers vorgenommen werden. Grundsätzlich ist der Exchange die Kommunikationsschnittstelle für den Produzenten und die Queue die Schnittstelle für den Konsumenten. Das Protokoll unterscheidet zwischen vier Routing Regeln [19](#).

1. Direct Exchange: ist die Standard Regel. Wenn diese gewählt ist, wird beim Erstellen einer neuen Queue direkt auch ein Exchange mit gleichem Namen erstellt. Diese Regel lässt es so aussehen, als ob Nachrichten ohne Exchange direkt auf einen Queue

gesendet werden können.

2. Fanout Exchange: sendet Nachrichten an alle Queues, die an den Exchange gebunden sind. Wenn also 5 Queues an den Exchange gebunden sind, wird je eine Kopie der Nachricht an diese 5 Queues gesendet. Diese Art ist besonders für „Broadcast“ (Many-to-Many)Nachrichten geeignet.
3. Topic Exchange: sendet Nachrichten zu einer oder vielen Queues. Die Queues werden durch einen „routing Key“ bestimmt. Stimmt dieser mit einem Pattern überein, das beim Erstellen der Queue übergeben wurde, erhält die Queue eine Kopie der Nachricht.
4. Headers Exchange: Nachrichten werden aufgrund von bestimmten im Nachrichten Header definierter Attribute weitergeleitet. Wenn eine Nachricht einen bestimmten Header aufweist, wird sie an eine bestimmte Queue geleitet.

Queues

Queues in AMQP funktionieren ähnlich wie Queues in anderen Message Protokollen. Sie halten Nachrichten solange, bis sie vom Konsumenten abgeholt werden. Queues haben einen eindeutigen Namen, mit welchem sie identifiziert werden. Um das Routing zu ermöglichen haben sie noch einige weitere Attribute. Um eine Queue zu nutzen, muss sie erstellt werden, hierzu sind der Name und die Routing Parameter notwendig. Sollte bei der Erstellung einer Queue schon eine mit gleichem Namen und Attributen bestehen, wird keine neue erstellt, sondern diese genutzt. Wenn die Attribute nicht übereinstimmen, wird ein Fehler zurück geliefert. [19](#).

Sicherheit

Aufgrund der Verwendung des TCP Protokolls, können AMQP Nachrichten und Header durch TLS/SSL abgesichert werden. Um Nachrichten Ende zu Ende zu verschlüsseln, kann die eigentliche Nachricht zusätzlich verschlüsselt werden. Dieses beeinflusst das Routing nicht, da der Nachrichten Header nicht verändert wird [20](#)

IoT Verwendung

AMQP eignet sich durch die vielseitigen Protokolleingeschäften zwar tendenziell für den Einsatz im IoT Umfeld, da es aber eigentlich für den Einsatz im "High Performance Server" Umfeld konzipiert ist, ist es grade auf kleinen eingebetteten Geräten schwierig einzusetzen [28](#).

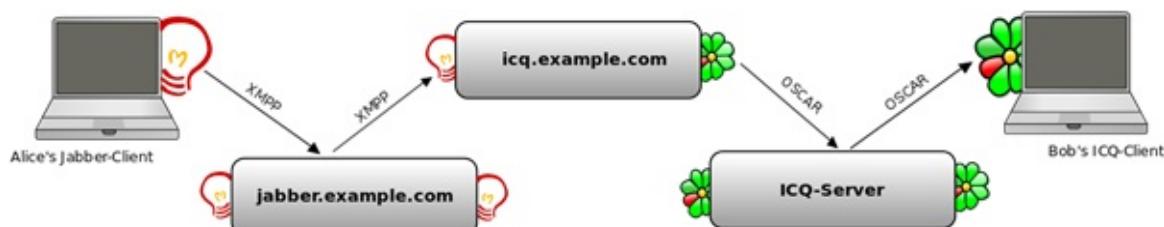
XMPP

Struktur & Nutzen

Das Extensible Messaging and Presence Protocol (XMPP, früher Jabber) wurde zur Nachrichten Übertragung zwischen Benutzern entwickelt. Unter anderem wird es daher zum Instant Messaging eingesetzt. Es wurde sowohl von Google als auch von Facebook für ihre jeweiligen Messenger benutzt. Es basiert auf dem TCP Protokoll und nutzt zur Nachrichtenübertragung den XML Standard. Das Protokoll ist quelloffen und die Entwicklung begann 1998. Standardisiert wurde es 2004 von der IETF. Aber Schon 2003 wurde es von über 10.000.000 Menschen genutzt [21](#).

Features

XMPP bietet Funktionen zum Nachrichtenaustausch zwischen zwei, aber auch mehreren Benutzern, kann den Onlinestatus übermitteln und Dateien und Zertifikate übertragen. Ähnlich wie alle anderen vorgestellten Protokolle nutzt XMPP einen Server um die Nachrichten der Nutzer zu verteilen. Jeder der Server kann mit anderen XMPP Servern kommunizieren, sodass die Kommunikation dezentraler wird. Um Nutzer zu identifizieren hat jeder Nutzer eine ID (JID). Diese ist einer Email Adresse sehr ähnlich und besteht aus einem Namen und einem Server, bei welchem sich der Nutzer befindet. Zusammengesetzt werden beide Daten mit einem @, sodass eine JID die Form „alice@example.com“ hat [21](#). Durch die Möglichkeit Server untereinander zu verbinden, können auch Gateways entstehen, sodass ein XMPP Nutzer mit einem Nutzer eines Anderen Protokolls kommunizieren kann.



XMPP kann durch verschiedene „Extensions“ erweitert werden, sodass es auch Implementierungen für Sprachnachrichten gibt. Diese Erweiterungen ermöglichen es aber auch XMPP im IoT Umfeld zu nutzen. Dieses hat den Vorteil, dass das Protokoll komplett quooffen ist und durch die Server Kommunikation gut skalierbar ist [21](#) [22](#) [23](#). Gegen die Anwendung im IoT Umfeld spricht aber, dass das Protokoll durch die verwendeten XML Nachrichten sehr viel Overhead hat. Dadurch ist es grade auf Low Power Systemen schwierig zu nutzen.

Sicherheit

Auch XMPP basiert auf dem TCP Protokoll, sodass TLS/SSL verwendet werden können. Um eine Ende zu Ende Verschlüsselung zu erzielen, können auch hier die Nutzdaten wiederrum verschlüsselt werden²¹.

IoT Verwendung

Tendenziell eignet sich XMPP gut für den Einsatz im IoT. Das Protokoll verfügt über viele Features (Many-to-Many, Online Status, Server-to-Server, [...]) die im IoT Umfeld sinnvoll erscheinen. Allerdings sind die Nachrichten durch das verwendete XML Format sehr lang, sodass es zu einem erheblichen Overhead kommt ²⁴. Dieses kann grade auf kleinen eingebetteten Geräten zu Problemen führen.

Vergleich und Fazit

	MQTT	REST	CoAP	AMQP	XMPP
Protokoll	TCP	TCP	UDP	TCP	TCP
Eigentliche Verwendung	IoT	Server Backend	IoT	Server Kommunikation	Instant Messaging
Push Nachrichten	Ja	Nein	Ja	Ja	Ja
Online Status *	Ja	Nein	Nein	Nein	Ja
IoT Optimiert	Ja	Nein	Ja	Nein	mit Extension
QoS	Ja	Nein	Ja	Ja	Nein
Many-to-Many	Ja	Nein	Nein	Ja	Ja
Arduino Library	Ja	Ja	Ja	Nein	Ja
Minimal Header	2 Byte ** 27	26 Byte ** 25	4 Byte *** 27	8 Byte ** 26	83 Byte ** 24

* Kommunikationspartner merkt, wenn Problem vorliegt (Online Status / LastWill) **

Zusätzlich 64 Byte für darunterliegende Protokolle *** Zusätzlich 52 Byte für darunterliegende Protokolle

Grundsätzlich lassen sich alle Protokolle mittels Kryptographie so verschlüsseln, dass es Dritten nahezu unmöglich ist, die gesendeten Nachrichten mitzulesen. Aber alle vorgestellten Protokolle haben ihre Vor- und Nachteile, sodass bei der Auswahl eines Protokolls immer auf den genauen Anwendungsfall geschaut werden muss. Grade die Kombination der Protokolle kann es verteilten Systemen ermöglichen äußerst vielseitig eingesetzt zu werden. Hier fällt besonders die Kombination REST / CoAP auf, da CoAP zu diesem Anwendungsfall konstruiert wurde. Aber auch andere Protokoll Kombinationen sind denkbar.

Quellen

- [1]<http://www.hivemq.com/blog/mqtt-essentials-part-8-retained-messages>
- [2]<http://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels>
- [3]<http://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament>
- [4]<https://www.heise.de/developer/artikel/Kommunikation-ueber-MQTT-3238975.html>
- [5]<https://jaxenter.de/iot-allrounder-27208>
- [6]https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php
- [7]<http://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption>
- [8]http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html#_Toc442180924
- [9]<https://www.heise.de/developer/artikel/RESTful-mit-CoAP-3251225.html>
- [10]<https://jaxenter.de/rest-der-bessere-web-service-8988>
- [11]<https://www.mittwald.de/blog/webentwicklung-webdesign/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt>
- [12]https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure
- [13]<http://www.itwissen.info/CoAP-constrained-application-protocol-CoAP-Protokoll.html>
- [14]<https://de.slideshare.net/zdshelby/coap-tutorial>
- [15]<http://coap.technology/>
- [16]https://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security
- [17]<http://www.sic-software.com/iot-protokolle-mqtt-vs-amqp/>
- [18]<https://spring.io/understanding/AMQP>
- [19]<https://www.rabbitmq.com/tutorials/amqp-concepts.html>
- [20]<https://www.amqp.org/product/architecture>
- [21]<https://en.wikipedia.org/wiki/XMPP>
- [22]<http://www.xmpp-iot.org/basics/>
- [23]<https://xmpp.org/uses/internet-of-things.html>

[24]<https://xmpp.org/rfcs/rfc3921.html#messaging>

[25]<https://stackoverflow.com/posts/25065027/revisions>

[26]<http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-transport-v1.0.html>

[27]<https://www.quora.com/What-are-the-pros-and-cons-of-MQTT-versus-CoAP-as-IoT-protocols-for-resource-constrained-devices>

[28]https://lists.oasis-open.org/archives/amqp/201202/msg00086/StormMQ_WhitePaper_-_A_Comparison_of_AMQP_and_MQTT.pdf

Alle Quellen wurden am 27.06.2017 besucht.

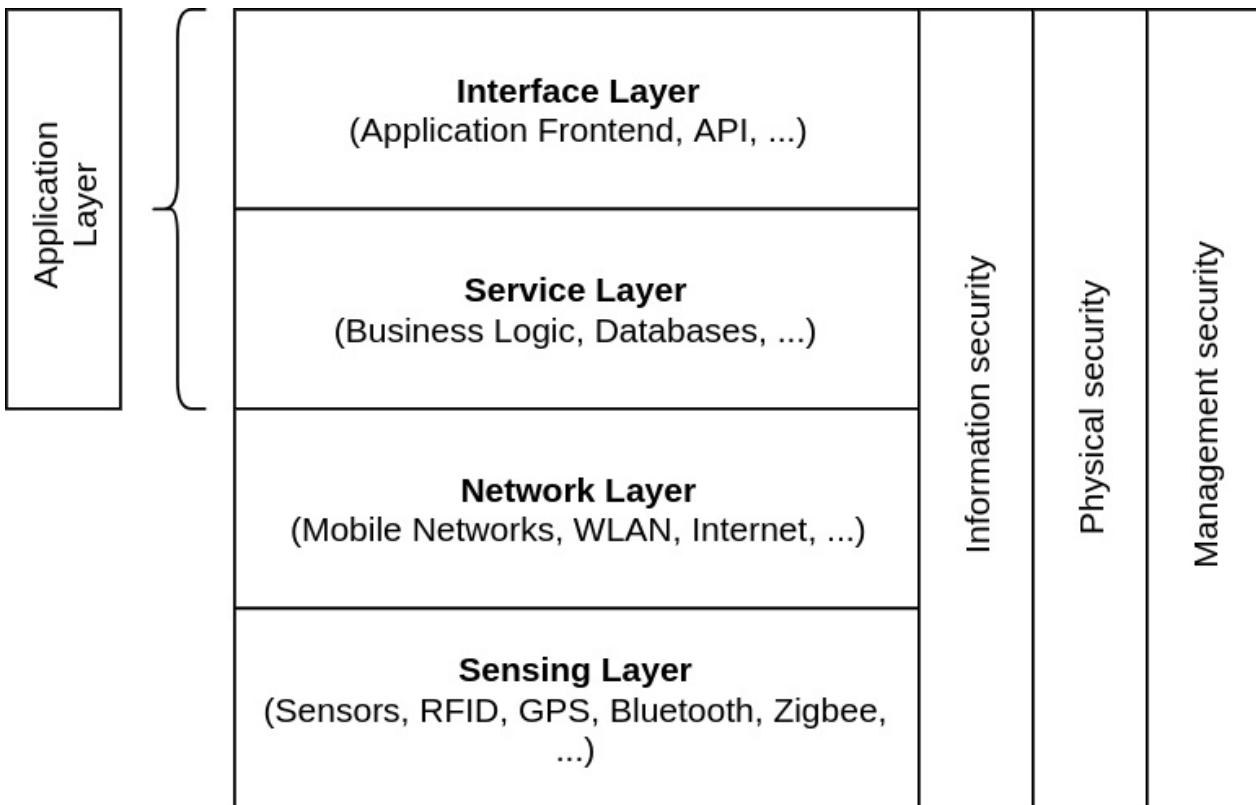
IoT-Sicherheit

Autor: Nils Kohlmeier

Einleitung

Das Internet der Dinge (engl. Internet of Things) wächst schnell. Es wird geschätzt, dass im Jahr 2020 über 40 Milliarden Geräte mit dem Internet verbunden sein werden. Trotz der vielen neuen Möglichkeiten, die uns das Internet der Ding bringt, gibt es auch Gefahren: Je mehr Geräte mit dem Internet verbunden sind, desto attraktiver wird es für Kriminelle, diese anzugreifen. Mit jedem neuen Gerät, das entwickelt wird, nimmt die Menge der möglichen Angriffsvektoren und Gefahren zu. Geräte, die vorher nicht vernetzt waren, können nun über das Internet angegriffen und für die Zwecke der Angreifer eingesetzt werden. [1](#)

Sicherheitsarchitektur



IoT-Sicherheitsarchitektur. Angelehnt an Fig. 3 in 3 und Fig. 2 in 2

Die Sicherheitsarchitektur des Internet der Dinge lässt sich, wie in der obigen Grafik abgebildet, in drei Schichten unterteilen: Wahrnehmungsschicht (Sensing Layer), Netzwerkschicht (Network Layer) und Anwendungsschicht (Application Layer). Jede Schicht hat jeweils bestimmte Anforderungen an die Informationssicherheit, physische Sicherheit und Management-Sicherheit. Die Informationssicherheit beschreibt, wie die Daten selbst gesichert sind, während die physische Sicherheit den Schutz der Hardware beschreibt. Unter Management-Sicherheit versteht man den Zugriffsschutz von Daten und Geräten. [2](#)

Wahrnehmungsschicht

In der Wahrnehmungsschicht befinden sich Geräte, die Daten von Sensoren zusammentragen, diese verarbeiten und anschließend weiterleiten. Viele Geräte sind sehr klein und haben meist nur wenig Leistungsressourcen. So besteht die Gefahr, dass Angreifer Geräte direkt angreifen und so Datenpakete direkt am Gerät abgreifen. Auch die Zerstörung und Diebstahl von IoT-Endgeräten ist möglich. Außerdem können Angreifer versuchen, eigene IoT-Geräte in das System einzubinden und falsche Daten zu liefern. Wegen der geringen Leistungsfähigkeit der IoT-Geräte und der meist sehr begrenzten

Bandbreite der eingesetzten Nahfunktechnologien wie Zigbee oder Bluetooth besteht zusätzlich die Gefahr von Denial of Service-Angriffen (DoS). Hierbei ist das Endgerät für legitime Teilnehmer des Systems nicht mehr erreichbar. [2](#)

Netzwerkschicht

Da sich die Netzwerkschicht um das Weiterleiten von Datenpaketen kümmert und diese über viele verschiedene Arten von Netzwerken übertragen werden, können viele Sicherheits- und Kommunikationsprobleme auftreten. Der Sicherheitsschwerpunkt der Netzwerkschicht liegt bei der Informationssicherheit. Informationen müssen sicher (zum Beispiel verschlüsselt) übertragen werden und die Datenintegrität muss sichergestellt werden. [2](#)

Anwendungsschicht

Die Anwendungsschicht erhält Informationen aus den darunterliegenden Schichten und verarbeitet diese. Sie ist in zwei weitere Schichten unterteilt: Die Service-Schicht (Service Layer), welche die Business Logik sowie Datenbanken enthält, und die Interface-Schicht (Interface Layer), welche verschiedenen Schnittstellen zur Verfügung stellt: Zum Beispiel Benutzeroberflächen (application frontend) oder APIs.

Die Service-Schicht ist dabei für die Verarbeitung und Speicherung der Informationen zuständig, wohingegen die darüber liegende Interface-Schicht nur Schnittstellen für Benutzer und andere Dienste/Programme anbietet. Der Sicherheitsschwerpunkt bei der Service-Schicht liegt bei der Informationssicherheit (sichere Speicherung von Daten), wobei bei der Interface-Schicht der Sicherheitsschwerpunkt bei der Management-Sicherheit liegt. Die Interface-Schicht hat dafür zu sorgen, dass unberechtigte Personen und Dienste sowie Programme nicht auf Daten und Funktionen zugreifen können. Dies wird meist über verschiedene Authentifizierungsverfahren gelöst. Bei Diensten mit mehreren Benutzern muss außerdem sichergestellt werden, dass Benutzer nicht auf Daten und Funktionen von anderen Benutzern zugreifen können. [2](#)

Vorfälle

In den letzten Jahren gabe es einige große Vorfälle, bei denen haupsächlich Internet-of-Things-Geräte eingesetzt wurden. Zwei Beispiele sind die Botnetze LizardStresser und Mirai.

LizardStresser

LizardStresser ist ein Botnetz, dass Ende 2014 entdeckt wurde. Es ist für die Angriffe auf die XBox- und Playstation-Netzwerke verantwortlich, welche als Demonstrationen für das DDoS-Angebot der Gruppe "Lizard Squad" galten. Die einzelnen Bots (Programme, die auf Rechnern oder Geräten laufen, Befehle von Command-And-Control-Servern empfangen und diese ausführen) laufen auf gehackten Routern, die nur mit Standardpasswörtern gesichert waren. Wird ein Router ins Botnetz übernommen, sucht der Router bzw. die Schadware auf dem Router im Internet nach weiteren Geräten, die Standardpasswörter verwenden und versucht diese zu infizieren und in das Botnetz zu übernehmen. LizardStresser kann im Internet für Angriffe gemietet werden, wobei der Server in Bosnien von einem Hoster gehostet wird, bei dem viele bösartige Seiten liegen. [7](#)

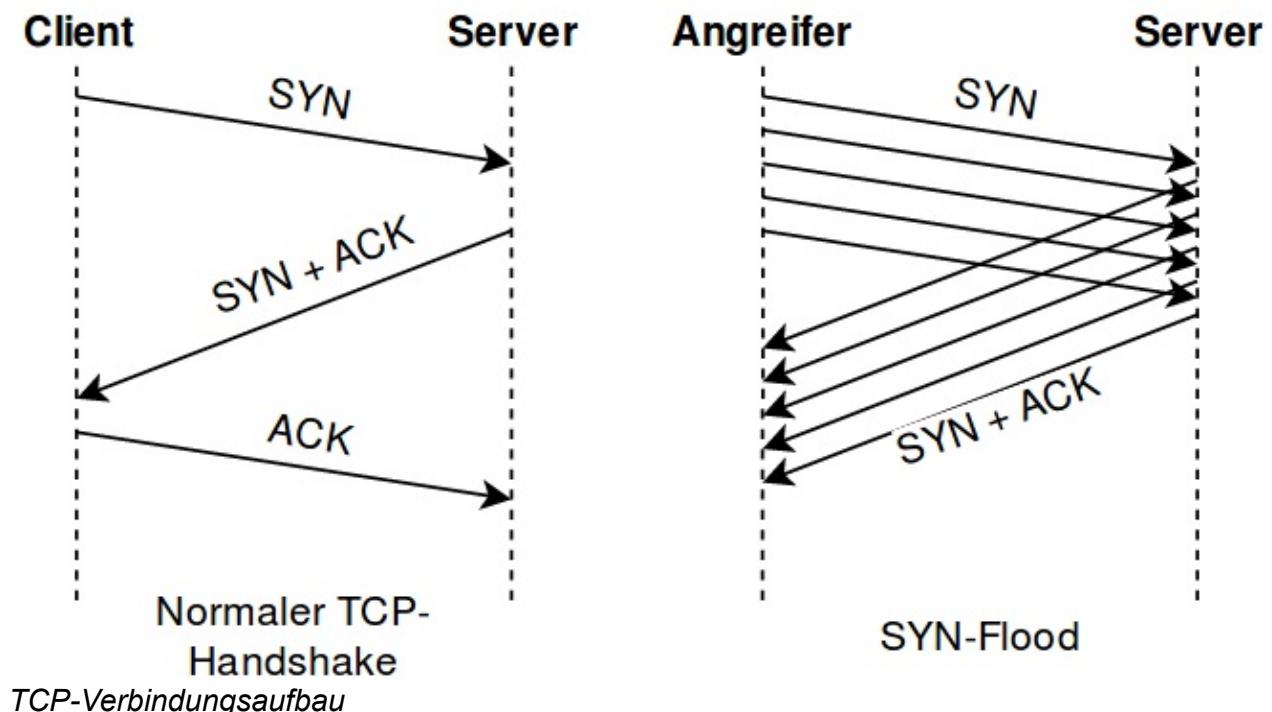
Mirai

Mirai ist ein Botnetz bestehend aus Internet-of-Things-Geräten, die Standardpasswörter verwenden. Das Botnetz wurde 2016 entdeckt und ist für viele große DDoS-Angriffe verantwortlich. So wurde zum Beispiel der Blog von dem IT-Sicherheitsforscher Brian Krebs mit einer Bandbreite von 620 Gbit/s angegriffen. Das Botnetz soll zu Spitzenzeiten aus 380.000 Geräten bestanden haben. Bei einem Angriff auf den französischen Hoster OVH soll Mirai sogar mit einer Bandbreite von 1,1 Terabit por Sekunden angegriffen haben. Kurze Zeit nach den Angriffen wurde der Quelltext von Mirai veröffentlicht, der nun von anderen Angreifern für andere Botnetze verwendet wird. [9](#) [10](#)

Distributed Denial of Service (DDoS)

Ein großes Problem für das gesamte Internet sind DDoS-Angriffe (Distributed Denial of Service), wobei schlecht gesicherte Geräte, die am Internet hängen, in ein großes Botnetz übernommen werden und gleichzeitig ein oder mehrere Ziele angreifen.

Eine mögliche Form des DoS-Angriffs ist hierbei das so genannte "SYN-Flooding", bei dem viele TCP-Verbindungsaufbauanfragen an den Zielserver geschickt werden, ohne auf die Antworten des Servers zu reagieren. Da bei einem TCP-Verbindungsaufbau ein Drei-Wege-Handshake stattfindet, bei dem zuerst ein Client ein TCP-SYN-Paket an den Server schickt, welcher mit einem SYN-ACK-Paket antwortet und ein Bestätigungspaket (ACK-Paket) vom Client erwartet. Der Server wartet eine gewisse Zeit auf das Bestätigungspaket vom Client und speichert deshalb die Anfragedaten des Clients in einem Puffer solange ab, bis entweder die Timeout-Zeit abgelaufen ist oder der Client geantwortet hat. Da die Puffer nur begrenzt groß sind, können diese schnell voll laufen, wenn Clients viele TCP-Verbindungsaufbauanfragen senden, aber nicht die Bestätigungsmessage senden, wenn der Server diese anfordert. So muss der Server für jede neue Verbindung warten, bis die eingestellte Timeout-Zeit abgelaufen ist, bis die Anfrage wieder aus dem Puffer entfernt werden kann. Ist der Puffer voll, kann der Server keine neuen Verbindungen aufbauen und ist somit für andere Clients nicht mehr erreichbar. [5](#)



Eine weitere Form von DoS-Angriffen ist der so genannte "Smurf-Angriff", bei dem von einem Angreifer sehr viele ICMP-Nachrichten mit Ping-Request an die Broadcast-Adresse des Netzwerks geschickt werden. Als Absenderadresse wird der Zielrechner angegeben,

der angegriffen werden soll. Alle Rechner in dem Netzwerk empfangen die Ping-Pakete und antworten mit Request-Paketen, die an den Zielrechner geschickt werden. Dadurch erzeugt der Anreifer, der selbst nur wenige Pakete verschickt, ein so großes Datenaufkommen, dass der Zielrechner überlastet werden kann.⁵

Probleme

Mit steigender Geräteanzahl und Leistungsfähigkeit der Geräte nimmt auch die mögliche Bandbreite für DDoS-Angriffe zu. Ein großes Problem ist, dass Hersteller bei der Entwicklung von Produkten nicht gut genug auf die Sicherheit achten. Viele Konsumenten achten bei dem Kauf von neuen Geräten weniger auf die Sicherheit und mehr auf die Leistungsfähigkeit und Features des Produktes. Somit haben Hersteller andere Prioritäten bei der Entwicklung und Herstellung neuer Produkte, da sie mit sichereren Geräten nicht mehr verkaufen.⁴

Ein weiteres Problem ist, dass viele Hersteller, die früher nichts mit der Entwicklung von Elektronik und Software zu tun hatten (und so auch wenig Erfahrung haben), plötzlich bei dem Trend "IoT" mitmachen wollen und ihre Produkte nun mit einer Internetanbindung ausstatten, ohne grundlegende Sicherheitsmaßnahmen zu bedenken. So implementieren viele Hersteller zum Beispiel keinen Updatemechanismus über den wichtigen Sicherheitspatches eingespielt werden können. Auch die unverschlüsselte Übertragung von Daten ist ein großes Risiko. Angreifer können Logindaten einfach abgreifen und Geräte damit für ihre Zwecke einspannen.⁴

Auch die Verwendung von Standardpasswörtern ist eine große Gefahr. Viele Hersteller vergeben bei der Produktion jedem Gerät das gleiche Passwort. Dadurch sind alle Geräte, die ein Standardpassword verwenden, für jeden frei zugänglich.⁷

Ein weiteres Problem ist, dass viele IoT-Geräte Daten sammeln und diese an den Hersteller schicken. Dieser kann die Daten auswerten und für Werbezwecke verwenden, um neben dem Verkauf von Produkten Geld mit den Daten zu verdienen. Durch diese Datensammlungen sind die Hersteller beliebte Ziele für Hacker, die die Daten auch weiterverkaufen ohne für eigene Zwecke nutzen können.¹¹

Zusätzlich besteht die Gefahr, dass Smarthome-Systeme (teilweise) nicht funktionieren, wenn bei dem Hersteller eine Störung vorliegt. Durch die Abhängigkeit vom Hersteller kann es passieren, dass Benutzer zum Beispiel das Licht nicht einschalten können, wenn der Dienst gerade nicht funktioniert. Auch ein Ausfall der Internetverbindung kann solche Probleme hervorrufen.¹²

Mögliche Lösungsansätze

Lösungsansätze für Hersteller und Behörden

Ein möglicher Lösungsansatz für das Unschädlichmachen infizierter Geräte ist, dass die Server, die die DDoS-Angriffe steuern, von Behörden abgeschaltet werden könnten. Damit besteht aber auch das Risiko, dass legitime Dienst versehentlich mitabgeschaltet werden.⁴

Einen ähnlichen Ansatz verfolgen einige Hacker, die selbst unsichere IoT-Geräte übernehmen und unschädlich machen. Dabei werden diese Geräte zum Teil zerstört.⁶

Ein etwas besserer Lösungsansatz ist die Einführung von Gütesiegeln, die die Käufer dazu animieren, sichere Geräte zu kaufen, die vorher zertifiziert wurden. Achten Käufer auf Gütesiegel, haben Hersteller auch einen Anreiz, sichere Produkte zu entwickeln und zertifizieren zu lassen. Ein Kriterium für die Zertifizierung könnte zum Beispiel eine Update-Verpflichtung des Herstellers sein, bei der er verspricht, während eines festgelegten Zeitraumes Updates für die Produkte anzubieten.⁴

Lösungsansätze für Benutzer

Aber auch Benutzer selbst können die Sicherheit ihrer Geräte erhöhen. Ein wichtiger Schritt, der bei jedem neuen Gerät durchgeführt werden sollte, ist das Ändern der Passwörter. Viele Hersteller verwenden für alle Geräte die gleichen Standardpasswörter. Auch das WLAN-Passwort sollte geändert werden. Als WLAN-Verschlüsselungsverfahren sollte WPA2 gewählt werden.⁷

Schon vor dem Kauf eines neuen Gerätes, kann der Benutzer im Internet nach Sicherheitsvorfällen bei dem bestimmten Gerät suchen. Geräte, die in der Vergangenheit in Bezug auf die Sicherheit schlecht aufgefallen sind, sollten dann nicht gekauft werden. Außerdem sollte man sich vor dem Kauf eines neuen Gerätes beim Hersteller erkundigen, ob er Updates für dieses Gerät zur Verfügung stellt. Nach dem Kauf eines Gerätes sollte in regelmäßigen Abständen überprüft werden, ob neue Updates verfügbar sind und diese einspielen.⁸

Eine weitere Maßnahme ist die Trennung von IoT-Geräten vom restlichen Netzwerk. Eine Firewall sollte alle nicht benötigten Ports dieses Netzwerkes blockieren. Auch das Deaktivieren von UPnP (Universal Plug and Play), mit dem Router und andere netzwerkfähige Geräte konfiguriert werden können, sollte durchgeführt werden, um zu verhindern, dass Malware mithilfe von UPnP Einstellungen ändern kann.⁸

Router und andere IoT-Geräte sollten regelmäßig neugestartet werden, damit Malware, die meist nur im Arbeitsspeicher der Geräte liegt, entfernt wird. Außerdem sollte auch der physische Zugriff auf IoT-Geräte für unbefugte Personen verhindert werden. So können zum Beispiel viele Router mit einer Reset-Taste zurückgesetzt werden, womit meist auch die Sicherheitseinstellungen wieder dem Auslieferungszustand entsprechen.⁸

Smart Home

Gamze Söylev Öktem

Einführung

„Smart“ bedeutet „Intelligent“ auf Deutsch. Das Smart-Haus ist vernetzt und denkt selber. Damit ist Zeitsparen möglich und es senkt Energiekosten. Knapp ein Drittel der Deutschen benutzt schon Smart Home-Komponenten in ihrem zuhause (Schiller 2016). Manche sehen im Smart Home sogar „die zeitgemäße Wohnform für die Anforderungen des 21. Jahrhunderts“ (Projektgruppe Smart Home 2015: 6). Doch wie genau funktioniert eigentlich das Smart Home?

In einem intelligenten Zuhause sind alle Geräte miteinander verbunden und man kann sämtliche dieser Geräte mit dem Smartphone, dem Computer oder einem Tablet überwachen und steuern (Internet of Things). Smart Homes erhöhen die Wohn- und Lebensqualität, sowie die Sicherheit der Bewohner. Man kann, aber muss nicht alles fernsteuern, es gibt auch automatisierbare Abläufe. Man kann viele verschiedene Geräte vernetzen, zum Beispiel Lampen, Jalousien, Heizung, aber auch Herd, Kühlschrank und Waschmaschine. Außerdem kann man auch Video- und Audio Eingaben benutzen, um die Geräte zu kontrollieren. Das Hauptziel ist, den Alltag komfortabler zu gestalten.

Die Idee des Smart Homes hat eine lange Geschichte. Der Begriff ‚smart house‘ wurde zuerst in den 1980er Jahren von der American Association of House Builders verwendet. Doch bereits Jahrzehnte vorher wurden die ersten Smart Homes in 1960ern von Privatpersonen gebaut. Obwohl die Idee des Smart Homes nicht neu ist, war die Entwicklung von Smart Homes eher langsam. Die Hauptgründe für die langsame Entwicklung des Smart Homes sind folgende:

- Mangelnde Motivation zur Produktivitätssteigerung in der Hausarbeit
- Beschränkte Beteiligung der Nutzer im Entwurfsprozess
- Die weit verbreitete Annahme, dass Technologie im Haushalt langweilig ist
- Der Fokus auf Einzelgeräte in der Entwicklung der neuen Technologien (Harper 2003: 1-2)

Die effiziente Entwicklung von Smart Homes ist immer noch schwierig. Ein Problem ist, dass ein Haushalt im Gegensatz zum Arbeitsplatz keine Bereiche wie den technischen Support hat. Ein anderes Problem ist, dass die Benutzergruppe sehr heterogen ist. Sie kann Kinder, Person mittleren Alters, sowie alte Menschen enthalten. Außerdem ist es schwer, ein Haus zu studieren, weil niemand möchte, dass er zuhause Tag und Nacht überwacht wird (Harper 2003: 1-2). Das ist nicht nur ein Problem beim Erforschen und Entwickeln des Smart Homes: Viele Menschen denken, dass die ‚intelligenten‘ Geräte zuhause alles speichern werden und ihre Privatsphäre verletzt wird. Daher haben Sie Bedenken gegen die Nutzung von Smart Home-Technologien.

Trotz dieser zahlreichen Probleme ist die Entwicklung von Smart Homes in den letzten Jahren vorangeschritten. Heutzutage sind Smart Homes in aller Munde. Immer mehr Häuser benutzen Smart Home-Komponenten, auch wenn sie nicht immer ganz ‚smart‘ sind. In dieser Hausarbeit werde ich einige dieser Smart Home Entwicklungen analysieren. Ich werde zuerst erklären, was genau ein Smart Home ist. Danach werde ich drei Aspekte des Smart Home näher beschreiben: Energieeffizienz, Ambient Assisted Living und Sicherheit. Im vierten Kapitel, werde ich auf verschiedene Möglichkeiten für die Technische Umsetzung eingehen, bevor ich im abschließenden Kapitel ein Fazit ziehe.

Smart Home: Definitionen

Eigentlich gibt es bisher keinen allgemeinen Begriff für Smart Home. Die Begriffe Connected Home, Elektronisches Haus, Intelligentes Wohnen, Smart House, Smart Environment, Home of the Future, Smart Living, Aware Home können als Synonyme von Smart Home verstanden werden. Genauso wie es verschiedene Begriffe für das Smart Home gibt, gibt es auch verschiedene Definitionen des Konzeptes (Strese et al. 2010). Das Verständnis von Smart Home hängt dabei stark davon ab, in welcher Branche man ist. Im Gesundheitsbereich zum Beispiel wird unter Smart Home eine Wohnung verstanden, die Möglichkeiten zur Krankheitsverhütung, zum Gesundheitsmonitoring und zur Unterstützung bei Gesundheitsproblemen der Bewohner bietet (Solaimani et al. 2015: 371).

Wir können folgende Definition benutzen:

„Das Smart Home ist ein privat genutztes Heim (z. B. Eigenheim, Mietwohnung), in dem die zahlreichen Geräte der Hausautomation (wie Heizung, Beleuchtung, Belüftung), Haushaltstechnik (wie z. B. Kühlschrank, Waschmaschine), Konsumelektronik und Kommunikationseinrichtungen zu intelligenten Gegenständen werden, die sich an den Bedürfnissen der Bewohner orientieren. Durch Vernetzung dieser Gegenstände untereinander können neue Assistenzfunktionen und Dienste zum Nutzen des Bewohners bereitgestellt werden und einen Mehrwert generieren, der über den einzelnen Nutzen der im Haus vorhandenen Anwendungen hinausgeht.“ (Strese et al. 2010: 8)

Damit eine Wohnung ein Smart Home ist, sollte sie folgende Eigenschaften haben (Projektgruppe Smart Home 2015):

- Viele Geräte sind vernetzt oder vernetzbar
- Verbindung zum Internet ist vorhanden, bzw. Remote Kontrolle ist möglich
- Energieeffizienz und Nachhaltigkeit
- Offene Schnittstellen sind vorhanden (APIs)
- Zumindest Media-Anwendungen verfügen über einen schnellen Breitbandzugriff
- Mindestens AAL-Funktionen (Ambient Assisted Living) sind barrierefrei
- Steuerung von außen ist möglich
- Eingebundene Geräte, Sensoren/ Aktoren sind updatefähig

Lobaccaro und Kollegen geben eine etwas abstrakttere Liste von Eigenschaften des Smart Home (Lobaccaro et al. 2016: 2):

- Automation: Die Fähigkeit Funktionen automatisch erfüllen
- Multifunktionalität: Die Fähigkeit verschiedene Aufgaben zu erfüllen

- Anpassungsfähigkeit: Die Fähigkeit die Bedürfnisse der Bewohner zu lernen, sie vorherzusagen und sie zu erfüllen
- Interaktivität: Die Fähigkeit mit Bewohnern zu interagieren
- Effizienz: Die Fähigkeit Aufgaben so zu erfüllen, dass Zeit und Geld gespart wird

Das Thema Smart Home ist also ein sehr komplexes Thema, mit vielen verschiedenen Unterbereichen. Im folgenden Kapitel werde ich drei Aspekte des Smart Home näher beschreiben: Energieeffizienz, Ambient Assisted Living.

Technische Umsetzung

Es gibt viele verschiedene Möglichkeiten, um Smart Home Anwendungen technisch umzusetzen. Bisher existiert noch keine Technologie, die sich allgemein durchgesetzt hat. Stattdessen konkurrieren verschiedene Systeme, wie Z-Wave, ZigBee, Enocean oder KNX, miteinander. Diese Systeme sind meist aber nicht zueinander kompatibel zueinander. Manche sehen dies als Problem und fordern: „Die Vernetzbarkeit von Geräten unterschiedlicher Hersteller muss weiter vereinfacht werden“ (Projektgruppe Smart Home 2015: 10). In diesem Kapitel werde ich verschiedene technische Umsetzungen vorstellen. Besonders konzentriere ich mich aber dabei auf eine mögliche technische Umsetzung, die Z-Wave Technologie. Sofern nicht anders beschrieben, folgt das Kapitel im Wesentlichen den Ausführungen von Christian Pätz (2011).

Eine Option für die technische Umsetzung in einem Smart Home ist eine Technologie, die kodierte Signale durch die Stromleitung des Hauses schickt. Sie heißt „Powerline Carrier Systems (PCS)“. Die Signale werden zu jeder Steckdose und jedem programmierbaren Schalter geschickt. Sie enthält die Adresse des Gerätes und den Befehl (z. B. „Switch off“).

Es gibt unterschiedliche Protokolle für PCS. X10 ist die erste von denen. X10 ist in 1975 bei einer schottischen Firma entwickelt. Mit X10 können die kompatiblen Geräte miteinander kommunizieren. Eine X10-Nachricht enthält folgende Informationen:

- Ein Warnsignal für das System,
- Die Nummer des Gerätes, das die Nachricht nehmen sollte
- Ein Code für die eigentliche Nachricht

X10 ist sehr schnell, aber es gibt Beschränkungen. Die Stromleitung ist nicht zuverlässig, um zu kommunizieren. Es kann Störungen durch die anderen Geräte geben. Die Geräte können die Nachrichten daher falsch oder gar nicht empfangen.

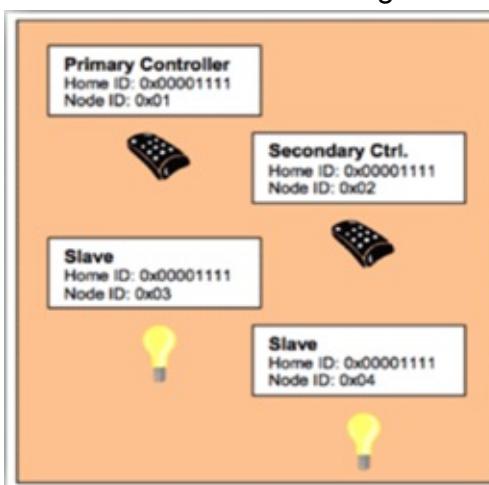
Es gibt andere Protokolle, die nicht die Stromleitung, sondern Radiosignale benutzen (für eine Übersicht siehe Lobaccaro et al. 2016: 9-12). Zwei Beispiele dafür sind ZigBee und Z-Wave. ZigBee und Z-Wave sind beide mesh-Netzwerke. Das bedeutet, dass die Nachrichten mehr als einen Weg benutzen können.

In ZigBee sucht die Nachricht den kurzen Weg, wie in Z-Wave. Allerdings ist ZigBee lediglich geeignet für die Standards vom IEEE (Institute for Electrical and Electronics Engineers) für private kabellose Netzwerke. ZigBee standardisiert nur Funkschichten. Deswegen entwickeln die unterschiedlichen Anwender ihre eigenen Funktionen. ZigBee-Geräte von unterschiedlichen Unternehmen können daher häufig nicht miteinander kompatibel sein. Z-Wave benutzt einen Algorithmus, um den kürzesten Weg zum Gerät zu

berechnen. Er heißt „Source Routing Algorithm“. Die Z-Wave Geräte haben einen Embedded-Code. Wenn ein Gerät zum System angeschlossen wird, erkennt der Kontroller den Code und findet seinen Platz. Es gibt eine Hierarchie zwischen den Geräten. Manche Kontroller können die Nachrichten auslösen, manche können sie nur tragen und antworten (Rosslin und Kim 2010a: 39-40).

Man kann die Gesamtfunktionalität von Z-Wave in drei Schichten aufteilen.

1. Funkschicht: In diesem Bereich wird der Austausch von Signalen zwischen Funkknoten definiert. Das ist der Hardware-Bereich.
 - Wir müssen Abschätzungen zur Funkausbreitung machen. Dabei muss man berücksichtigen, dass Baumaterialien Funksignale unterschiedlich stark dämpfen. Mit diesen Schätzungen treffen wir Entscheidungen, wo die Funkkomponenten aufgestellt werden sollen.
 - Wir müssen dabei Störquellen berücksichtigen. Funkempfänger sollten mehr als 50 cm Abstand von den Störquellen (z. B. Computer, Mikrowellengeräte, Audio- und Videoanlagen) haben.
 - Metallische Möbel oder Gebäudeteile können die elektromagnetischen Wellen beschirmen. Dies nennt man Funkschatten. In dem Funkschatten ist kein Direktempfang möglich (Pätz 2011: 18-31).
2. Netzwerkschicht: Der Datenaustausch von zwei Kommunikationspartnern wird in diesem Bereich definiert. Die Netzwerkebene hat drei Glieder. In der Ebene „Media Access Layer“ wird die Nutzung des Funkkanals kontrolliert. In der Ebene „Transport Layer“ wird kontrolliert, ob der Austausch von einem Datenpaket zwischen zwei Funkknoten korrekt und fehlerfrei ist. In der Ebene „Routing Layer“ wird kontrolliert, ob die Kommandos zwischen den richtigen Kommunikationspartnern ausgetauscht



werden. Wenn der Empfänger eine Nachricht kriegt, schickt er dem Sender eine Bestätigung (ACK). Es gibt verschiedenen Knoten, die miteinander kommunizieren. Jeder Knoten eine Home-ID (gemeinsamer Kennzeichner aller Knoten in einem Netz) und eine Node-ID. Die Knoten, die unterschiedliche Home-IDs haben, können nicht miteinander kommunizieren. Jedes Z-

Wave- Gerät ist entweder „Controller“ oder „Slave“. Slaves steuern nicht die anderen Geräte. Es gibt zwei verschiedene Typen von Slaves: Slaves und Routing Slaves. Es kann einen oder mehrere Kontroller geben. Kontroller kennen alle Wege zu allen anderen Geräten im Netzwerk und können alle Nachrichten schicken. Slaves können nur einer Nachricht antworten, aber selber keine Nachricht schicken. Routing Slaves kennen die Wege zu ihrem eigenen Kontroller und bestimmte Slaves und können nur diesen Nachrichten schicken (Pätz 2011: 32-73).

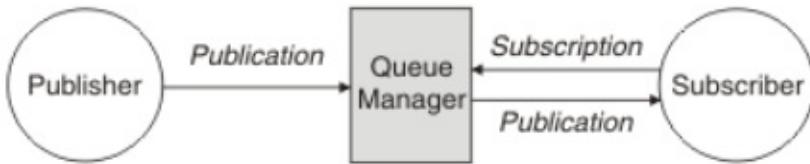
3. Anwenderebene: In diesem Bereich werden die Szenarien der Anwendungen definiert. Man kann viele verschiedene Geräte als Z-Wave-Geräte benutzen (z. B. elektrische Schalter, Motor, verschiedenen Sensoren, Fernbedienung etc.). Die Geräte müssen mit einem Z-Wave- Funkchip ausgerüstet werden. Die Kommunikation mit Geräten ist über Kommandoklassen organisiert. Für jeden Gerätetyp gibt es eine Kommandoklasse. In dieser Kommandoklasse sind die Funktionen des Gerätes enthalten. Außerdem gibt es eine Basic-Kommandoklasse, um mit den Geräten zu kommunizieren. Zusätzlich gibt es Geräteklassen: Basis, Generische und Spezielle Geräteklassen. Dies ermöglicht es, mit Geräten von unterschiedlichen Herstellern zu arbeiten. Die Basis Gerätekasse zeigt uns, ob das Gerät Controller, Slave oder Routing-Slave ist. Die Generische Gerätekasse zeigt uns die Grundfunktion des Gerätes (z. B. Allgemeiner Controller, Statischer Controller oder Eingangscontroller). Spezielle Gerätekasse kann man definieren, wenn das Gerät eine spezielle Funktion hat. Beispielsweise ist Rücksetzthermostat (SETBACK_THERMOSTAT) eine spezielle Gerätekasse der generischen Klasse „THERMOSTAT“.

Über die Geräte- und Kommandoklassen lässt sich sagen, ob ein Gerät dem Z-Wave Standard entspricht. Es muss „mindestens einer Basis- und einer generischen Gerätekasse zugeordnet werden [...], diese korrekt auf Anfrage [angeben] und deren Pflichtfunktionen – sprich Pflichtkommandoklassen - korrekt [implementieren]“ (Pätz 2011: 81). Zusätzlich müssen spezielle Funktion in speziellen Kommandoklassen angegeben werden (Pätz 2011: 74-83).

Ein anderes, sehr interessantes Protokoll ist EnOcean. Die Sensoren und Aktoren in einem EnOcean- System verbrauchen sehr wenig Strom und benutzen „energy harvesting“ Methoden, so dass sie ohne Batterie funktionieren können. Allerdings sind die Kosten für ein EnOcean-System relativ hoch, deswegen ist EnOcean nicht so weit verbreitet.

MQTT (Message Queuing Telemetry Transport) ist ein anderes Beispiel. MQTT wurde im Jahre 1999 entwickelt. MQTT benutzt die Bandbreite effizient und verbraucht sehr wenig Batterie. Damit ist es gut geeignet für Geräte mit wenig Verbindungstärke, beziehungsweise schwachen Netzwerken. MQTT benutzt eine „publish/ subscribe“ Architektur statt einer „request/ response“ Architektur. In dieser Architektur gibt es „publishers“ (Sender) und „subscribers“ (Empfänger). „Publishers“ schicken die Nachrichten unter bestimmten

Klassennamen. Sie wissen nicht, wer die Nachrichten annimmt. „Subscribers“ definieren welche Klassen von Nachrichten sie empfangen wollen. Sie wissen nicht, wer die Nachrichten verschickt. Die folgende Grafik von IBM zeigt eine einfache „publish/ subscribe“ Konfiguration (IBM 2017):



Eine MQTT Session hat vier Stufen: Verbindung, Authentifizierung, Kommunikation und Beendigung. Der Client verbindet sich über eine TCP/IP Verbindung mit dem „broker“. Der Client authentifiziert sich entweder mit SSL/TLS oder Klartext Benutzername und Password. Manchmal werden auch anonyme Clients ohne Benutzername- und Passwortangabe zugelassen.

Die Nachrichten sind sehr kurz. Sie enthalten einen 2-Bytes-Header, einen optionalen Header, die eigentliche Nachricht (maximal 256 MB) und eine Quality of Service (QoS) Eingabe. Der QoS Level zeigt an wie die Nachricht behandelt werden soll. Wenn ein Client (publisher oder subscriber) die Verbindung beenden möchte, schickt er eine „DISCONNECT“ Nachricht zum Broker.

Wie oben beschrieben, ist die Authentifizierung in MQTT in manchen Systemen recht einfach. MQTT ist ursprünglich nicht für unsichere Umgebungen entworfen worden. Wenn SSL/TLS nicht benutzt wird, kann dies zu Sicherheitsproblemen führen. Wenn es benutzt wird, ist die Nachricht grösser. Ein anderes Problem ist, dass man leicht schädliche Nachrichten in das Netzwerk schicken kann. Der Grund ist, dass „Subscribers“ nicht wissen, wer die Nachricht schickt (Rouse 2015).

Fazit

In diesem Artikel habe ich den Bereich Smart Home beschrieben. Das Smart Home und das Internet of Things gewinnt immer mehr an Bedeutung. Von manchen wird das Smart Home sogar als „die zeitgemäße Wohnform für die Anforderungen des 21. Jahrhunderts“ (Projektgruppe Smart Home 2015: 6) gesehen. In dieser Arbeit habe ich zuerst die Bereiche Energieeffizienz, Ambient Assisted Living und Sicherheit vorgestellt. Danach habe ich verschiedene Möglichkeiten der technischen Umsetzung näher beschrieben.

Bei der technischen Umsetzung habe ich mich vor allem auf Z-Wave Systeme und das MQTT Protokoll konzentriert. Ein Problem des Smart Home ist, dass verschiedene Systeme existieren, die häufig nicht miteinander kompatibel sind. Daher wird von manchen gefordert, dass die „Vernetzbarkeit von Geräten unterschiedlicher Hersteller [...] weiter vereinfacht werden [muss]“ (Projektgruppe Smart Home 2015: 10). Es wird sich zeigen, welche Standards und Protokolle sich in Zukunft durchsetzen werden.

Smart Homes machen das Leben einfacher und komfortabler. Besonders für die alten und behinderten Menschen sind sie sehr hilfreich und wichtig. Sie haben auch viele Sicherheitsvorteile, zum Beispiel, weil sie Überwachungsmöglichkeiten bieten. Andererseits bringen sie eigene Sicherheitsprobleme mit sich. Dies wurde im Oktober 2016 deutlich als ein Hackerangriff das Internet of Things benutzte um Webseiten wie Twitter und PayPal anzugreifen. Man befürchtet, dass diese Angriffe in den nächsten Jahren häufiger und stärker werden, wenn immer mehr Menschen Smart-Home-Geräte nutzen (Thielman 2016).

Bibliographie

Aldrich, Francis K. (2003) Smart Homes: Past, Present and Future. In: Harper, Richard (ed.) (2003) Inside the Smart Home, pp.17-40. London: Springer.

Harper, Richard (2003) Inside the Smart Home: Ideas, Possibilities and Methods. In: Harper, Richard (ed.) (2003) Inside the Smart Home, pp.1-14. London: Springer.

HIVEMQ (2017) MQTT 101 – How to Get Started with the lightweight IoT Protocol.

Aufgerufen am 01. Juli 2017 unter: <http://www.hivemq.com/blog/how-to-get-started-with-mqtt>

IBM Knowledge Center (2017) IBM MQ, Version 9.0. Publish/subscribe messaging.

Aufgerufen am 01. Juli 2017 unter:

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.pro.doc/q004870_.htm

Lobaccaro, Gabriele; Carlucci, Salvatore und Löfström Erica (2016) Review of Systems and Technologies for Smart Homes and Smart Grids. Energies, Vol. 9(5).

Nationaler IT-Gipfel Berlin 2015 (2015) Smart Home. Ergebnisdokument der Projektgruppe Smart Home. Konvergenz der Netze. Nationaler IT-Gipfel Berlin 2015. Aufgerufen am 30. Juni 2017 unter: http://www.bmvi.de/SharedDocs/DE/Anlage/Digitales/it-gipfel-fg-konvergenz-pg-smarthome.pdf?__blob=publicationFile

Pätz, Christian (2011) Z-Wave Grundlagen. Funksteuerung im Smart Home. Norderstedt: Books on Demand.

Robles, Rosslin John und Kim, Tai-hoon (2010a) Applications, Systems and Methods in Smart Home Technology: A Review. International Journal of Advanced Science and Technology, Vol. 15, February 2010.

Robles, Rosslin John und Kim, Tai-hoon (2010b) A review on security in smart home development. International Journal of Advanced Science and Technology, Vol. 15, February 2010.

Rouse, Margaret (2015) MQTT (MQ Telemetry Transport). Aufgerufen am 01. Juli 2017 unter: <http://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport>

Saad al-sumaiti, Ameena; Ahmed, Mohammed Hassan und M. A. Salama, Magdy (2014) Smart Home Activities: A Literature Review. Electric Power Components and Systems, 42(3-4), 294-305.

Schiller, Kai (2016) Was ist ein Smart Home? Geräte und Systeme. Aufgerufen am 30. Juni 2017 unter: <https://www.homeandsmart.de/was-ist-ein-smart-home>

Solaimani, Sam; Keijzer-Broers, Wally und Bouwman, Harry (2015) What we do – and don't – know about the Smart Home: An analysis of the Smart Home literature. Indoor and Built Environment, Vol. 24(3), 370-383.

Strese, Hartmut; Seidel, Uwe; Knappe, Thorsten und Botthof, Alfons (2010) Smart Home in Deutschland. Untersuchung im Rahmen der wissenschaftlichen Begleitung zum Programm Next Generation Media (NGM) des Bundesministeriums für Wirtschaft und Technologie. Berlin: Institut für Innovation und Technik (iit) in der VDI/VDE-IT. Aufgerufen am 30. Juni 2017 unter: <http://www.iit-berlin.de/de/publikationen/smart-home-in-deutschland>

Thielman, Sam (2016) Can we secure the internet of things in time to prevent another cyber-attack? The Guardian, 25. Oktober 2016. Aufgerufen am 30. Juni 2017 unter: <http://www.iit-berlin.de/de/publikationen/smart-home-in-deutschland>

Thielman, Sam und Hunt, Elle (2016) Cyber attack: hackers 'weaponised' everyday devices with malware. The Guardian, 22. Oktober 2016. Aufgerufen am 30. Juni 2017 unter: <https://www.theguardian.com/technology/2016/oct/22/cyber-attack-hackers-weaponised-everyday-devices-with-malware-to-mount-assault>

Gesichtserkennung mit OpenCV

Autor Justin Jagieniak

Einführung

Das Thema beschäftigt sich mit der Gesichtserkennung mit Hilfe des OpenCV-Frameworks. Die Gesichtserkennung ist ein Programm, das durch Computer Vision und Bildverarbeitung in der Lage ist Gesichter zu erkennen. Computer Vision beschäftigt sich mit der Analyse, der Verarbeitung und dem Verstehen von Bildern. Der Computer versucht dabei die Verarbeitung der Bilder im Gehirn des Menschen zu imitieren. Das Gehirn des Menschen kann Formen in den Bildern, die von der Iris ankommen, erkennen und sie verarbeiten. Gesichtserkennung wird heute immer interessanter. Zum Beispiel wird von Thomas de Maizière vorgeschlagen, Gesichtserkennung flächendeckend im öffentlichen Raum zur Personenüberwachung und damit zur Terrorabwehr oder auch das Auffinden von Kriminellen einzusetzen.[2]

Aber auch im Bereich des Embedded Computing (welches das Thema meines Projektes ist) wird der Gesichtserkennung eine immer größer werdende Bedeutung beigemessen. Man will mit Hilfe der Gesichtserkennung die Stimmungen von Menschen erkennen, erkennen um welche Person es sich handelt oder auch einzelne Personenmerkmale zuordnen. Zum Beispiel die Entscheidung, ob die Person männlich oder weiblich, ein Kind oder erwachsen ist. Beliebte Anwendungsgebiete der Gesichtserkennung im Bereich des Embedded Computing finden sich in Robotern, Smartphones, Digital- und Überwachungskameras usw.

Natürlich gibt es dabei auch Problematiken zu berücksichtigen. Durch die Gesichtserkennung ist der Weg in einen Überwachungsstaat ein leichter. Man braucht sich nur vorzustellen, dass eine jede Person durch eine Überwachungskamera und ein Programm identifiziert wird. Das heißt man könnte von der gesamten Bevölkerung für jede einzelne Person ein Überwachungsprofil erstellen. Man könnte zum Beispiel orten, wo er einkauften war, wen er besucht hat und so weiter. Natürlich ist dies sehr nützlich für die Verbrechensbekämpfung. Verbrechen ließen sich durch Bewegungsprofile sehr schnell aufklären und es würde den Gerichten und der Kriminalpolizei einiges an Arbeit ersparen. Aber ist auch die Gefahr für den Missbrauch sehr hoch. Wer sollte die Personen, die die Kontrolle darüber haben, daran hindern dies zu missbrauchen? Man muss sich hierbei natürlich die ethische Frage stellen, ob man die Freiheit zugunsten der Sicherheit aufgibt. Man könnte am Ende beides verlieren.

Das Prinzip der Gesichtserkennung

Um ein Gesicht zu erkennen, muss erst einmal eine Strategie überlegt werden, wie man das Gesicht am besten in einem Bild erkennen kann. Eine einfache Kantenerkennung, die im Kapitel OpenCV besprochen wird, reicht leider nicht aus bzw. funktioniert nicht immer zuverlässig. Man muss also eine Strategie entwickeln, mit der man zuverlässig ein Objekt (hier ein Gesicht) auf einem Bild erkennen kann und es entsprechend klassifiziert.

Im folgenden Verlauf werden wir uns damit beschäftigen, wie man ein Gesicht erkennen und klassifizieren kann. Das heißt wir wollen erkennen, ob ein Gesicht auf dem Foto vorhanden ist und dieses Gesicht einer uns bekannten Person aus einem Trainingsset von Personen zuordnen.

Für die Erkennung eines Gesichtes auf einem Foto benutzen wir den Haar-Cascade. Für die Zuordnung des erkannten Gesichtes bedienen wir uns drei verschiedenen Algorithmen. Die Namen der Algorithmen lauten "Eigenface", "Fisherface" und "LBPH".

OpenCV

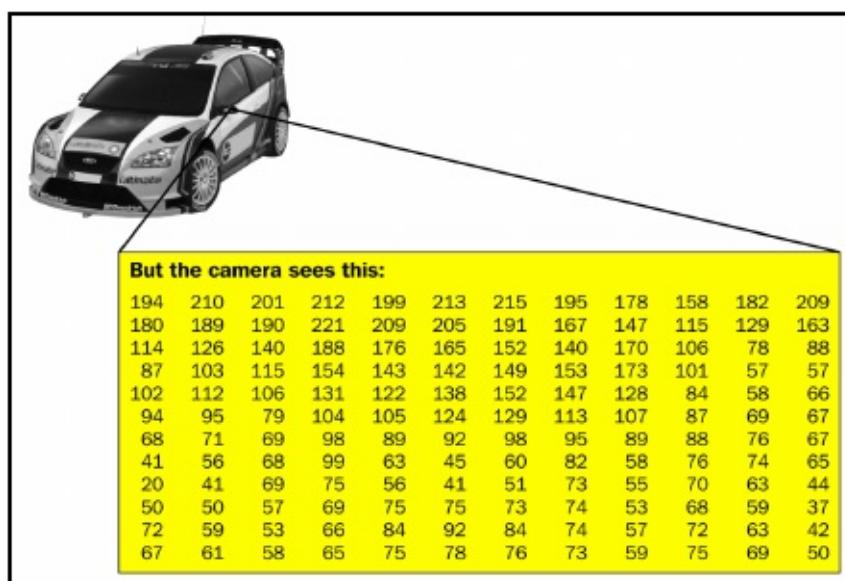
Was ist OpenCV?

OpenCV ist eine Programmbibliothek für Computervision und Maschinenlernen, welche unter einer OpenSource Lizenz steht. Es ermöglicht Softwareentwicklern Applikationen, die mit der Computervision zusammenhängen, zu entwickeln. Dabei besitzt OpenCV eigene Mechanismen für die effiziente Speicherverwaltung, der Bildverarbeitung und den Beschaffen von Bildern. OpenCV selbst ist in C++ programmiert, kann aber sowohl in den Programmiersprachen C++, als auch Python benutzt werden. Auch gibt es Schnittstellen von OpenCV für andere Programmiersprachen, wie zum Beispiel Java.[1.1]

Die Klasse Mat

Den Kernpunkt von OpenCV bildet eine Klasse namens Mat. Das was wir als Bilder sehen, ist in Wirklichkeit eine Aneinanderreihung von Informationen, auch Pixel genannt. Ein Pixel besteht bei einem Graustufenbild aus 8 Bits. Genauer gesagt aus 255 Möglichkeiten an Intensitäten zwischen Schwarz und Weiß. Ein Farbbild wird aus den Elementarfarben Rot, Grün, Blau gebildet. Dabei kann jede einzelne der drei Elementarfarben in einer Intensität von 0 bis 255 vorkommen. Aus eben jenen Informationen können bis zu 24 Bit Farben entstehen. Das entspricht einer Zahl von 16.777.216 Farben. Diese sind für ein Farbbild ausreichend und auch für das menschliche Auge wahrnehmbar.[1.1]

Ein Farbbild oder auch Graubild sind damit in Wirklichkeit eine 2-dimensionale Matrix in einer Aneinanderreihung von Informationen bzw. Pixeln. Die Klasse Mat sorgt für eine effiziente Speicherverwaltung und besteht aus einem Value und einem Header. Der Value ist die Matrix, die das Bild repräsentiert. Der Header enthält Informationen über die Matrix, wie zum Beispiel die Größe und die Anzahl der Pixel, sowie einen Pointer, der den Speicherbereich im Speicher addressiert. Farbbilder werden in 3 Channels in der Klasse unterteilt. Jeweils ein Channel für eine der drei Elementarfarben. Sollte nur ein Channel vorhanden sein, so ist das Bild schwarz/weiß. Aus der Klasse Mat kann ein Objekt abgeleitet werden. Dieses Objekt repräsentiert dann ein Bild.[1.1]



Veranschaulichung einer Matrix zu einem Bild.[1.1]

Kanten- und Blurerkennung in OpenCV

OpenCV ist insgesamt eine mächtige Library. Verschiedene Arten von Detektoren, Filtern und Bildableitungen erlauben es Informationen aus Bildern zu extrahieren. So ist es zum Beispiel möglich, Kanten in OpenCV zu erkennen. Eine Kante in einem Bild wird von OpenCV als starke Intensitätsveränderung von zwei Pixeln definiert. Wie man auch in folgendem Beispiel erkennen kann:[1.2]



Beispiel von Kanten in OpenCV.[1.3]

Anhand des rechten Bildes sieht man, dass OpenCV die Kanten dort zeichnet, wo es zu einer starken Intensitätsänderung zwischen Pixeln kam. Damit ergibt sich für die Maschine das, wo das menschliche Auge Konturen von Gegenständen wahrnimmt. Um ein solches Bild zu zeichnen, kann man bei OpenCV den Gaussian Filter oder den Sobel derivative Filter" einsetzen. Auch kann OpenCV die Kanten selbst erkennen, dies geschieht zum Beispiel mit den "Sobel Detector", den "Canny Edge Detector" oder den "Laplacian Operator".[1.3]

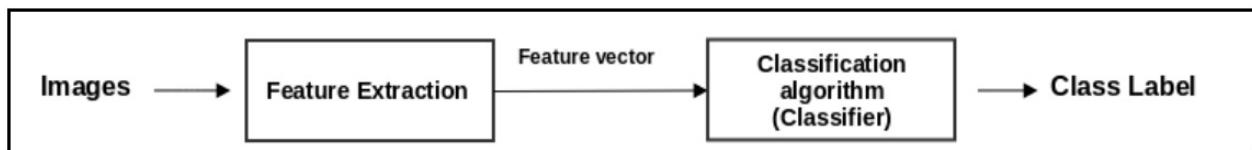
Auch ist es möglich eine sogenannte "Blur" auf Deutsch "Weichzeichnung" in einem Bild zu erkennen. Dies geschieht ebenfalls mit den "Laplacian Operator". Eine Weichzeichnung ist das Gegenteil einer Kante. Eine Weichzeichnung tritt dann auf, wenn die Intensitäten zwischen Pixeln ähnlich sind und nicht stark von einander abweichen.[1.3]

Haar Cascades

Paul Viola und Michael Jones entwickelten in ihren Paper "Rapid Object Detection Using a Boosted Cascade of Simple Features" 2001 eine Methode, um mit Hilfe einer Trainingsmenge an positiven (Bilder mit dem Objekt) und negativen Bildern (Bilder ohne Objekt) eines Objektes ein ähnliches Objekt auf einem Bild zu erkennen.[1.4,3-4]

Das Prinzip

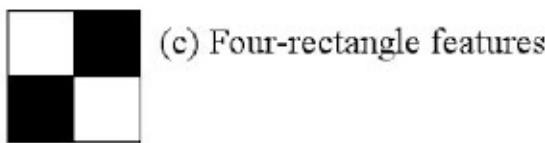
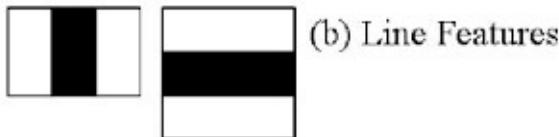
Benötigt werden eine große Menge an positiven und negativen normalisierten Graustufenbildern. Je mehr unterschiedliche Bilder man hat, desto größer ist auch die Genauigkeit der Erkennungsrate. Das Ziel ist es eine Cascade Function herauszuarbeiten, die ein ähnliches oder identisches Objekt auf einem beliebigen Bild erkennen kann.
Folgender Flowchart erklärt das Grundprinzip:



Vorgehensweise für Objekterkennung in OpenCV.[1.4]

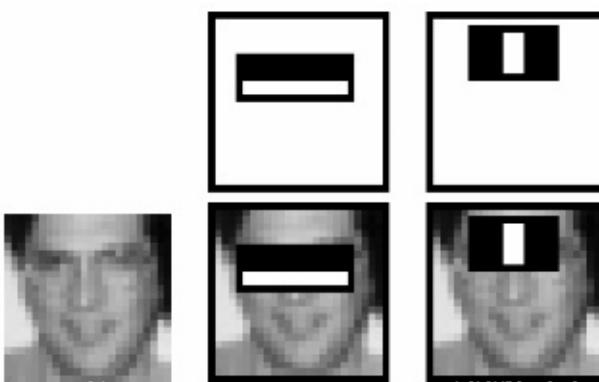
Im gezeigten Flowchart kann man erkennen, dass man am Anfang ein Set von Bildern hat. Anschließend wird eine Methode namens "Feature Extraction" angewendet. Die Feature Extraction sorgt dafür, dass aus den Bildern mit Hilfe von verschiedenen Haar Features (siehe Abbildung) ein Feature Vector extrahiert wird. Dabei werden die Haar Features in verschiedenen Skalierungen auf die Graustufenbilder angewandt. Ein Feature Vector (auf deutsch Merkmalsvektor) ist ein Vektor von Feature, die ein bestimmtes Muster wiedergeben, das vom Programm als gemeinsames Muster bei einer Mehrheit von Bildern

(siehe Abbildung) ein Feature Vector extrahiert wird. Dabei werden die Haar Features in verschiedenen Skalierungen auf die Graustufenbilder angewandt. Ein Feature Vector (auf deutsch Merkmalsvektor) ist ein Vektor von Feature, die ein bestimmtes Muster wiedergeben, das vom Programm als gemeinsames Muster bei einer Mehrheit von Bildern aus dem Bildset erkannt wurde.[1,4, 3]



Verschiedene Haar Features.[3]

Existiert ein solcher Feature Vector kann vom Programm berechnet werden, welche Bilder aus dem Bilderset als positive oder negative Bilder zu klassifizieren sind. Dies erfolgt durch Prüfung der gesammelten Features im Feature Vector. Da man an Rechenzeit sparen möchte, werden nicht alle Features für jedes einzelne Bild durchgegangen, sondern man prüft die Features nacheinander. Sollte ein Feature als negativ erscheinen, wird das Bild als negativ bewertet und es wird zum nächsten Bild im Bildset übergegangen. Hier sieht man, dass das Klassifizieren den Klassifizierungsalgorithmus (Classification algorithm) entspricht und dass die positiven und negativen Bilder die Klassennamen (Class label) sind. Diese könnten beispielsweise "Gesicht" und "KeinGesicht" sein.[3]



Anwendung von Haar Features als Beispiel.[3-4]

Als Beispiel in der Abbildung oben sieht man, dass ein Haar Feature im Gesicht erkannt wurde. Im ersten Feature (Bild mitte) wurde erkannt, dass die Region der Augen öfters dunkler ist, als die Region um die Nase und den Wangen. Im zweiten Feature wurde erkannt, dass die Augen dunkler sind als die Nasenbrücke.[3]

Glücklicherweise nimmt OpenCV dabei einiges an Arbeit ab. Es ist für eine Gesichtserkennung in OpenCV nicht notwendig einen eigenen Haarcascade für den Algorithmus anzulernen. OpenCV implementiert im Framework den Algorithmus und hat verschiedene Default Haar Cascades im OpenCV Ordner. Sollte man ein eigenes Objekt erkennen wollen, so ist es notwendig vom Objekt ein eigenes Haar Cascade anzulernen. [1.5] Weiterhin sieht man auch anhand der Haar Features, dass auch hier die Kantenerkennung, die im Kapitel OpenCV kurz erwähnt wurde, eine Rolle spielt.

Eigenfaces

Eigenfaces (auf deutsch Eigengesichter) ist ein von Matthew Turk und Alex Pentland entwickeltes Verfahren, um Gesichter einer Person oder bestimmten Merkmalen zuordnen zu können. Dabei setzt Eigenfaces auf eine Principal Component Analyse (PCA) auf deutsch Hauptkomponentenanalyse. Die Principal Component Analyse dient dazu mit einer Anzahl an statistischen Variablen sich einer möglichst aussagekräftigen Linearkombination zu nähern. Dieses Verfahren ist besonders in der Bildverarbeitung beliebt.[10]

Funktionsweise

Siehe: [5] (Die Rechnung ist ein wenig verkürzt aus Wikipedia kopiert und steht hier zum Verständnis der Funktionsweise.)

Hinweis: Die Rechnung erfolgt mit Hilfe der Matrix des Bildes (wie auch im Kapitel OpenCV beschrieben). Man darf sich dabei ein Bild als Matrix vorstellen. Auch das Differenz- und Durchschnittsbild sind Matrizen.

- Schritt 1: In jenem Verfahren werden eine Trainingsmenge an Bildern von Gesichter in lexikographischer Reihenfolge eingelesen und in Vektoren einer Länge k gespeichert.
 $\Gamma_1, \Gamma_2, \Gamma_3 \dots \Gamma_M$
- Schritt 2: Aus der Trainingsmenge wird schließlich ein Durchschnittsgesicht gebildet.

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n$$

- Schritt 3: Anschließend wird von jedem Bild aus dem Trainingsset ein Differenzbild zum Durchschnittsgesicht gebildet.
 $\Phi_i = \Gamma_i - \Psi$
- Schritt 4: Sind die Differenzbilder vollständig berechnet, kann man eine Kovarianzmatrix erstellen.

$$C = \frac{1}{M} \sum_{n=1}^M \Phi_n \Phi_n^T = AA^T$$

- Schritt 5: Allerdings ist bei einer Kovarianzmatrix der Nachteil vorhanden, dass für einen modernen Computer schon das Rechnen von einer Anzahl von 300 100x100 Pixel großen Bildern rechenaufwendig ist. Um dies zu verhindern sollten die Eigenvektoren statt in einer Kovarianzmatrix in einer neuen Marix berechnet werden.

$$L := A^T A \in \mathbb{R}^{M,M}$$

Das dies möglich ist, wird bewiesen durch:

- Schritt 5.1: Sei die Eigenwertzerlegung von der Kovarianzmatrix gegeben durch:

$$Cv_i = AA^T v_i = \lambda_i v_i$$
- Schritt 5.2: Da die Kovarianzmatrix eine zu große Matrix ist, wird in diesem Schritt die Eigenwertzerlegung für L betrachtet.

$$Lu_i = A^T Au_i = \lambda_i u_i$$
- Schritt 5.3: Bei linksseitiger Multiplikation mit A ergibt sich:

$$AA^T Au_i = \lambda_i Au_i$$
- Schritt 5.4: Sei nun $v_i := Au_i$, So ergibt sich aus der Eigenwertzerlegung von L dieselbe wie von C. Damit ist es bewiesen. Die erhaltenen Vektoren von v sind die Eigenvektoren, wobei nur die mit den höchsten Eigenwert von Interesse sind. Die u's müssen noch normalisiert werden.
- Schritt 6: Schlussendlich können die ausgerechneten Eigengesichter dann in einem Gesichtsraum projiziert werden, so dass man den daraus erhaltenen Vektor für die Gesichtswiedererkennung nutzen kann.

$$\omega_k = u_k^T (\Gamma - \Psi) \quad k = 1 \cdots M'$$
- Schritt 7: Der aus Schritt 6 erhaltene Vektor lässt sich für die Gesichtswiedererkennung nutzen:

$$\Omega^T = [\omega_1, \dots, \omega_{M'}]$$

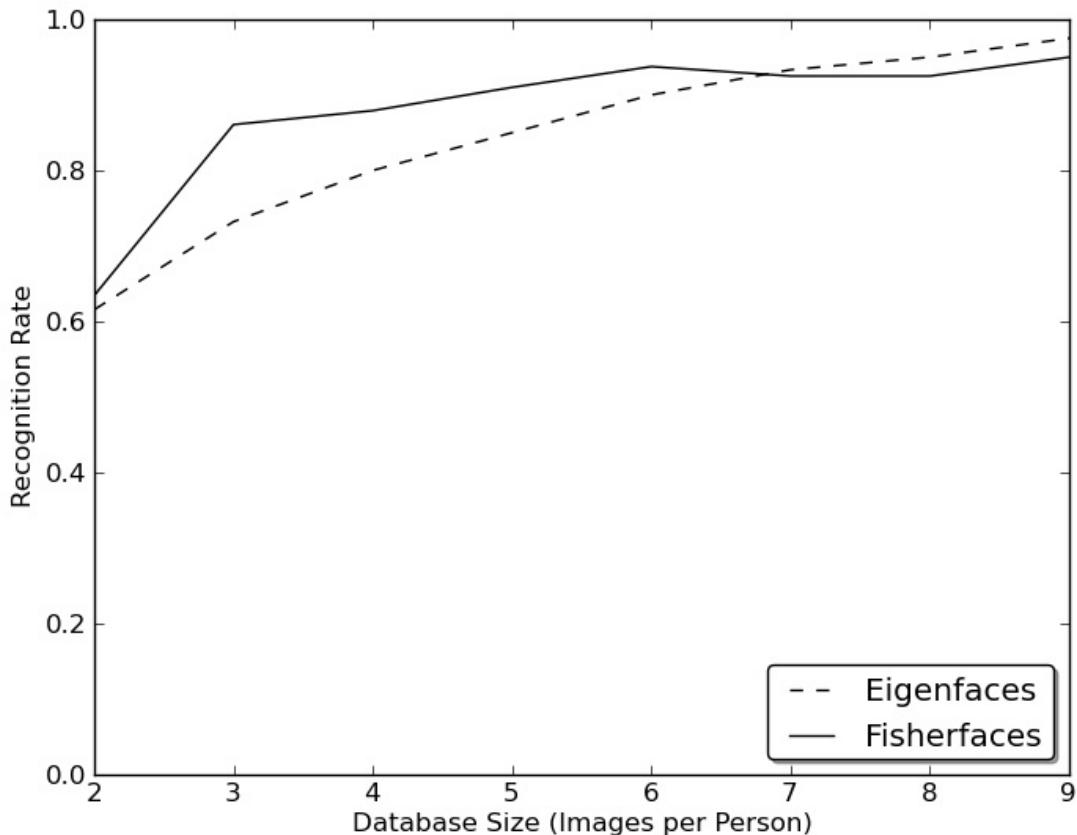
Eigenfaces in OpenCV

Natürlich ist es nicht notwendig in OpenCV die komplette Berechnung selbst einzuprogrammieren. Dies wäre entsprechend viel Arbeit, ließe sich aber durch die Mat Klasse bewerkstelligen. Die Berechnung der Eigenfaces ist in OpenCV in einem Zusatzmodul namens Contrib vorhanden. Dieses muss zusätzlich zur OpenCV Library kompiliert oder kann auch in den Binaries zusätzlich installiert werden. Die Darstellung der Funktionsweise dient zum Verständnis, wie das Ganze eigentlich funktioniert.

OpenCV bietet noch eine kleine Besonderheit. Wenn man ein Bild auf das Trainingsset überprüfen lässt, ist es möglich einen Wert namens "Confidence" aus der Funktion herauszuholen. Der Confidence Wert errechnet sich nach der kNN (k-Nearest-Neighbour [auf dt. Nächste-Nachbarn-Klassifikation]) Methode und gibt die Distanz zu dem am ähnlichsten gefundenen Bild aus dem Trainingsset wieder.[6-7]

Fisherfaces

Fisherfaces funktionieren ungefähr vergleichbar wie die Eigenfaces und ist ein Versuch den Eigenface Algorithmus mit einem Trainingsset von wenigen Bildern zu optimieren. Der Hauptunterschied zwischen den beiden ist, dass Eigenface auf die Principal Component Analyse (kurz PCA) setzt, während die Fisherfaces auf die Linear Discriminant Analyse (kurz LDA) setzen. Mit Hilfe dieser Analyse möchte man eine lineare Kombination von Features finden, welche zwei oder mehr Objekte von Trainingsdaten klassifizieren oder trennen. Dies bedeutet, dass während kleinere Eigenvalues empfindlich auf die Wahl der Trainingsdaten an Bildern reagieren, sind Fisherfaces dagegen toleranter. Veranschaulichen lässt sich dies durch ein Diagramm mit Hilfe von Erkennungsraten.[7,11]



Vergleich der Erkennungsraten von Eigen- und Fisherfaces[8]

Am Bild sieht man, dass Eigenfaces bei weniger Bildern pro Person eine niedrigere Erkennungsrate besitzen als Fisherfaces. Dennoch kann es passieren, dass bei mehr Bildern Eigenfaces eine höhere Erkennungsrate als Fisherfaces haben, auch wenn die Erkennungsrate bei Fisherfaces schwankt, während bei Eigenfaces die Erkennungsrate wie eine Parabel mit oberer Schranke nach oben geht.

Fisherfaces in OpenCV

Genauso wie im Kapitel Eigenfaces beschrieben, gibt es eine Funktion in der Contrib Erweiterung von OpenCV für Fisherfaces. Auch verhalten sich die Confidence gleich. Man kann beide Funktionen auf dieselbe Weise nutzen. Der einzige Unterschied liegt im Funktionsnamen.

LBPH

Die Kurzform von LBPH steht für Local Binary Pattern Histograms und kann auch für die Gesichtserkennung eingesetzt werden. Die Local Binary Pattern (LBP) sind Teil des Textur Spektrum Modells von 1990 und wurden 1994 erstmals beschrieben.[12] LBPH wird ebenfalls zur Gesichtserkennung eingesetzt, in dem Sinn, dass das Gesicht klassifiziert werden kann.

Funktionsweise von LBP

Geht man wieder bei einem Bild von einer Matrix aus (s. Kapitel OpenCV), so lässt sich eine Tabelle, wie in folgender Abbildung aufbauen:

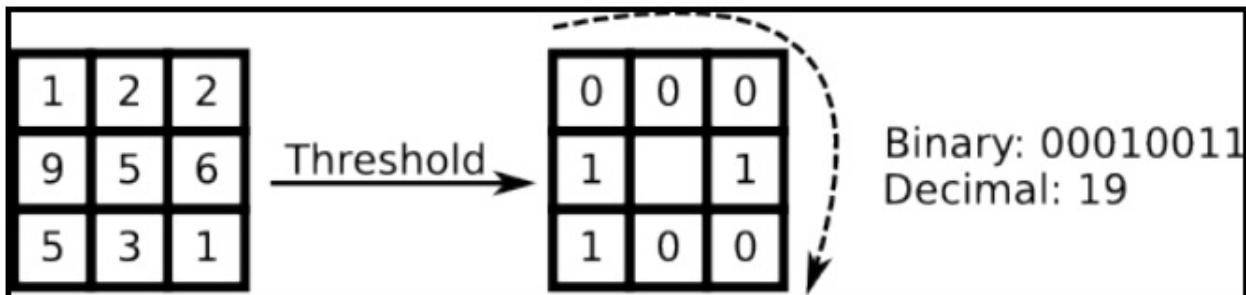


Abbildung einer Matrix[1.6]

Hier handelt es sich um ein Bild mit einer 3x3 Matrix (stark vereinfacht). LBP gibt es dabei in verschiedenen Varianten.

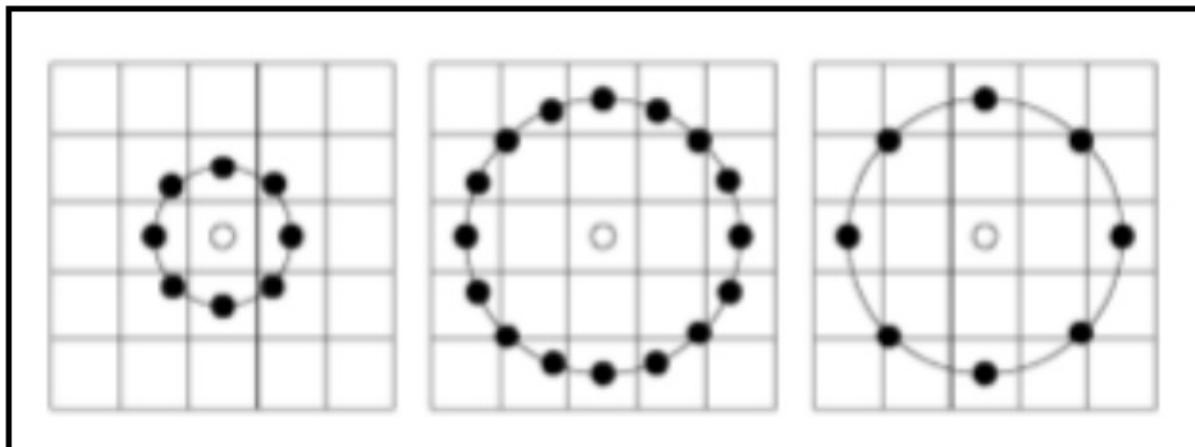
Die erste Variante (siehe folgende Abbildung links) und die übliche Standardvariante ist es, dass man den Intensitätswert eines Mittelpunktes als Basis nimmt und anschließend für alle Nachbarpixel einen Wert zwischen 0 und 1 festlegt. 1 ist es dann, wenn der Nachbarpunkt größer ist als der Wert des Mittelpunktes. 0 dann, wenn es kleiner oder gleich ist.[1.6]

Anschließend werden in einer beliebigen Reihenfolge im Uhrzeigersinn die Nachbarpunkte in eine Zahlenreihe zusammengestellt und im Binärformat aufgeschrieben. Aus dem Binärformat wird dann die Dezimalzahl ermittelt und als neuer Intensitätswert für den Mittelpunkt festgelegt. Dieses Verfahren wiederholt sich für jeden Pixel im Bild. Wichtig ist hierbei, dass man die einmal zuvor beliebig festgelegte Reihenfolge für jeden Pixel in derselben Reihenfolge wiederholt. Diese Variante nennt man auch LBP 8,1.[1.6]



LBP Anwendung als Beispiel - oben sieht man das Original, unten nach Anwendung von LBP[1,6]

Die zweite Variante ist ähnlich wie die Erste. Mit dem Unterschied, dass hier statt den direkten Nachbarn die einen weiter außen liegenden Nachbarn verwendet werden (siehe Mitte der folgenden Abbildung). Dann hat man 8 statt 16 Nachbarn und daraus ergibt sich der Name LBP 16,2.[1,6]



Illustrierung von LBP Varianten[1,6]

Die zweite Variante verbraucht natürlich doppelt soviel Rechenzeit für jedes Pixel, wie die Erste. Man kann dies auch verkürzen indem man nur jedes 2. Nachbarpixel verwendet. Dies wäre dann im Namen LBP 8,2. Auffallend beim Namen ist, dass die erste Zahl der Anzahl der Nachbarpixel und die zweite Zahl die Zahl des Nachbarrings entspricht.[1,6]

Weiterhin gibt es noch eine Variante von LBP, die wichtig für die Erstellung von LBP Histogramms ist (und damit auch wichtig für die Gesichtserkennung). Es ist bewiesen worden, dass nur bestimmte uniform Pattern (auf deutsch bekannte Muster) für LBP Histogramms sind.[1,6]

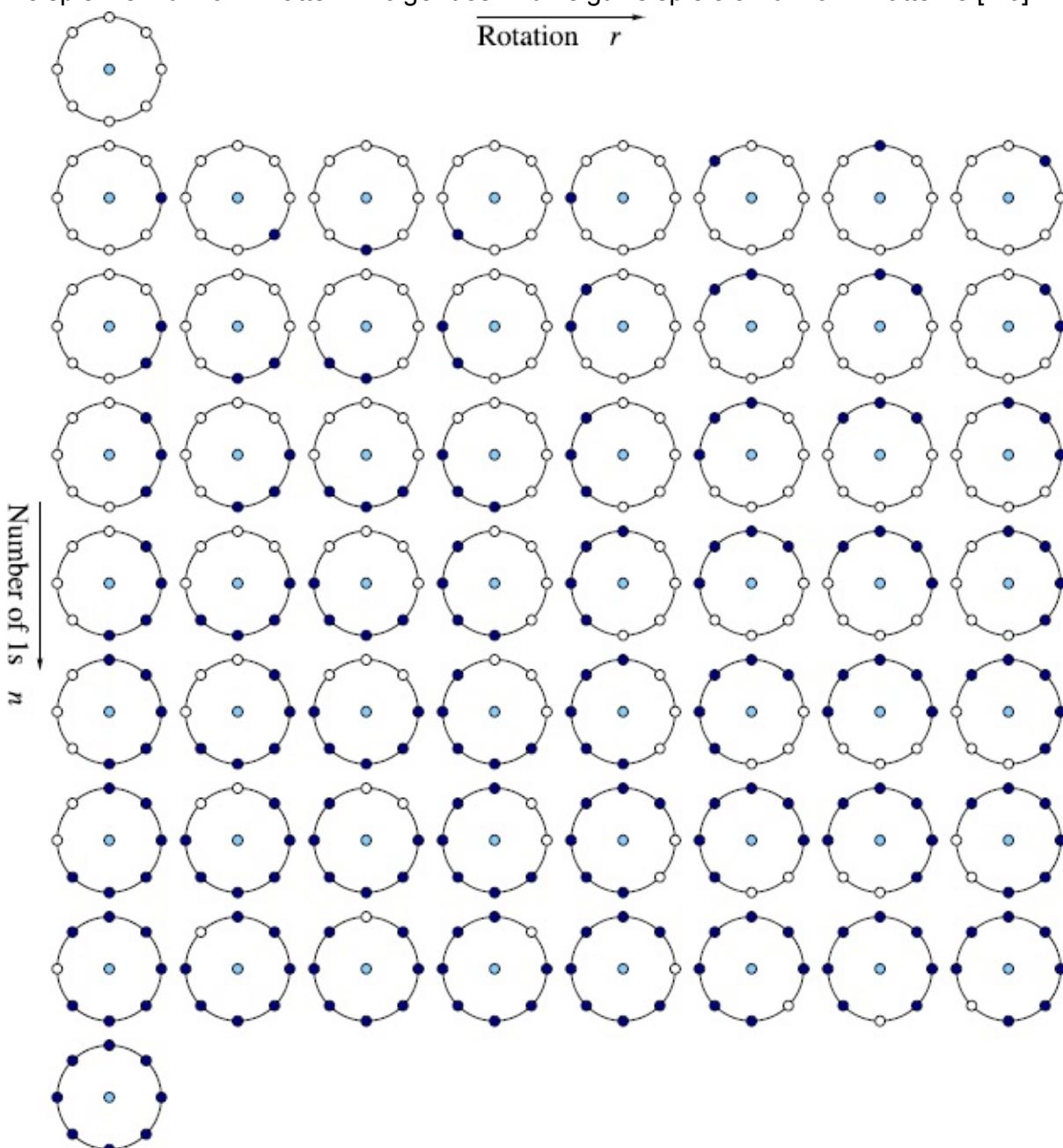
Um ein uniform Pattern zu erkennen braucht man eine binäre Zahl (hier werden 8 Bits als Beispiel genommen) und geht sie von links nach rechts durch. Dabei werden Bittransitionen gezählt. Eine Bittransition ist dann vorhanden, wenn die Zahl sich von 0 auf 1 verändert und umgekehrt genauso. Wichtig hierbei ist zu beachten, dass auch der letzte Bit mit den ersten Bit verglichen wird.[1.6]

Ein Beispiel wäre: 00011101

Folgende Transitionen sind vorhanden:

- Vom 3. zum 4. Bit 0 -> 1
- Vom 6. zum 7. Bit 1 -> 0
- Vom 7. zum 8. Bit 0 -> 1
- Vom 8. zum 1. Bit 1 -> 0

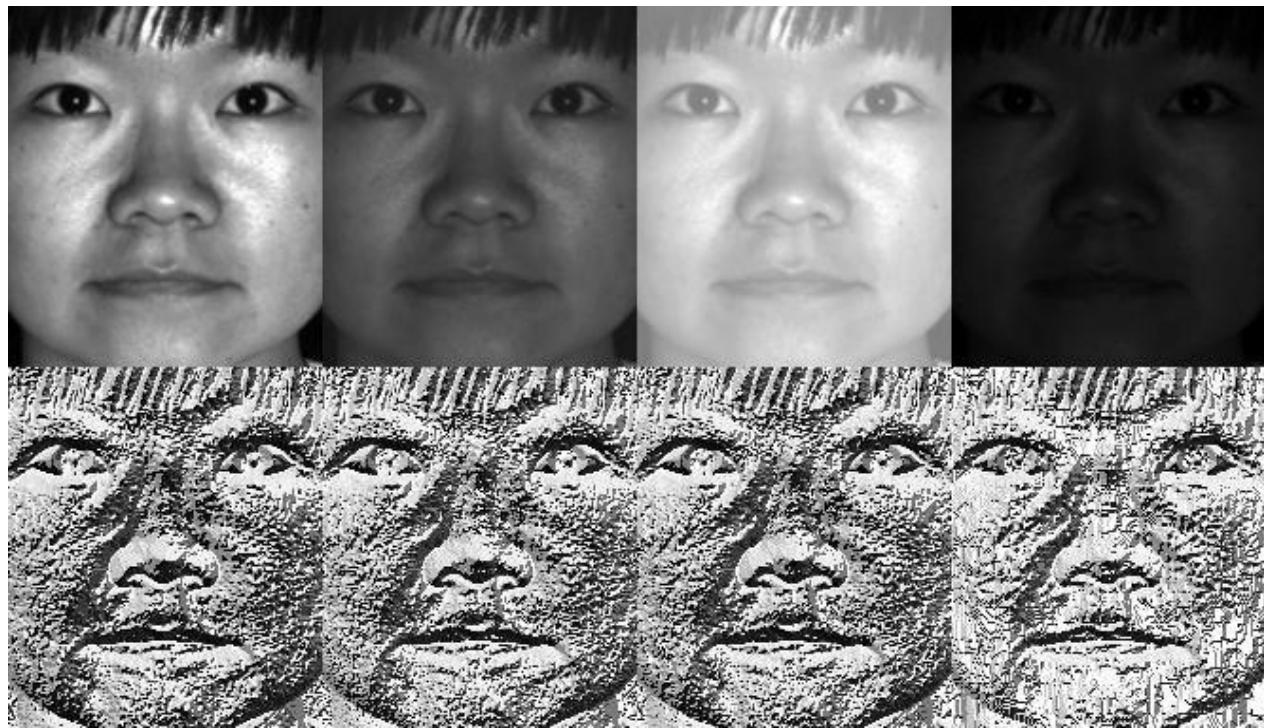
Ein uniform Pattern darf allerdings nur maximal 2 Bittransitionen haben. Damit wäre unser Beispiel kein uniform Pattern. Folgendes Bild zeigt Beispiele an uniform Patterns.[1.6]



Beispiel für ein uniform Pattern.[13]

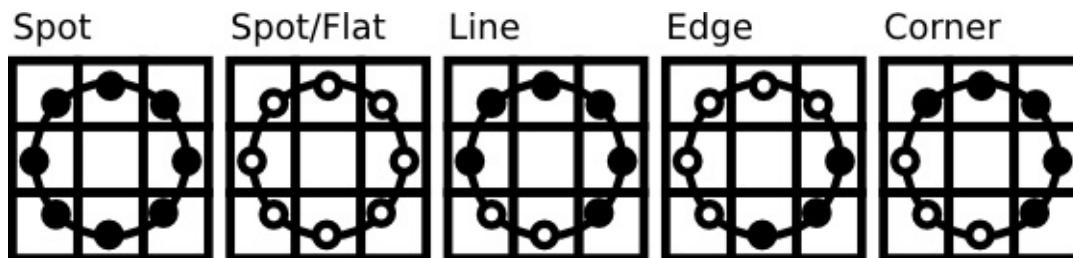
Damit ist gezeigt, dass es bei einem 8 Bit Binärkode es insgesamt 58 uniform Patterns gibt. Diese werden auch "uniform pattern LBP histogram" genannt.[1.6]

Uniform Patterns führen zu einer stärkeren Effizienz und kompakteren Repräsentierung des LBPH. Nun muss noch betrachtet werden, was LBPH überhaupt über das Bild erzählt. [1.6]



Beispiel eines verschieden beleuchteten Jungen in LBP.[14]

Betrachten wir den Jungen aus dem Beispieldfoto oben. Auffallend hierbei ist, dass LBP besonders dort Konturen zeichnet, wo eine Intensitätsveränderung im Bild zu sehen ist. Wenn man jetzt sich vorstellen möchte, wo die Uniform Patterns auftauchen werden. Dann kommt man zum Endschluss, dass sie dort auftauchen, wo mindestens 6 Nachbarpixel entweder heller oder dunkler sind. Diese Pixel werden als gesondert betrachtet und sind vergleichbar mit einer Kante, einer Linie usw. im Bild. (Siehe nachfolgende Abbildung.) Auch ist ein klarer Vorteil von LBPH, dass das Bild überbelichtet oder unterbelichtet sein kann, dennoch wird es in der Regel zum selben LBP Ergebnis führen. [1.6]



LBPH übertragen auf herkömmliche Bildmuster.[15]

Betrachtet man den Jungen im Kontext zur Erkennung des Gesichts, so wird einen schnell klar, dass ein Gesicht fast immer übereinstimmende uniform Patterns aufweisen wird. Egal wie beleuchtet das Gesicht ist. Anhand eines Trainingssets ist es dann möglich, das Gesicht aus unterschiedlichen Perspektiven zu erkennen. Damit ist LBPH eine gute Methode zur Gesichtserkennung. [1.6]

LBPH in OpenCV

Die LBPH Funktion findet sich wie die Fisher- und Eigenfaces im OpenCV Contrib. Anzumerken ist nur, dass LBPH das Extended Local Binary Pattern nutzt und man die Zahl der Nachbarpixel, den zu betrachtenden Radius und den zu betrachtenden Grid im Bild selbst einstellen kann. (Dies ist aber optional!) Auch kann man sich die model infos vom LBPH Recognizer nochmal hinterher per Funktion anzeigen lassen. Confidence gibt es dann per kNN, wie auch bei den anderen Gesichtserkennungen. [16]

Fazit

Abschließend kann festgestellt werden, dass es mehrere Möglichkeiten zur Gesichtserkennung gibt. Es wurde aufgezeigt, dass es für die Gesichtserkennung wichtig ist, das Gesicht zu erkennen und anschließend zu klassifizieren. Wie man vorgeht, um das Ziel zu erreichen, ist jedem selbst überlassen. Es wurde die Möglichkeit der Haarcascade (um Gesichter zu erkennen) veranschaulicht, welche bei OpenCV der Standard ist, aber natürlich gibt es auch noch andere Möglichkeiten.

Um Gesichter zu klassifizieren, kann man Eigenfaces, Fisherfaces oder LBPH einsetzen. Dabei muss man natürlich die Pros- und Contras abwägen. Möchte man eine möglich effiziente Gesichtserkennung oder hat man über- oder unterbelichtete Bilder, so eignet sich LBPH am besten. Bei LBPH muss berücksichtigt werden, dass Genauigkeit und Effizienz von der Konfiguration abhängen. Möchte man eine möglichst genaue Gesichtserkennung, dann sollt man eher Eigenfaces einsetzen. Hat man dabei nur wenig Bilder einer Person im Trainingsset, sollt man eher Fisherfaces statt Eigenfaces verwenden.

Quellen

- [1] Datta, Samyak. "Learning OpenCV3 Application Development" Packt Publishing Ltd. ISBN 978-1-78439-145-3 (Dec. 2016)
 - [1.1] vgl. [1] S.7-12 Kapitel 1 "Laying the Foundation"
 - [1.2] vgl. [1] S.135-137 Kapitel 5 "Image Derivatives and Edge Detection"
 - [1.3] vgl. [1] S.150-165 Kapitel 5 "Image Derivatives and Edge Detection"
 - [1.4] vgl. [1] S.169-175 Kapitel 6 "Face Detection Using OpenCV"
 - [1.5] vgl. [1] S.186 Kapitel 6 "Face Detection Using OpenCV"
 - [1.6] vgl. [1] S.226-241 Kapitel 8 "Feature Descriptors in OpenCV"
- [2] <https://www.heise.de/newsticker/meldung/de-Maiziere-Mit-Software-zur-Gesichtserkennung-nach-Terroristen-fahnden-3740349.html> (eingesehen am 29.06.2017)
- [3] http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html (eingesehen am 30.06.2017)
- [4] http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf (eingesehen am 30.06.2017)
- [5] <https://de.wikipedia.org/wiki/Eigengesichter> (eingesehen am 01.07.2017)
- [6] <http://answers.opencv.org/question/12036/confidence-value-and-threshold-in-opencv/> (eingesehen am 01.07.2017)
- [7] <https://de.wikipedia.org/wiki/N%C3%A4chste-Nachbarn-Klassifikation> (eingesehen am 01.07.2017)
- [8] <http://www.scholarpedia.org/article/Fisherfaces> (eingesehen am 01.07.2017)
- [9] http://docs.opencv.org/2.4/_images/at_database_small_sample_size.png (eingesehen am 01.07.2017)
- [10] <https://de.wikipedia.org/wiki/Hauptkomponentenanalyse> (eingesehen am 01.07.2017)
- [11] https://en.wikipedia.org/wiki/Linear_discriminant_analysis (eingesehen am 01.07.2017)
- [12] https://en.wikipedia.org/wiki/Local_binary_patterns (eingesehen am 02.07.2017)
- [13] <https://i.stack.imgur.com/AMLGu.png> (eingesehen am 02.07.2017)
- [14] http://docs.opencv.org/2.4/_images/lbp_yale.jpg (eingesehen am 02.07.2017)
- [15] http://docs.opencv.org/2.4/_images/patterns.png (eingesehen am 02.07.2017)

IOT-Effizienz und IOT-Hub-Cloud (Azure)

Autor Wladimir Streck

Einführung

Cloud-Computing ist eine moderne Form des klassischen ~~on-premises~~ Datacenters nur mit dem Unterschied, dass der Provider die Verantwortung für die Hardware, deren Wartung, die Plattformen und allem was zu einem Datencenter gehört,trägt. Dem Kunden gehört somit nichts aus dem Datencenter, sondern es besteht ein Mietvertrag zwischen dem Kunden und dem Provider. Der Kunde gibt an, zu welchem Zeitpunkt was (Leistung, Service, Software, usw.) benötigt wird und zahlt auch nur dies. Der Provider stellt die Ressourcen dann zur Verfügung. Es gibt viele Arten von Dienstgruppen im Cloud-Computing, jedoch gehören zu den drei größten: SaaS, PaaS und IaaS. PaaS (Platform as a Service) bedeutet, dass z.B. eine Laufzeitumgebung, eine Plattform oder ähnliches vom Provider zur Verfügung gestellt wird. IaaS (Infrastructure as a Service) ist ein Geschäftsmodell, welches z.B. den Server oder PC aus der Firma in die Cloud auslagert und SaaS (Software as a Service) lagert jede Form von Software in die Cloud aus.

Problemstellung

Im Embedded Computing ist Leistung, Speicher und ständige Erreichbarkeit meist sehr teuer. Es wäre sehr teuer hinter jeden Sensor einen PC oder Ein-Platinen-Computer zu hängen um Daten zu sammeln und auszuwerten. Mit der Anzahl der vernetzten Alltagsgeräte und Sensoren wächst die Datenmenge, die gesammelt und auf sogenannten Hubs oder in der Cloud gespeichert wird.

Das Verwalten der Geräte, speichern und auswerten der Daten und Steuern der Aktoren sowie auslesen der Sensoren ist zu einem Full-Time-Job geworden. Und für genau diese Anwendungsgebiete gibt es im Cloud-Computing einen besonderen Service, die IOT-Hubs. IOT-Hubs verbinden, überwachen und kontrollieren sämtliche IT-Ressourcen, die unter verschiedenen Betriebssystemen und mit verschiedenen Protokollen ausgeführt werden. IOT-Hubs ermöglichen auch die Überwachung von Sensoren, die nicht kontinuierlich erreichbar sind, sondern gegebenenfalls die Daten nur in einem Intervall senden und auch nur ein kurzes Zeitfenster als Antwort zulassen auf eventuelle Reaktion auf Telemetrie Daten. So können auch komplizierte oder zeitaufwendige Aufgaben schneller von einem Dienst der Cloud gelöst werden und belasten das Endgerät oder den Sensor nicht und ermöglichen so auch ein schlankeres Design.

About Azure

Azure ist eine Cloud-Computing Plattform. Azure bietet ein komplettes Portal für die Verwaltung der Cloud. Für die Verwaltung stellt Azure sowohl online ein Web-Interface als auch mehrere IDE's wie zum Beispiel Visual Studio zur Verfügung. Über diese Verwaltungswerzeuge kann dann ein Service der Cloud gebucht werden oder sogar eine Software/ein Service selbst programmiert und in der Cloud zur Verfügung gestellt werden. Zu den Services gehören:

- Virtual Machines
- Apps
- SQL-Datenbanken
- Storage
- Backups

Und sehr viele mehr. Eine Liste kann hier eingesehen werden:

<https://azure.microsoft.com/de-de/services/>

Ein Vorteil von Azure ist, dass jeder Service der angeboten wird mindestens eine komplette SDK mit sich bringt und meist sogar mehrere. So gibt es z.B. für Hadoop-Lösungen eine SDK der Apache Software Foundation, welche voll in Azure integriert ist und eine Hadoop SDK von Microsoft.

IOT Hub virtual Machines

Meist reicht PaaS nicht aus um alle Anwendungsfälle abzudecken. Gerade bei Arbeiten mit Hardware und externen Sensoren ist es notwendig Zugriff auf viele Komponenten des Betriebssystems zu haben. An dieser Stelle kommt die IaaS in Form einer Virtuellen Maschine zum Einsatz.

Virtual machine models

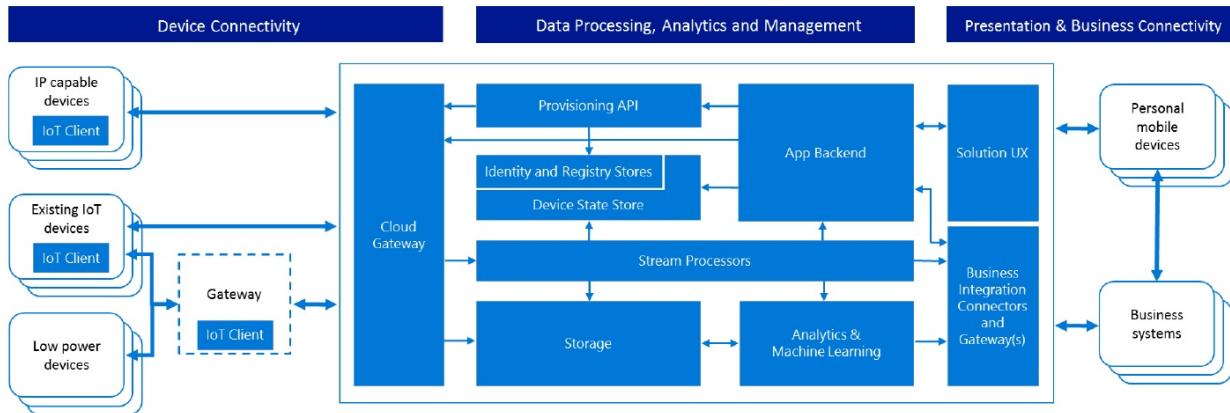
IOT Hub stellt zwei Modelle für Virtuelle Maschinen zur Verfügung, das Azure Resource Manager (ARM) und Azure Service Management (ASM). Das ARM ist das moderne von beiden und kommt deshalb am häufigsten zum Einsatz. Das ARM bietet Kontrolle über jede Funktion einer Ressource wie bei einem gewöhnlichen Desktop-Computer. So können zum Beispiel Netzwerke, Speicher, Rechenleistung und vieles mehr, hinzugefügt, entfernt und konfiguriert werden. Für das Verwalten dieser Komponenten wird ein Ressourcen Manager in der Verwaltung der Virtuellen Maschine zur Verfügung gestellt. Der Network Resource Provider behandelt alle Aspekte der Netzwerkkonnektivität wie IP-Adressen, Load Balancer und NICs. Eine Festplatte einer Virtuellen Maschine existiert nicht mehr im herkömmlichen Sinn als Hardware. Ein Speicherressourcenanbieter übernimmt die Aufgabe virtuelle Festplatten zu erzeugen und als Speicher zur Verfügung zu stellen. Dabei kann es sein, dass eine virtuelle Festplatte auf mehreren geografisch unterschiedlichen Orten über mehrere physische Festplatten aufgeteilt ist. Eine virtuelle Festplatte kann trotzdem als logisches Laufwerk verwendet werden. Der Compute Resource Provider sorgt dabei für eine angemessene dynamische Verteilung der Ressourcen um zu vermeiden, dass diese ungenutzt sind oder im idle-Betrieb sind. Auch ist die Kontrolle, Bereitstellung und Verwaltung von verwandten Ressourcen als Teil einer Ressourcengruppe möglich. Eine erweiterte Rollenbasierte Zugriffskontrolle (RBAC) ist möglich und sorgt für mehr Sicherheit. Der klassische ASM bietet hingegen eine herkömmliche Virtualisierung der Ressourcen. Alle Funktionen werden dabei in einem Container gehostet und nicht in mehreren Einzelcontainern hinterlegt.

Virtual machine components

Wie auch ein normaler Desktop-Computer besteht eine Virtuelle Mschine aus unterschiedlichen Komponenten. Dazu gehört das Betriebssystem, Prozessoren, Arbeitsspeicher, Festplatten, Netzwerke, usw. Im Vergleich mit gewöhnlichen Komponenten die als Hardware für einen Desktop-Computer zur Verfügung stehen, stellt ein jeweils virtuelles Modell einer Komponente eine dar, dessen Limit nur durch die Gesamtheit des Systems gesetzt ist. So können in einem gewöhnlichen Computer aktuell maximal 1048

TB Arbeitsspeicher verbaut und genutzt werden. Dies ist jedoch mit sehr hohen Kosten verbunden und die meisten Zeit wird der große Speicher nicht benötigt. Deshalb macht es Sinn sich den Arbeitsspeicher oder Prozessorkerne nur für einen gewissen Zeitraum zu mieten und die Ressourcen im Idle-Betrieb an andere Cloud-Benutzer abzugeben.

Storage



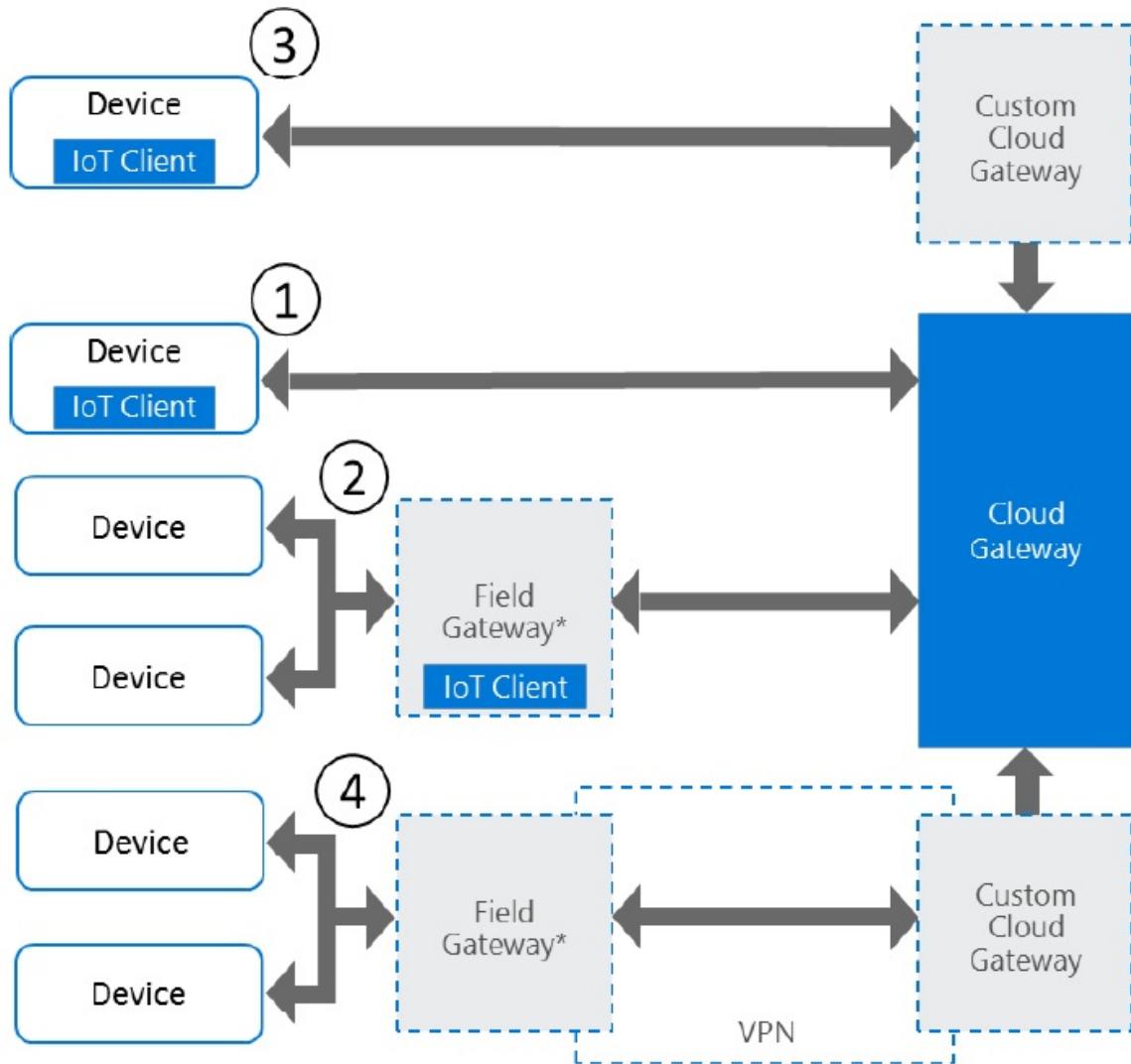
[Quelle: http://download.microsoft.com/download/A/4/D/A4DAD253-BC21-41D3-B9D9-87D2AE6F0719/Microsoft_Azure_IoT_Reference_Architecture.pdf - Seite 4]

Die Speicherfrage auf eingebetteten System wie zum Beispiel Sensoren stellt immer eine große Herausforderung, weil der Sensor diese Daten meist selbst nicht braucht und um große Datenmengen vorzuhalten ist ein Speicher auf jedem Sensor meist zu teuer. Um

IOT-Storage

Das Verständnis von Datenkonzepten ist ein wichtiger erster Schritt zum Aufbau von gerätezentrischen Datenerfassungs-, Analyse- und Steuerungssystemen. Dabei wird zwischen einem internethohen Gerät und einem nicht-internethohen Gerät unterschieden, welche sich dann jeweils in eine normale und eine Low-Power-Gruppe teilen. Die Internethohen Geräte besitzen ein Protokoll, über welches es direkt möglich ist mit dem Cloud-Gateway zu kommunizieren. Ein nicht-internethohes Gerät muss von einem Gateway unterstützt werden. Dieses Gateway kann über ein Kabel oder eine kabellose Schnittstelle wie Bluetooth mit dem Sensor kommunizieren, die Daten sammeln und an die Cloud weitergeben.

Bei diesen Daten handelt es sich um die Telemetrie der Hardware. Dabei handelt es sich meist um Punkt-Beobachtungen, die über einen definierten Zeitraum gemacht wurden, der für das Signal als geeignet erachtet wird. Diese Signale können sogar von der Hardware vorverarbeitet werden, wie zum Beispiel die Integration über einen gewissen Zeitraum, werden jedoch danach immer Kodiert zu einem für die Sequenz passenden Format. Für Audiodaten bietet sich zum Beispiel das Codec MP3 an und Wetterdaten wie Temperatur, usw. das Codec TEMPS.



[Quelle: http://download.microsoft.com/download/A/4/D/A4DAD253-BC21-41D3-B9D9-87D2AE6F0719/Microsoft_Azure_IoT_Reference_Architecture.pdf - Seite 9]

Eine Gerätebereitstellungs-API sorgt für das Erfassen neuer Geräte oder entfernen Alter. Ebenso kann die Gerätebereitstellung Sensoren aktivieren und deaktivieren. Wenn sie deaktiviert sind, haben sie keinen Zugriff auf das System, aber alle Zugriffsregeln, Schlüssel und Metadaten bleiben bestehen um eine leichtere Reaktivierung zu ermöglichen. Die Geräteidentität und Konfiguration speichert jedoch nicht das Gerät selbst sondern der Gerätebereitstellungsdiensst. Das Provisioning bietet jedoch keinem externen Gerät die Möglichkeit etwas an dem Gerätebereitstellungsdiensst zu verändern, sofern es nicht teil des Dienstes selbst ist. Auch der Status eines Gerätes wird in einem speziellen Speicher erfasst. Zu dem Status gehören nicht nur Informationen wie Online oder Offline sondern auch die Art der Kommunikation.

Ist ein Gerät zugelassen kann es über das Gateway der Cloud Daten zu einem Stream Prozessor schicken. Dieser übernimmt die Aufgaben der Analyse der Daten. Ein Stream Prozessor kann dabei mehrere Datenströme verwalten oder ein Datenstrom kann auf mehrere Stream Prozessoren aufgeteilt werden. Dies hängt von der Komplexität des

Datenstroms ab. So kann ein Stream Prozessor nur auf spezielle Arten von Ereignissen hören, während ein anderer eine komplexe Ereignisverarbeitung parallel durchführen kann. Diese Prozessoren können den Pfad von Daten und Routen ohne Umformung oder komplexe Ereignisverarbeitungsaufgaben wie Datenaggregation, Datenanreicherung durch Korrelation mit Referenzdaten sowie analytische Aufgaben wie Erkennung von Grenzwerten oder Anomalien und Erstellung von Alarmen ermitteln. Modelle für Stream Prozessoren können Raw telemetry, Bulk upload, Device state, Device metadata, Special events, Diagnostics telemetry, Hot path analytics oder Machine learning sein.

Für das Durchführen einer modellbasierten Analyse ist dann ein weiteres Modul zuständig. Dieses Modul hat die Möglichkeit Daten an den Stream Prozessor nach Verarbeitung zurück zu geben oder direkt zu speichern. So kann ein Sensordatenstrom mithilfe von Diagnosewerkzeugen zusammengefasst oder verkleinert werden und ein neuer Stream Prozessor entscheidet dann, ob weitere Schritte notwendig sind.

Ein App-Backend (Geschäftslogik) sorgt für die Verwaltung aller Module und die Weitergabe der Daten an eine Solution UX oder einen Business Integration Connector. Die Solution User Experience bietet dabei viele Möglichkeiten an. So kann eine Visualisierung der Daten über ein Webinterface, eine Mobile App oder ein Desktopprogramm erfolgen oder die Daten werden über eine API herausgegeben. Zusätzlich wird die Möglichkeit für eine Interaktion mit dem System geboten. Es kann zum Beispiel Einfluss auf einen Stream Prozessor genommen werden, wie z.B. das Ändern von Temperaturschwellwerten für die Heizungssteuerung.

Über den Business Integration Connector ist es möglich das gesamte IOT Hub in ein ERP, CRM oder LOB-System einzubetten.

Redundanz

Was passiert, wenn das Speichern von Daten fehlschlägt oder ein kompletter Speicherknoten ausfällt? Mit dieser Frage beschäftigt sich die Redundanz in der Datenhaltung des IOT Hubs. Ein großer Vorteil ist jedoch, dass es nicht mehr notwendig ist alles zu implementieren, denn der Cloud-Dienst übernimmt dies mithilfe von einfachen Konfigurationen. Dabei werden die Systeme Locally Redundant Storage (LRS), Geo-Redundant Storage (GRS), Read-Access Geo-Redundant Storage (RA GRS) und Zone-Redundant Storage angeboten. Das LRS sorgt dafür, dass zunächst mindestens drei Kopien eines Speicherkontens erstellt werden, bevor das Schreiben zugelassen wird. Ist das Speichern in einem Knoten erfolgreich gewesen, wird dies für alle anderen wiederholt. Das bedeutet, dass alle Kopien immer synchron sind. GRS beginnt wie LRS, es werden zunächst mindestens drei Kopien der aktuellen Knoten erstellt mit dem Unterschied, dass

alle Knoten asynchron an einem geografisch anderen Standort gespiegelt werden. RA GRS erweitert GRS um die Möglichkeit die Geografisch entfernten Daten zu lesen. Dies ermöglicht dem Kunden eine bessere Überwachung der Daten.

Sicherheit

Das IOT Hub bietet viele Möglichkeiten für die Sicherheit. Die Datensicherung geschieht durch Role-Based Access Control (RBAC) und Microsoft Azure Active Directory (Azure AD). Durch Rollen sollen Daten nicht für jeden sichtbar gemacht werden. Für die Übermittlung der Daten werden Protokolle wie HTTPs oder SMB verwendet. Shared access signature (SAS) bietet die Möglichkeit alle Daten, Datenbanken oder ganze Laufwerke mit asynchronen Verfahren zu verschlüsseln. Auch eine Storage Service Encryption (SSE) kann verwendet werden, wenn die Daten zusätzlich Serverintern verschlüsselt werden sollen.

Business cases

Für IOT Hubs gibt es viele Szenarien. Von einer Testumgebung bis zur Auslagerung des Smarthome ist alles dabei. Um etwas kleines schnell zu versuchen gibt es die notwendige Flexibilität, aber auch für Leistungstests gibt es die passende Umgebung.

Zusammenfassung

Das Cloud-Computing bietet mittlerweile dem IOT mittlerweile viele Möglichkeiten. Sensoren können immer schlanker gestaltet werden, denn der Speicher und die Leistung findet nicht mehr auf jedem einzelnen Gerät statt, sondern in der Cloud. Durch ein dynamisch alloziertes System ist es nicht mehr notwendig dauerhaft jede Plattform zu jeder Zeit mit voller Leistung vorhalten zu müssen.

Quellen

Barnes, Jeff. Microsoft Azure Essentials Azure Machine Learning. Microsoft Press, 2015.

Collier, Michael, and Robin Shahan. Microsoft Azure Essentials-Fundamentals of Azure. Microsoft Press, 2015.

Chappell, David. "Introducing Azure Machine Learning." A GUIDE FOR TECHNICAL PROFESSIONALS, Sponsored by Microsoft Corporation (2015).

Cusumano, Michael. "Cloud computing and SaaS as new computing platforms." Communications of the ACM 53.4 (2010).

Familiar, Bob. Microservices, IoT and Azure: Leveraging DevOps and Microservice Architecture to deliver SaaS Solutions. Apress, 2015.

Fullstack Development

Teilbereich(e): Realtime DBs, SPAs, Responsive Design, NoSQL, Mobile Computing

Projektleiter:

Projektteam: Timo Rolfsmeier, Niklas Harting, Alexander Schwietert, Tolga Aydemir, Malte Berg, Lutz Winkelmann, Fabian Lorenz, Benjamin Schmidt

Pflichtenheft: [PDF](#)

GitLab-Group: [ShareBase](#)

NoSQL

Autor: Fabian Lorenz

Einführung

Einleitung (vgl. [1 Kap. 1.4],[2 Kap. 1-2],[12 Kap. 1],[13 Kap. 1],[14 Kap. 2.18,5],[66-69])

Mit dem Einzug des Internets in das allgemeine Leben entstanden auch neue Anforderungen bezüglich Datenbanken. Das Arbeiten mit enorm großen Datenmengen brachte viele Probleme mit sich, die meist von klassischen relationalen Datenbanken nicht gelöst werden konnten. Es entstanden daher viele neue Datenbanken, die sich in ihren Eigenschaften stark von den herkömmlichen relationalen Datenbanken abheben, um diese Probleme zu lösen.

Diese Datenbanken werden heutzutage unter dem Begriff NoSQL Datenbanken zusammengefasst. Während dieser Begriff zwar schon 1998 genutzt wurde, um eine relationale Datenbank ohne SQL Zugriffsmöglichkeiten zu beschreiben ("No SQL"), wird der Begriff seit 2009 generell als "Not only SQL" verstanden. NoSQL Datenbanken zeichnen sich häufig durch folgende Eigenschaften aus:

- Das zugrundeliegende Datenmodell ist nicht relational.
- Sie verteilen ihre Daten auf viele verschiedene Serverknoten und Knoten lassen sich zwecks Skalierung leicht hinzufügen/entfernen.
- Es gibt nur eine schwache oder gar keine Schema-Restriktion.
- Sehr einfache Replikation von Daten.
- Es gibt eine einfache API.
- Es wird nicht das klassische ACID Transaktionskonzept genutzt, sondern beispielsweise BASE.

NoSQL Datenbanken bieten meist eine höhere Leistungsfähigkeit als relationale Datenbanken, sind durch schemafreie Datenstrukturen flexibler und dank verteilter und redundanter Datenhaltung ausfallsicherer.

In diesem Kapitel werden zunächst die Grundlegenden Begriff und Technologien beschrieben, die von NoSQL Datenbanken meist verwendet werden. Anschließend werden die wichtigsten verschiedenen NoSQL Arten beschrieben.

ACID, BASE und das CAP-Theorem

ACID (vgl. [1 Kap. 4.2.1],[2 Kap. 2.2.1],[12 Kap. 1.3],[13 Kap. 1],[14 Kap. 2.6],[15-23])

Ein wichtiger Aspekt einer jeden Datenbank ist ihr Transaktionskonzept. Viele der altbekannten relationalen Datenbanken setzen hier auf das ACID Prinzip um die Integrität der Daten zu gewährleisten. Eine Transaktion ist dabei beliebige Folge zusammenhängender Verarbeitungsschritte aus den Grundoperationen Create, Read, Update und Delete (CRUD). Damit mehrere Nutzer mit einer Datenbank arbeiten können ohne dabei Konflikte in den Daten hervorzurufen, müssen die Transaktionen bestimmten Regeln folgen. Im Falle von ACID müssen sie daher folgende Eigenschaften einhalten:

- Atomarität (A=Atomicity): Eine Transaktion ist atomar, wenn sie in ihrer Gesamtheit entweder ganz oder gar nicht ausgeführt wird (Alles oder nichts Prinzip). Folglich müssen die einzelnen Operationen einer Transaktion komplett und ohne Fehler durchlaufen um einen neuen Ergebniszustand zu erreichen. Tritt jedoch ein Fehler während der Transaktion auf, wird der Zustand der Datenbank auf den Anfangszustand zurückgesetzt, den sie zu Beginn der Transaktion hatte. Alle Operationen der fehlerhaften Transaktion werden somit ungültig und bereits durchgeführte rückgängig gemacht.
- Konsistenz (C=Consistency): Eine Transaktion muss die Datenbank von einem konsistenten Anfangszustand in einen konsistenten Endzustand überführen. Um Konsistent zu sein, muss die Integrität und Plausibilität der Daten gewährleistet sein, beispielsweise müssen Beziehungen über Fremd- und Primärschlüssel korrekt sein. Kann kein konsistenter Zustand nach dem Ausführen der Transaktion hergestellt werden, so wird die gesamte Transaktion als ungültig deklariert. Die Datenbank wird dann auf den Zustand vor Beginn der Transaktion zurückgesetzt.
- Isolation (I=Isolation): Isolation sagt aus, dass Transaktionen vollkommen getrennt voneinander ablaufen. Gleichzeitig ablaufende Transaktionen liefern somit dieselben Resultate, so als würden sie sequentiell hintereinander ausgeführt werden. Es soll damit verhindert werden, dass Transaktionen sich gegenseitig beeinflussen und beispielsweise nicht zwei gleichzeitig denselben Datenwert ändern.
- Dauerhaftigkeit (D=Durability): Jede erfolgreiche Transaktion setzt die Datenbank in einen dauerhaften Zustand, der sich erst ändert, wenn eine neue Transaktion durchgeführt wurde. Diese Dauerhaftigkeit muss auch bei Systemabstürzen gewährleistet sein.

Es ist ersichtlich, dass die Einhaltung des ACID Prinzips vor allem die Konsistenz der Daten zum Ziel hat. In der heutigen Zeit spielen bei verteilten Datenbanksystemen allerdings noch zwei weitere Aspekte eine Rolle: Verfügbarkeit (Antwortzeiten) und Ausfalltoleranz.

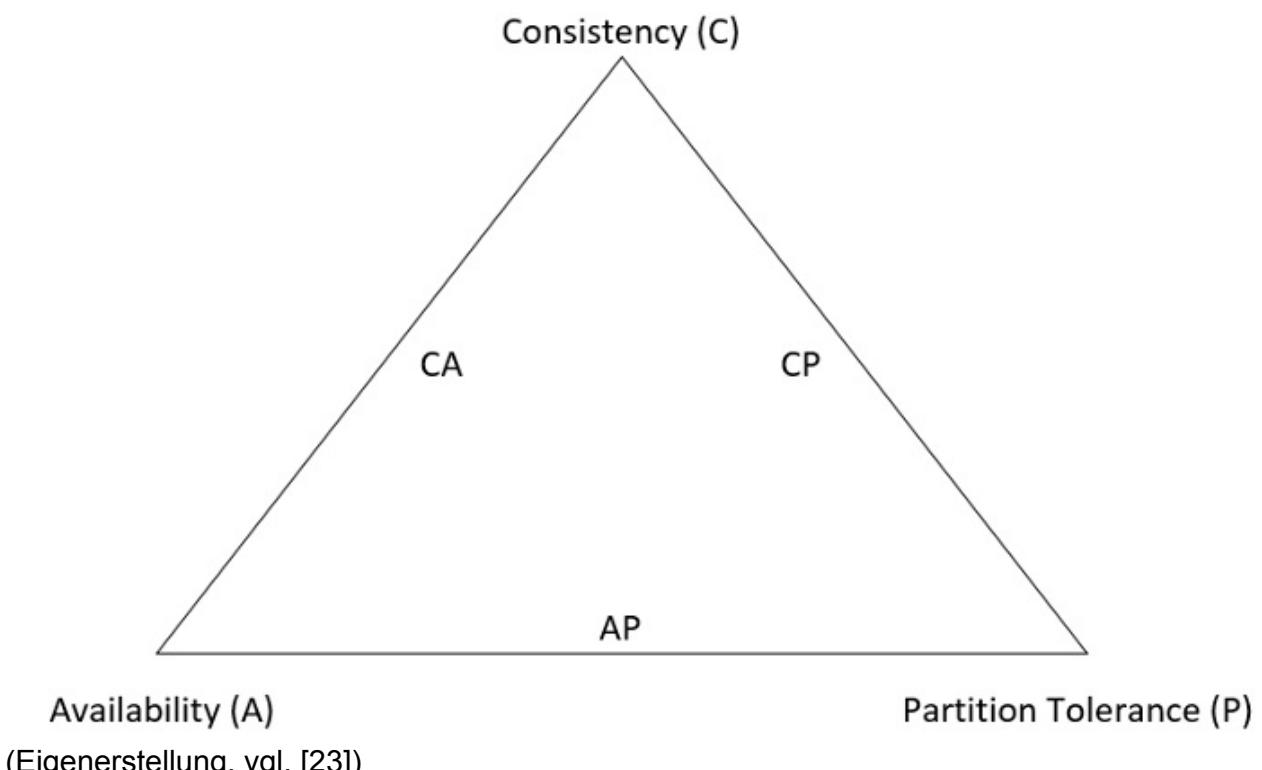
CAP (vgl. [1 Kap. 4.3.1],[2 Kap. 2.2.2],[12 Kap. 1.5],[13 Kap. 1],[14 Kap. 5.3],[23-28])

- Konsistenz (C=Consistency): Bei verteilten Datenbanksystemen mit replizierten Knoten

muss für die Konsistenz sichergestellt werden, dass bei Änderung der Daten auf einem Knoten alle folgenden Transaktionen auch mit diesen veränderten Werten arbeiten, selbst wenn sie mit einem anderen Knoten arbeiten.

- Verfügbarkeit (A=Availability): Verfügbarkeit bezeichnet die Fähigkeit ununterbrochen auf die Datenbank in angemessener Zeit zugreifen zu können.
- Ausfalltoleranz (P=Partition Tolerance): Ausfalltoleranz besagt, dass der Ausfall eines (oder mehrerer) Knoten eines verteilten Datenbanksystems, oder der Ausfall von Verbindungen zwischen den Knoten nicht den Ausfall des Gesamtsystems zur Folge hat.

Optimal wäre es, wenn alle drei Aspekte, Konsistenz, Verfügbarkeit und Ausfalltoleranz gleichzeitig im vollen Umfang erreicht werden könnten. Es hat sich jedoch gezeigt, dass dies innerhalb von verteilten Datenbanksystemen nicht möglich ist und sich immer nur zwei der drei Aspekte umsetzen lassen. Dieses Theorem wird dabei als CAP-Theorem bezeichnet. In der folgenden Abbildung ist das CAP-Theorem und die Tatsache, dass immer nur zwei Aspekte umgesetzt werden können, als Dreieck dargestellt.



Je nachdem welche zwei der drei Anforderungen an die Datenbank gestellt werden, werden diese Systeme als CA (Konsistenz und Verfügbarkeit), CP (Konsistenz und Ausfalltoleranz) oder AP (Verfügbarkeit und Ausfalltoleranz) Systeme bezeichnet. Da traditionelle relationale Datenbanksysteme wie bereits erwähnt meist die Konsistenz als oberstes Ziel haben, lassen sie sich als CA/CP Systeme bezeichnen. Viele NoSQL Datenbanksystem haben jedoch

einen viel stärkeren Fokus auf die Verfügbarkeit oder Ausfalltoleranz, die Konsistenz spielt dabei keine so große Rolle. Für diese Systeme ist es ausreichend, wenn die Konsistenz der Daten nach einer Änderung erst im Laufe der Zeit wiederhergestellt ist.

BASE (vgl. [1 Kap. 4],[2 Kap. 2.2.3],[12 Kap. 1.6],[13 Kap. 1], [14 Kap. 5.4],[23,25,29-30])

Diese Eigenschaft wird dabei auch häufig als BASE Modell (Basically Available, Soft State, Eventually Consistent) bezeichnet. Bei BASE wird die Verfügbarkeit über die Konsistenz gestellt und es gilt des Öfteren als Gegenstück zu ACID.

Die Verfügbarkeit wird dabei beispielsweise durch eine große Anzahl an Knoten erreicht (Basically Available).

Eine Änderung der Daten in einem Knoten hat dabei nicht zur Folge, dass die Änderungen direkt allen anderen Knoten mitgeteilt wird, sondern nur schrittweise. Es kann somit passieren, dass zwei Nutzer beim Lesen derselben Daten zwei unterschiedliche Datenwerte erhalten (Soft State).

Irgendwann haben jedoch alle Knoten die Änderungen verarbeitet und die Datensätze sind wieder konsistent (Eventually Consistent).

Dieses Konsistenzmodell wird von vielen NoSQL Datenbanksystemen genutzt.

Skalierung (vgl. [2 Kap. 2.2],[3-7])

Unter Skalieren versteht man die Eigenschaft eines Systems, sich an verändernde (meist wachsende) Ansprüche anzupassen. Im Bereich der Datenbanken kann beispielsweise der verfügbare Speicherplatz oder die Zugriffsgeschwindigkeit auf Daten eine Rolle spielen.

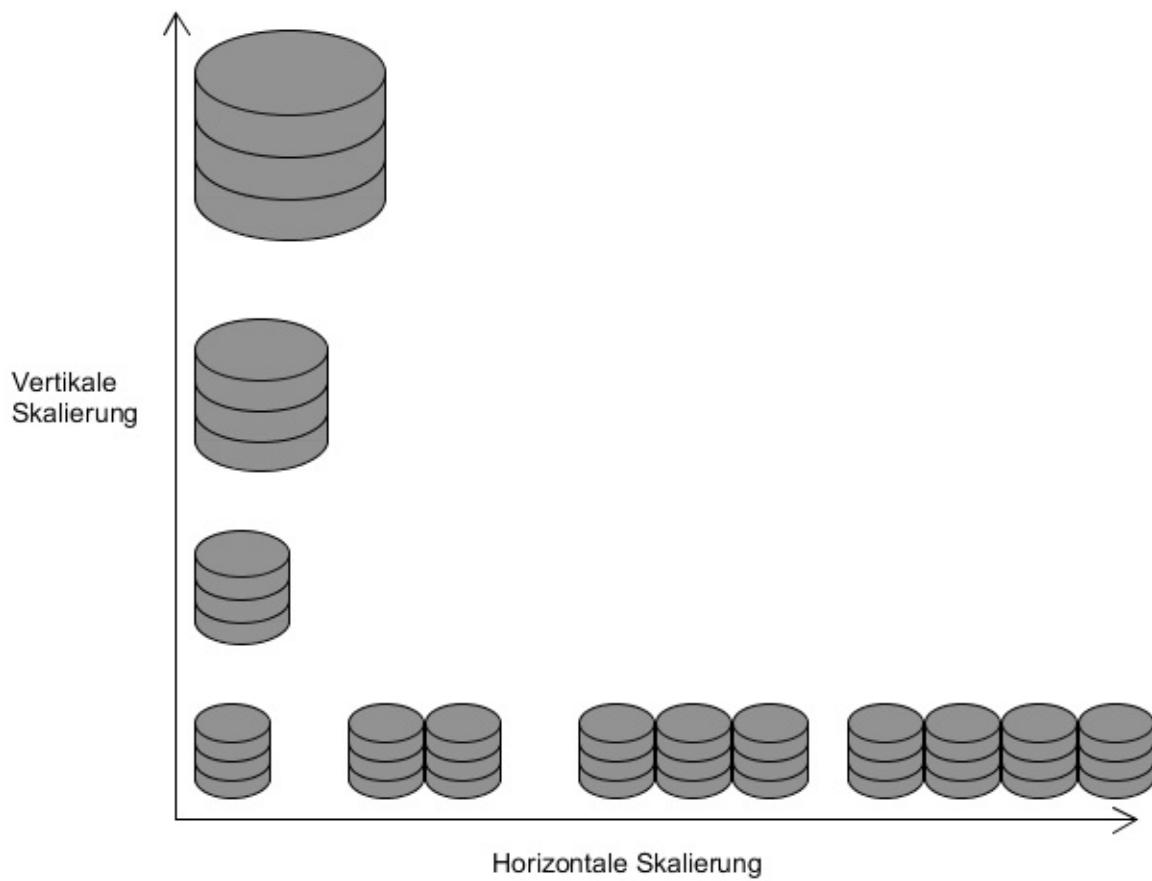
Eine Verbesserung eines Systems kann dabei auf zwei verschiedene Arten erfolgen:

- Vertikale Skalierung (Scale-Up)

Von einer vertikalen Skalierung spricht man, wenn das vorhandene System durch bessere Hardware/Komponenten aufgerüstet wird. Dies kann beispielsweise durch Hinzufügen einer CPU oder das Austauschen der alten durch eine neue erfolgen, um die Leistung des Systems zu erhöhen. Vergrößerung des Speicherplatzes kann durch Hinzufügen einer weiteren Festplatte erfolgen.

- Horizontale Skalierung (Scale-out)

Von horizontaler Skalierung ist die Rede, wenn das vorhandene System durch Hinzufügen neuer Ressourcen Einheiten (Rechner/Server) erweitert wird.



(Eigenerstellung, vgl. [6])

In der Abbildung ist das Prinzip der vertikalen und horizontalen Skalierung anhand des Beispiels eines Datenbanksystems dargestellt. Zu Beginn existiert ein Server, bei der vertikalen Skalierung wird dieser durch bessere Komponenten aufgerüstet, bei der horizontalen Skalierung werden weitere Server hinzugefügt.

Der Nachteil bei einer vertikalen Skalierung besteht darin, dass sie irgendwann an ihre Grenzen stößt. Wenn bereits die besten Komponenten, die auf dem Markt zu finden sind, verwendet werden, kann keine Verbesserung mehr erfolgen. Der Vorteil ist jedoch, dass meist keine Änderungen beispielsweise an der Anwendungssoftware durchgeführt werden müssen, da die Struktur des Systems gleich bleibt. Bei horizontaler Skalierung können Änderungen jedoch von Nöten sein, wenn Software stark an die bisherige Struktur angepasst ist. Viele NoSQL-Datenbanken haben dieses Problem jedoch nicht, da sie auf diese Art der Skalierung ausgelegt sind und Mechanismen dafür bieten.

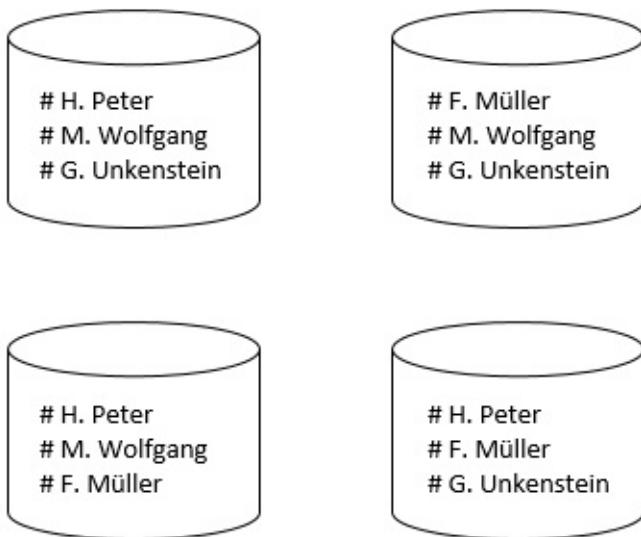
Clustering, Replikation, Sharding und Hashing

Clustering (vgl. [1 Kap. 7.2],[2 Kap. 2],[12 Kap. 1],[44-49])

Das Zusammenfassen mehrerer einzelnen Rechner/Server zu einem System wird im Allgemeinen als Clustering bezeichnet. Je nach Einsatzgebiet und genutzter Technologie kann dieses System dabei einen unterschiedlichen Aufbau haben. Beispielsweise kann es einen Master und mehrere Slaves geben, oder es können mehrere gleichberechtigte Knoten existieren. Im Allgemeinen ist der Vorteil eines Clusters jedoch, dass eine Lastverteilung stattfinden kann. Ist ein Knoten stark ausgelastet, können seine Aufgaben an andere Knoten im System verteilt werden. Auch können Aufgaben, die an das System gestellt werden, direkt an den Knoten geleitet werden, der zurzeit am wenigsten ausgelastet ist. Ein weiterer Vorteil besteht meist auch in einer verbesserten Ausfallsicherheit. Fällt ein Knoten aus, können die anderen Knoten des Systems diesen Ausfall kompensieren.

Replikation (vgl. [1 Kap. 6.2],[2 Kap. 2],[13 Kap. 3],[14 Kap. 3.11],[44-47,50-52])

Als Replikation wird das Speichern gleicher Datenbestände auf mehrere Knoten im System bezeichnet. Auch hier kann es je nach Datenbankmanagementsystem unterschiedliche Anwendungen dieses Prinzips geben. Alle Knoten des Systems können beispielsweise dieselben Daten besitzen und jede Änderung untereinander aktualisieren. Auch ist es möglich, dass es einen Master gibt, der für alle Schreibzugriffe zuständig ist und seine Datenbestände an die Slaves weitergibt. Leseanfragen können dabei über die Slaves abgearbeitet werden. Eine weitere Möglichkeit ist, dass nicht jeder Knoten dieselben Daten hält, sondern jeder hält immer nur einen bestimmten Teil der Daten, sodass am Ende ein bestimmter Datensatz zwar des Öfteren im Gesamtsystem vorkommt, aber nicht auf jedem einzelnen Knoten. Häufig wird dabei mit einer Replikationsrate von drei oder fünf gearbeitet, soll heißen ein bestimmter Datensatz ist auf drei, bzw. fünf verschiedenen Knoten im Clustersystem vorhanden. In der folgenden Abbildung ist ein kleines Beispiel mit einer Replikationsrate von drei dargestellt.

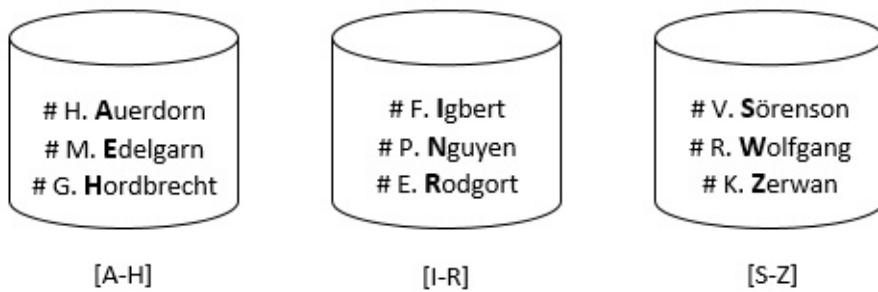


(Eigenerstellung, vgl. [45])

Vorteile der Replikation bestehen in der verbesserten Ausfallsicherheit und kürzeren Antwortzeiten. Fällt ein Knoten aus kann dennoch auf die betroffenen Datensätze zugegriffen werden, da diese auch von anderen Knoten gespeichert werden. Durch die Verteilung der Daten an verschiedene Orte kann eine bessere Lastverteilung stattfinden und dementsprechend die Antwortzeiten verringert werden. Probleme bestehen jedoch immer bei der Konsistenzbewahrung. Änderungen auf einem Knoten müssen auf alle anderen betroffenen Knoten übertragen werden.

Sharding (vgl. [1 Kap. 6.2],[14 Kap. 5.8],[44-47,53-55])

Beim Sharding werden Datenbestände möglichst gleichmäßig aufgeteilt und auf verschiedene Knoten im Cluster verteilt. Ein Stück eines solchen geteilten Datenbestands wird dabei als Shard (Bruchstück/Splitter) bezeichnet. Ein einfaches Beispiel dafür ist in der folgenden Abbildung dargestellt. Es werden die Namen anhand der alphabetischen Reihenfolge sortiert und auf drei Knoten aufgeteilt.



(Eigenerstellung, vgl. [45])

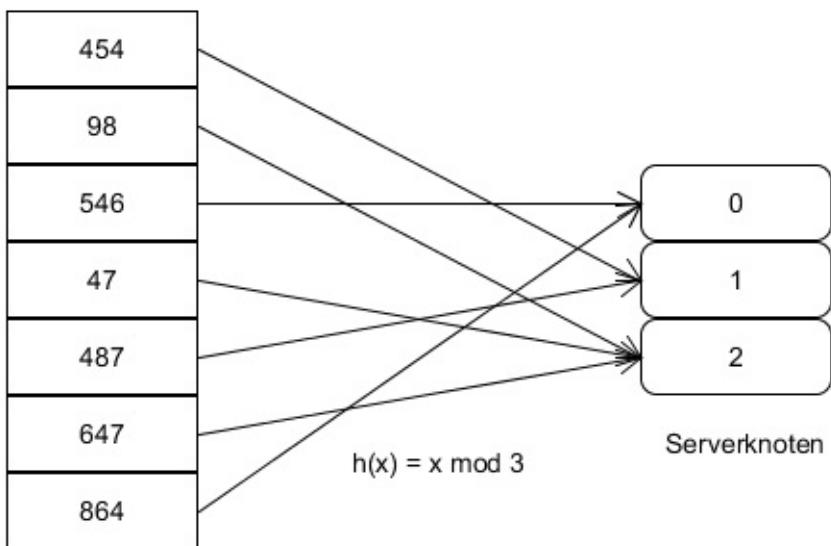
Durch Sharding ist es möglich auch sehr große und umfangreiche Datenmengen zu verwalten. Datenmengen, die die Kapazität eines Servers übersteigen würden, können auf mehrere aufgeteilt werden. Dadurch kann auch die Last verteilt werden und sich somit die Antwortzeiten verbessern. Der große Nachteil ist jedoch, dass beim Ausfall eines Shards auch nicht mehr auf die von ihm verwalteten Daten zugegriffen werden kann.

Dementsprechend wird Sharding sehr häufig mit Replikation zusammen verwendet. Gleiche Shards werden dabei wieder auf verschiedene Knoten verteilt, sodass beim Ausfall eines Knoten auf das Shard auf einem anderen Knoten zurückgegriffen werden kann. Sehr viele NoSQL Datenbanksysteme bieten automatische Mechanismen an, die die Verwendung von Sharding und Replikation stark vereinfachen. Datensätze werden bei Bedarf automatisch umverteilt, beispielsweise wenn ein neuer Knoten hinzugefügt wird oder die Datensätze nicht mehr gleichmäßig aufgeteilt sind.

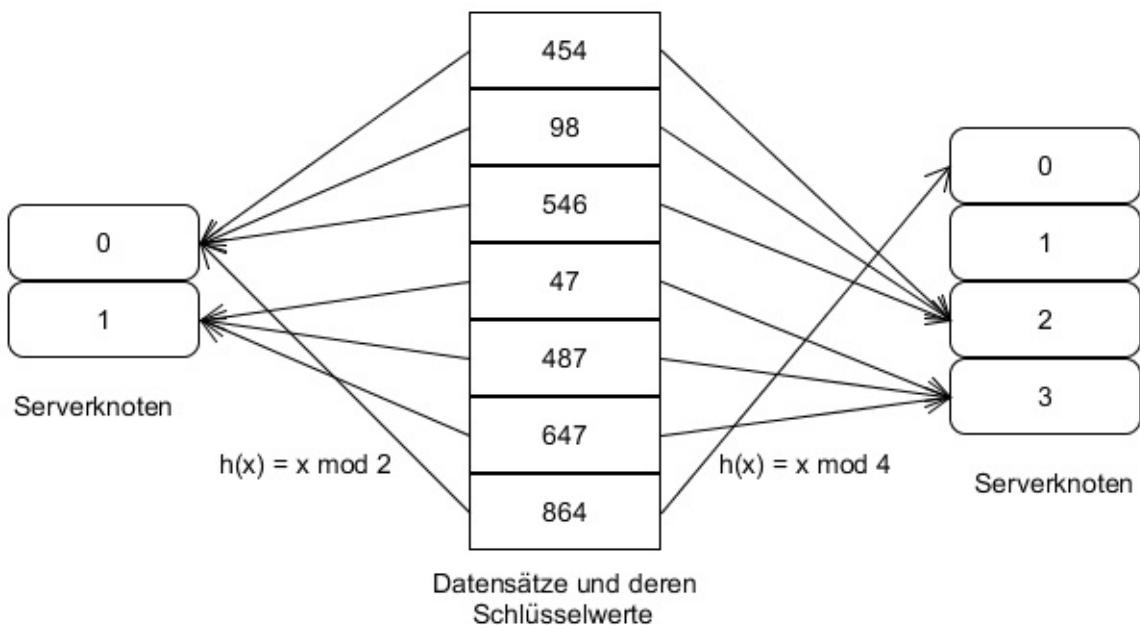
Hashing (vgl. [1 Kap. 5.2.2,5.2.3],[2 Kap. 2.3],[12 Kap. 2],[44-47,56-65])

Die verwendete Methode, um Datensätze in verteilten Datenbanksystemen auf mehrere Knoten aufzuteilen, bzw. zu speichern, ist dabei das Hashing. Beim Hashing wird durch eine Funktion ein Wert aus einem großen Wertebereich auf einen Wert aus einem kleineren Wertebereich abgebildet. Dabei wird meist eine möglichst gleichmäßige Verteilung angestrebt.

Eine einfache Hashfunktion, die für die Verteilung von Daten auf mehrere Datenbankknoten verwendet werden kann, ist dabei die Modulo Funktion $H(x) = x \bmod n$. Der Wert x ist dabei der Datensatz der auf einen der verfügbaren Knoten verteilt werden soll, bzw. ein entsprechender Schlüsselwert der den Datensatz identifiziert. Der Wert n ist die Anzahl der verfügbaren Knoten. Durch diese einfache Hashfunktion können alle Datensätze auf die zur Verfügung stehenden Knoten aufgeteilt werden. Problematisch ist es jedoch wenn ein Serverknoten ausfällt oder ein neuer hinzugefügt werden soll. Es muss für jeden Datensatz neu berechnet werden, auf welchen Knoten im Cluster er gespeichert werden soll und häufig müssen sehr viele Datensätze auf einen anderen Knoten als vorher verschoben werden. Dies wird in den folgenden Abbildungen deutlich. Die erste zeigt die Verteilung bei drei Serverknoten, die zweite die Verteilung wenn ein Knoten ausfällt oder dazukommt.



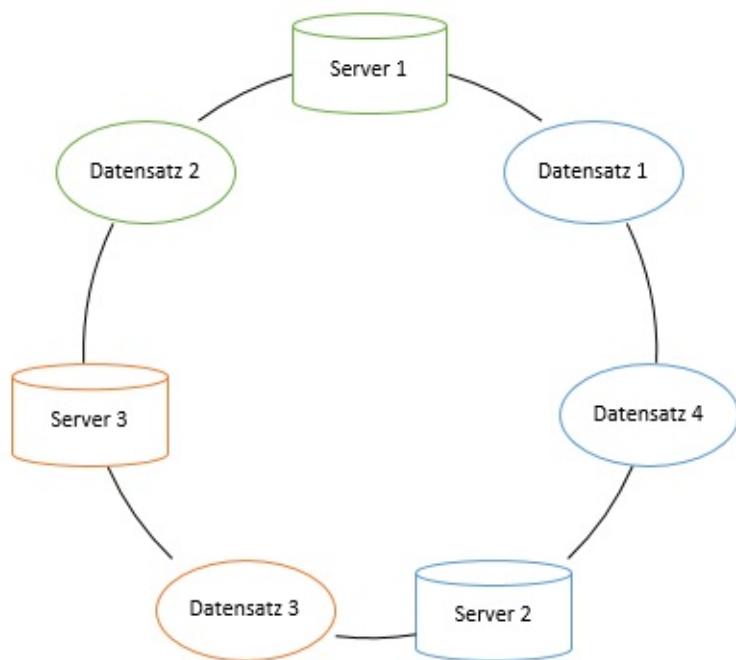
Datensätze und deren
Schlüsselwerte



(Eigenerstellungen, vgl. [2 Kap. 2.3])

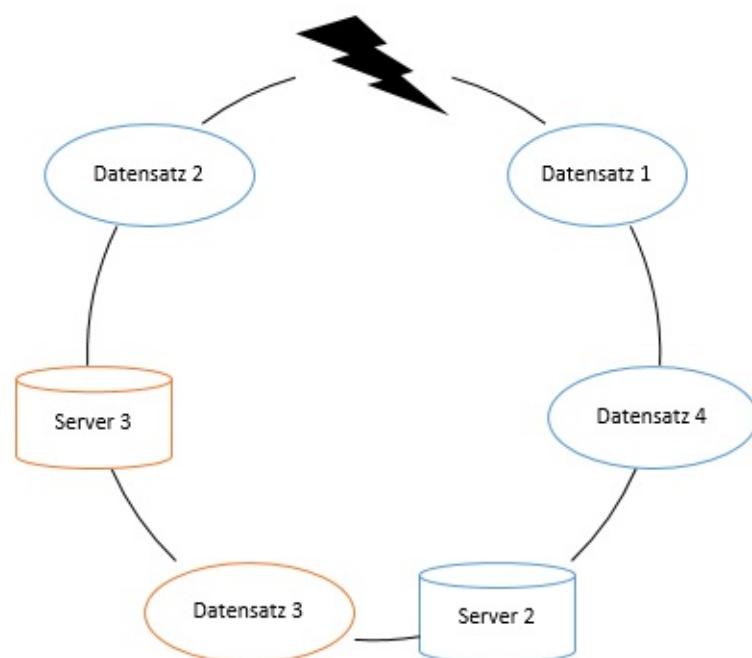
Der erste Datensatz wird bei drei Knoten auf dem zweiten gespeichert. Fällt der dritte aus muss der Datensatz auf den ersten verschoben werden, kommt ein vierter dazu muss er auf den dritten kopiert werden. Dies zeigt deutlich den Aufwand der Neuverteilung bei Änderungen der verfügbaren Serverknoten.

Um dieses Problem zu umgehen wird das Consistent Hashing verwendet. Dabei wird ein fester Adressbereich definiert, der von einer Hashfunktion abgedeckt ist. Jeder Serverknoten wird dann mithilfe der Hashfunktion in diesem Adressbereich platziert. Jeder Datensatz wird nun über die Hashfunktion in genau den Knoten gespeichert, der dem Hashwert am nächsten ist. Bildlich lässt sich dies am besten als Ring darstellen. Die Knoten werden auf dem Ring platziert und halten alle Datensätze die beispielsweise entgegengesetzt des Uhrzeigersinns vor ihnen liegen. Dieses Verfahren ist in der folgenden Abbildung dargestellt.



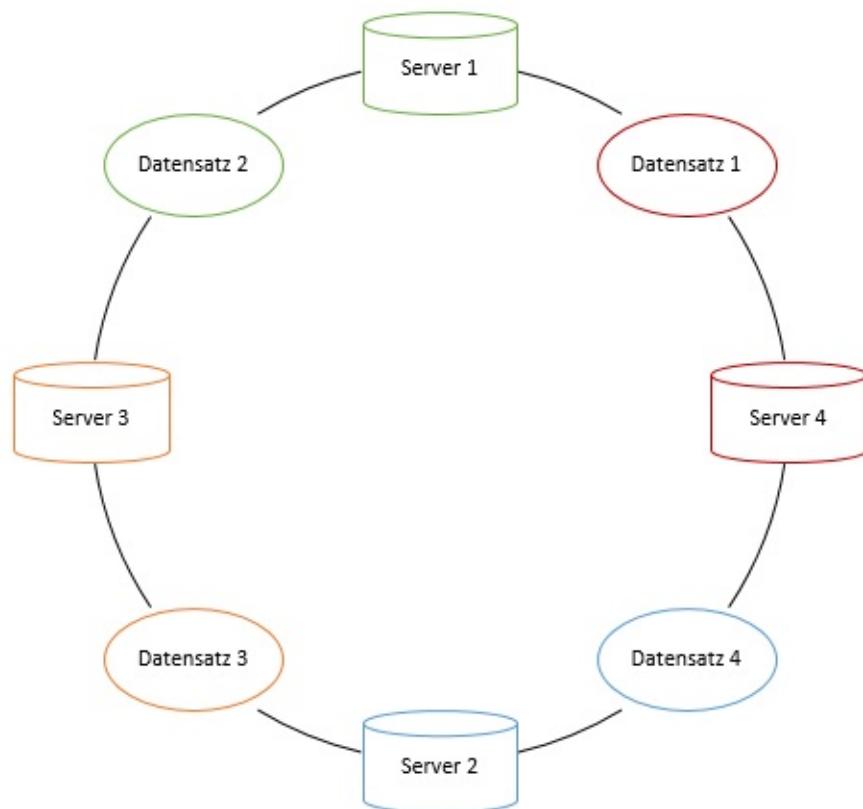
(Eigenerstellung, vgl. [2 Kap. 2.3])

Fällt nun ein Serverknoten aus, so müssen nur die Datensätze verschoben werden, die bisher von ihm gehalten wurden. Diese werden dann vom nächsten Knoten im Uhrzeigersinn gehalten. Alle anderen Datensätze müssen nicht neuverteilt werden und können weiterhin auf den bisherigen Knoten bleiben.



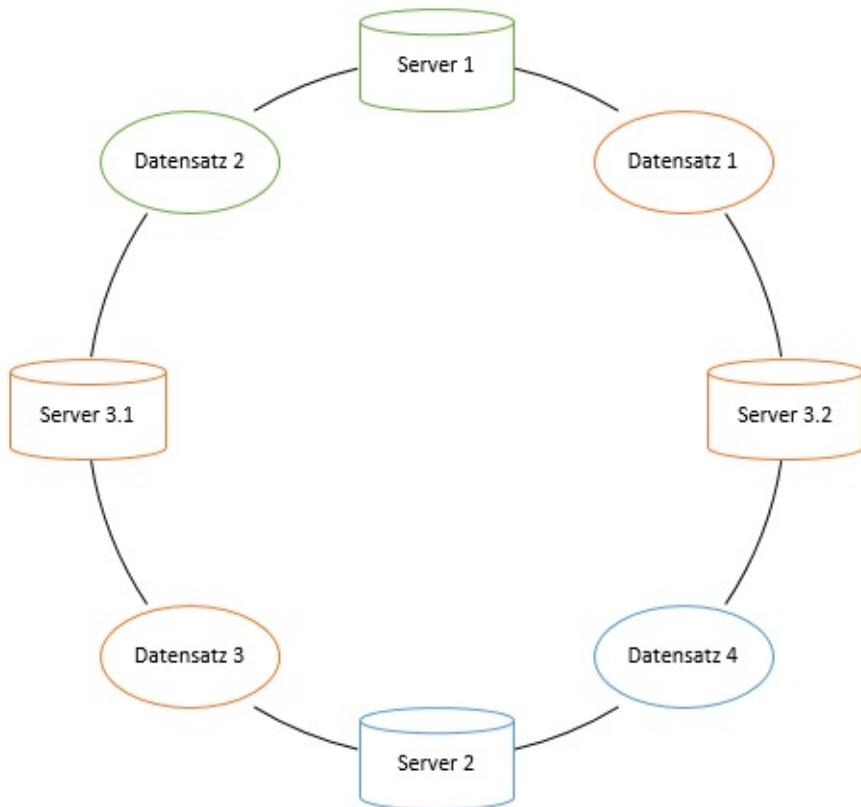
(Eigenerstellung, vgl. [2 Kap. 2.3])

Genauso verhält es sich, wenn ein neuer Knoten hinzugefügt wird. Der neue bekommt dann alle Datensätze, die entgegengesetzt des Uhrzeigersinns vor ihm liegen.



(Eigenerstellung, vgl. [2 Kap. 2.3])

Des Weiteren ist es möglich, virtuelle Serverknoten einzufügen um eine bessere Lastverteilung zu erreichen. Gibt es beispielsweise einen Knoten, der leistungsstärker ist als die anderen, oder mehr Speicherplatz besitzt, so wird für ihn ein virtueller Serverknoten auf dem Ring platziert werden. Jeder Datensatz, der nun diesem virtuellen Serverknoten zugeordnet ist, wird auf dem Knoten gespeichert, zu dem der virtuelle gehört. In der folgenden Abbildung besitzt beispielsweise Server drei mehr Speicherplatz als die anderen. Dementsprechend wird ein virtueller Knoten für ihn auf dem Ring platziert, sodass nun mehr Datensätze auf dem dritten Server gespeichert werden.



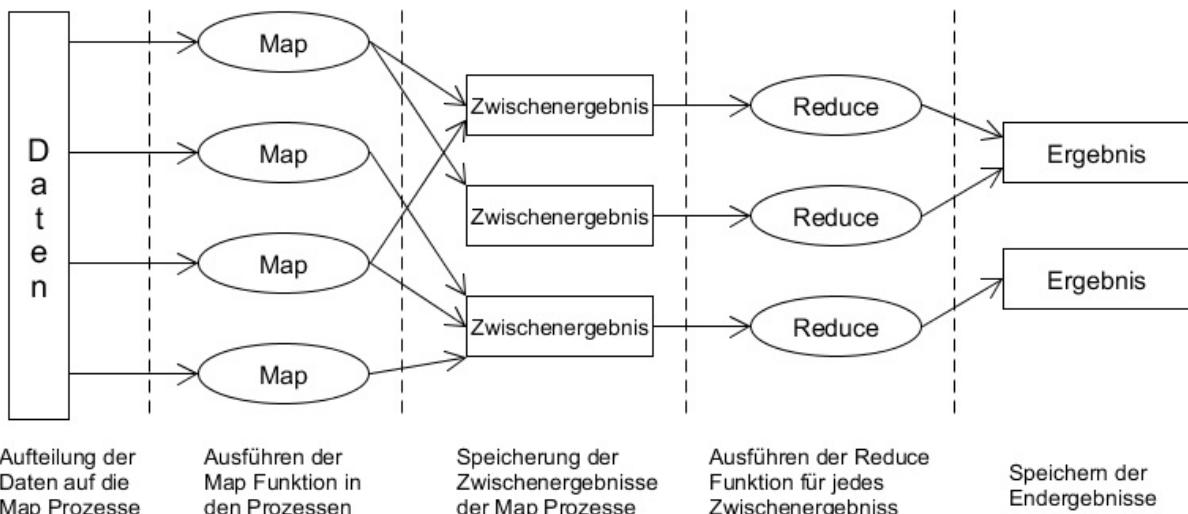
(Eigenerstellung, vgl. [2 Kap. 2.3])

Consistent Hashing wird von vielen NoSQL Datenbanksystemen verwendet. Im Zusammenspiel mit verschiedenen Formen der Replikation und Sharding lassen sich dadurch robuste Datenbankservercluster aufbauen, die durch einfaches hinzufügen oder entfernen von Knoten sich leicht an ändernde Begebenheiten anpassen lassen.

Map-Reduce (vgl. [1 Kap. 5.4],[2 Kap. 2.1],[12 Kap. 4],[14 Kap. 6.4.1,6.5],[31-43])

Durch den Bereich Big-Data wird es immer wichtiger, sehr große verteilte Datenmengen verarbeiten zu können. Da dies auch in angemessener Zeit erfolgen soll, ist es von Vorteil wenn die Datenverarbeitung parallel ausgeführt wird. Um dies umzusetzen wurde von Google das Map-Reduce Framework entwickelt.

Die Funktionsweise von Map-Reduce lässt sich in verschiedene Phasen untergliedern und ist in der folgenden Abbildung dargestellt.



(Eigenerstellung, vgl. [2 S. 17])

- Im ersten Schritt werden die Daten auf die verschiedenen Knoten im Netz verteilt, die auch die Map-Funktionen ausführen.
- Anschließend werden in den verschiedenen Prozessen parallel die vom Nutzer spezifizierte Map-Funktion ausgeführt.
- Die Ergebnisse der Map-Prozesse werden dann in unterschiedlichen Zwischenergebnissen gespeichert.
- Daraufhin wird für jedes Zwischenergebnis ein Reduce Prozess gestartet, die dann die vom Nutzer definierte Reduce-Funktion ausführen.
- Im letzten Schritt werden dann alle Ergebnisse der Reduce-Prozesse gespeichert.

Da das Framework die Aufteilung und Parallelisierung der Daten und Prozesse übernimmt, muss der Nutzer selbst nur die Map und Reduce Funktionen angeben. Diese spezifizieren dann die gewünschte Logik. Dabei wird mit Key-Value Wertpaaren gearbeitet. In der Map Phase können beispielsweise die Vorkommnisse einzelner Wörter gesammelt werden. In der Reduce Phase können dann Aggregationen durchgeführt werden, beispielsweise um Durchschnittswerte zu berechnen.

Es gibt einige NoSQL Datenbanksysteme, die das Map-Reduce Prinzip nutzen, um mit verteilten Datensätzen zu arbeiten. Häufig muss dabei keine Verteilung der Daten mehr stattfinden, da diese bereits auf verschiedene Knoten verteilt sind. Jeder Serverknoten kann daher für seine eigenen Datensätze die Map-Funktion ausführen.

In Memory Datenbanken (vgl. [8-11],[14 Kap. 8])

Klassischerweise werden Daten jeglicher Art meist auf Festplatten gespeichert. Dies ist auch generell bei Relationalen Datenbankmanagementsystemen der Fall, die Datenbank, bzw. die entsprechenden Daten werden auf Festplattenlaufwerke gespeichert. Es gibt jedoch auch eine andere Möglichkeit Daten zu speichern: Die Speicherung im

Arbeitsspeicher (RAM). Datenbanken die ihre Daten innerhalb des RAM speichern werden als In-Memory Datenbanken bezeichnet. Es gibt einige NoSQL Systeme, die diese Art der Datenhaltung nutzen.

Die Speicherung der Daten im Arbeitsspeicher hat den Vorteil, dass Zugriffsgeschwindigkeiten deutlich besser sind als im Vergleich zum Festplattenspeicher. Lese- und Schreiboperationen auf den Arbeitsspeicher sind deutlich schneller als auf Festplattenspeicher. Häufig finden In-Memory Datenbanken daher Anwendung im Big-Data Bereich, da dort mit sehr großen Mengen an Daten gearbeitet wird, die analysiert und ausgewertet werden müssen. Ein Nachteil ist im Allgemeinen jedoch die Tatsache, dass Arbeitsspeicher nur flüchtig ist. Ein Absturz des Systems führt daher dazu, dass alle Daten, die im Arbeitsspeicher gehalten wurden, verloren gehen. Eine dauerhafte Speicherung der Daten (Persistenz) ist somit nicht möglich. Es gibt jedoch Möglichkeiten, wie eine Persistente Datenhaltung trotzdem erreicht werden kann:

- Es können Schnappschuss-Dateien erstellt werden, die den Zustand der Datenbank zum Zeitpunkt des Schnappschuss vollständig erfassen. Diese Dateien können dann auf einem persistenten Speichermedium gespeichert werden. Der Nachteil bei dieser Variante ist, dass bei einem Systemabsturz alle Änderungen der Daten zwischen dem letzten Schnappschuss und dem Ausfall verloren gehen.
- Zusätzlich zum Anlegen von Schnappschüssen kann ein Protokoll geführt werden. Dieses speichert alle Änderungen, die an den Daten der Datenbank vollzogen werden. Somit kann bei einem Absturz mithilfe des Schnappschuss und des Protokolls der Zustand der Datenbank wiederhergestellt werden.
- Eine weitere Möglichkeit ist das Nutzen von nichtflüchtigem RAM Speicher (NVRAM). Die Datenhaltung im RAM wird dabei durch einen Energiespeicher, beispielsweise eine Batterie, sichergestellt, bis das System wieder am Laufen ist.

Da Arbeitsspeicher jedoch deutlich teurer ist im Vergleich zu Festplattenspeicher werden häufig auch Hybride Ansätze genutzt. Die Daten der Datenbank können dabei sowohl im Arbeitsspeicher, als auch auf der Festplatte gespeichert werden.

Spaltenorientierte Datenbanken (vgl. [1 Kap. 7.3],[2 Kap. 3],[12 Kap. 2],[13 Kap. 3], [14 Kap. 5.5,5.6],[72,98-106])

Spaltenorientierte Datenbanken (auch Wide-Column-Stores genannt) speichern ihre Daten, vereinfacht gesagt, nicht wie relationale Datenbanken in Zeilenform, sondern in Spaltenform. Eine relationale Datenbank würde Datensätze beispielsweise auf folgende Weise speichern:

- 1, Peter, 40000
- 2, Maria, 45000
- 3, Dieter, 35000

Der erste Wert stellt dabei eine ID dar, der zweite den Namen der Person und der dritte das Gehalt.

Eine spaltenorientierte Datenbank würde die gleichen Datensätze auf folgende Weise speichern:

- 1, 2, 3
- Peter, Maria, Dieter
- 40000, 45000, 35000

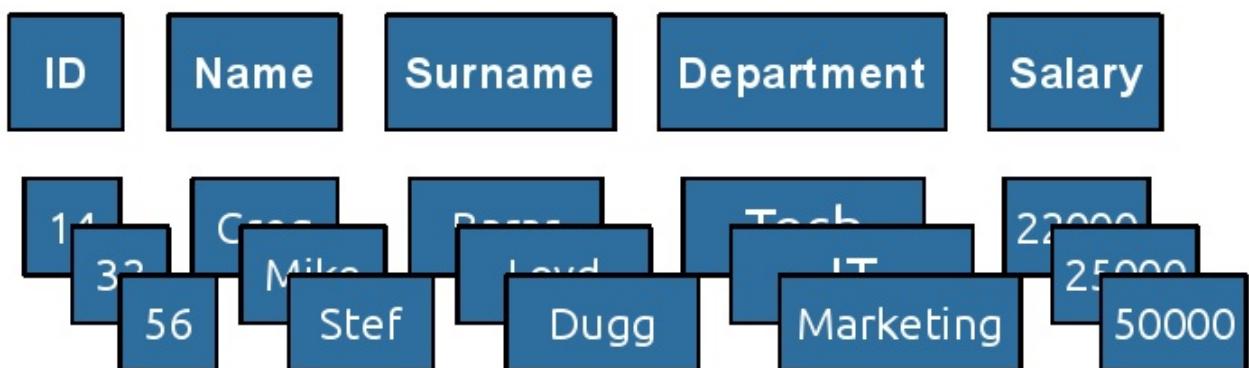
Dies ist zwar nur ein sehr vereinfachtes Beispiel, da die tatsächliche Art der Speicherung von spaltenorientierten Datenbanken anders umgesetzt ist, es zeigt jedoch direkt den Nutzen einer solchen Form der Datenstrukturierung. Würde man beispielsweise das Durchschnittsgehalt aller Personen berechnen wollen, so wird bei relationalen Datenbanken zuerst der erste Datensatz ausgewählt werden und in diesem dann der dritte Wert, der das Gehalt beschreibt. Dasselbe müsste für den zweiten und dritten Datensatz geschehen. Bei spaltenorientierten Datenbanken müsste nur der dritte Datensatz ausgewählt und all seine Werte gelesen werden. Statt auf drei muss also nur auf ein Datensatz zugegriffen werden, was deutlich schneller ist.

Funktionsweise

Die tatsächliche Art der Speicherung der Daten unterscheidet sich teilweise bei den verschiedenen Anbietern. Im Allgemeinen gibt es jedoch Spalten, die einen Namen besitzen, Daten halten und einen Zeitstempel haben. Die Struktur einer Spalte ist in folgender Abbildung dargestellt [99].



Spalten die ähnliche oder verwandte Inhalte speichern können in einer Column Family zusammengefasst werden. Beispielsweise könnten alle Spalten, die eine Person beschreiben, Teil einer Column Family werden, was in folgender Abbildung abgebildet ist [99].



[99]

Eine solche Column Family hat dabei keine logische Struktur und kann sogar aus Millionen Spalten bestehen.

Vorteile

Wie bereits durch das Beispiel deutlich wurde, sind spaltenorientierte Datenbanken besonders dafür geeignet Aggregate von Daten zu bilden, wie beispielsweise der Durchschnittswert oder Minima/Maxima. Dadurch, dass die Daten einer Spalte zusammen gespeichert werden, ist der Zugriff darauf bei spaltenorientierten Datenbanken deutlich schneller als bei relationalen Datenbanken. Auch das Ändern von allen Daten einer Spalte, beispielsweise bei einer allgemeinen Gehaltserhöhung ist deutlich effizienter. Des Weiteren lassen sich Daten in einer spaltenorientierten Datenbank leichter komprimieren. Allgemein sind sie also gut für Einsatzzwecke wie analytische Informationssysteme und Finanzberichte.

Nachteile

Der Nachteil dieser Datenbanken ist jedoch, dass Zugriffe auf alle Attribute eines Objekts, also eine Zeile, dementsprechend deutlich langsamer sind. Das Hinzufügen neuer Datensätze, beispielsweise ein neuer Mitarbeiter, ist bei spaltenorientierten Datenbanken deutlich weniger effizient als bei relationalen Datenbanken.

Anbieter [107]

Die am häufigsten genutzten Spaltenorientierten Datenbanken laut db-engines.com (Stand Juli 2017)

- Cassandra: 124,12 Punkte
- HBase: 63,62 Punkte
- Microsoft Azure Cosmos DB: 7,71 Punkte

Dokumentenorientierte Datenbanken (vgl. [1 Kap. 7.4],[2 Kap. 4],[13 Kap. 3],[14 Kap. 5.8],[72,108-114])

Diese Art der NoSQL Datenbanken speichern ihre Daten in Dokumenten ab, beispielsweise in Form von JSON, XML aber auch in beliebig anderer Form. Dabei sind diese Dokumente Schemafrei, das heißt es existieren keine Regeln nach denen der Inhalt dieser Dokumente aufgebaut werden muss. Jedes einzelne Dokument ist also in der Lage, einen völlig anderen Inhalt zu speichern. Häufig werden diese Datenbanken im Bereich der Web-Applikationen verwendet und bei unstrukturierten Daten.

Funktionsweise

In den meisten Fällen bestehen die Dokumente aus einer Sammlung von Key-Value Paaren. Die entsprechenden Values müssen dabei nicht atomar sein, sondern können auch aus Arrays, Listen oder ganzen weiteren Objekten bzw. Datensätzen bestehen. Das folgende Beispiel zeigt ein Dokument im JSON Format.

```
{  
  "Vorname": "Dieter",  
  "Nachname": "Skolmsund",  
  "Gehalt": "40000",  
  "Bisherige Positionen": ["Gärtner", "Entwickler", "Abteilungsleiter"],  
  "Kinder":  
    [  
      {"Name": "Bjorn"},  
      {"Name": "Freya"}  
    ]  
}
```

In vielen Datenbanken wird zusätzlich eine ID für das Dokument gespeichert, diese kann manuell angegeben werden oder wird vom Datenbankmanagementsystem selbst angelegt.

Vorteile

Aufgrund der Schemafreiheit bieten dokumentorientierte Datenbanken eine große Freiheit bei der Gestaltung der Datenstrukturen. Änderungen können meist sehr einfach umgesetzt werden, gerade in der Entwicklungsphase sind sie daher äußerst praktisch. Des Weiteren können zusammenhängende Objekte bzw. Datensätze innerhalb eines einzelnen Dokuments gespeichert werden. Operationen die auf all diese Daten zugreifen müssen, sind daher schneller als bei relationalen Datenbanken, bei denen viele Join Befehle dafür nötig wären.

Nachteile

Zwischen den einzelnen dokumentorientierten Datenbankmanagementsystemen existieren viele Unterschiede, sodass ein Wechsel von einem Anbieter zu einem anderen nicht immer ganz einfach ist. Beispielsweise erlauben manche Anbieter das Speichern eines Dokuments in XML Format, andere wiederum nicht. Auch in den entsprechend genutzten Abfragesprachen gibt es häufig Unterschiede. Die Eigenschaft der Schemafreiheit kann unter Umständen auch als Nachteil angesehen werden. Anwendungsfälle, bei denen es auf eine Konsistente und strukturierte Speicherung der Daten ankommt, können mit dokumentorientierten Datenbanken nur deutlich schwerer gelöst werden als mit klassischen relationalen Datenbanken. Während bei relationalen Datenbanken ein Anwendungsprogramm keine Daten speichern kann, die nicht zum Schema passen, muss bei dokumentorientierten Datenbanken eine Überprüfung auf Programm Ebene erfolgen. Bei Fehlern in der Programmierung kann es also schnell dazu kommen, dass Daten in der Datenbank gespeichert werden, die an anderer Stelle nicht mehr korrekt interpretiert werden können.

Anbieter [115]

Die am häufigsten genutzten dokumentenorientierten Datenbanken laut db-engines.com (Stand Juli 2017)

- MongoDB: 332,77 Punkte
- Amazon DynamoDB: 36,46 Punkte
- Couchbase: 33,02 Punkte

Key-Value Datenbanken (vgl. [1 Kap. 7.2],[2 Kap. 5],[12 Kap. 6],[13 Kap. 3],[72,116-123])

Key-Value Datenbanken existieren schon seit den 70er Jahren und gehören somit zu den ältesten NoSQL Datenbanken. Durch das Internet und der damit einhergehenden Notwendigkeit mit extrem großen Datenmengen arbeiten zu können, haben diese Datenbanken einen neuen Aufschwung erlebt. Sie speichern ihre Daten in Form von Schlüssel-Wert Paaren, eine sehr einfache, aber dadurch auch sehr effiziente Form der Datenverwaltung.

Funktionsweise

Je nach Anbieter unterscheiden sich die Freiheiten beim Setzen der Schlüssel und Werte. Im Allgemeinen sind sie jedoch Schemafrei, die Schlüssel und die Werte können daher vollkommen beliebig aufgebaut sein. Häufig können Werte daher nicht nur einfache Strings halten, sondern auch Listen und Sets, also mehrere Werte. Auch Hashs können gespeichert werden, welche aus weiteren Key-Value Paaren aufgebaut sind.

Vorteile

Der größten Vorteile von Key-Value Datenbanken sind ihre Skalierbarkeit und ihre Geschwindigkeit. Die Daten lassen sich sehr leicht verteilen und der Zugriff auf Werte über ihre Schlüssel erfolgt sehr schnell. Durch die Schemafreiheit können beliebige Strukturen aufgebaut werden, daher sind sie flexibel und lassen sich leicht anpassen. Oft werden diese Datenbanken auch als In-Memory Datenbanken verwendet, wodurch sie noch schneller werden.

Nachteile

Da der Zugriff auf Werte nur über deren Schlüssel erfolgen kann, sind die Abfragemöglichkeiten bei diesen Datenbanken deutlich eingeschränkt. Komplexe Abfragen müssen somit von den Entwicklern in ihrem Programmcode selbst erstellt werden. Des Weiteren lassen sich Objektbeziehungen nur sehr schwer darstellen. Integritätsbedingungen

dieser Beziehungen müssen immer im Programmcode definiert werden. Key-Value Datenbanken sind daher nicht dafür geeignet, Datensätze, mit vielen komplexen Beziehungen untereinander, zu speichern.

Anbieter [123]

Die am häufigsten genutzten Key-Value Datenbanken laut db-engines.com (Stand Juli 2017)

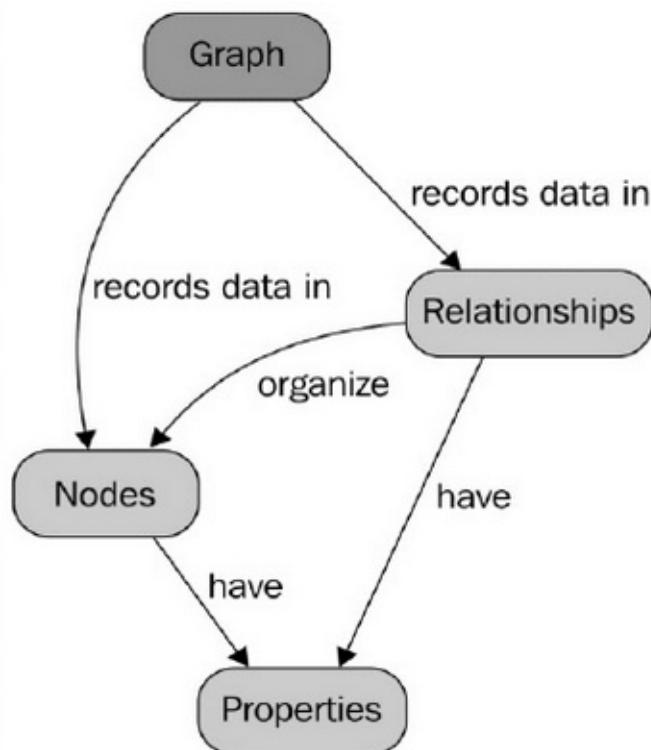
- Redis: 121,51 Punkte
- Memcached: 28,53 Punkte
- Hazelcast: 8,91 Punkte

Graphen Datenbanken (vgl. [1 Kap. 1.4,2.4,7.6],[2 Kap. 6],[12 Kap. 3],[13 Kap. 3],[72,89-96])

Graphen Datenbanken eignen sich dafür, Objekte und ihre komplexen Beziehungen untereinander zu speichern und zu verwalten. Die Daten werden dabei als Graphen gespeichert, Objekte werden als Knoten realisiert, Beziehungen über Kanten dargestellt. Sie eignen sich daher besonders gut für Daten mit sehr vielen Beziehungen und bei Graphen theoretischen Problemen. Anwendungsfälle sind beispielsweise Soziale Netzwerke, Empfehlungssysteme und Geoinformationssysteme.

Funktionsweise

Die meisten Graphen Datenbanken speichern ihre Daten in einem Property-Graphen. Dies ist ein Graph, der zusätzlich zu Knoten und Kanten (Beziehungen) auch beliebige Eigenschaften zu diesen Elementen speichern kann. Knoten und Beziehungen können somit durch Attribute weiter spezifiziert werden. Die nachfolgende Abbildung stellt grafisch dar, wie eine Graphen Datenbank aufgebaut ist [13].



[13]

Eine weitere wichtige Eigenschaft ist, dass es zwischen zwei Knoten auch mehrere Kanten geben kann. So lassen sich auch mehrere komplexe Verbindungen zwischen beliebigen Objekten, bzw. Knoten darstellen.

Die Funktionsweise von Graphen Datenbanken basiert dabei auf den mathematischen Eigenschaften der Graphentheorie.

Vorteile

Der größte Vorteil dieser Datenbanken besteht in ihrem speziellen Einsatzgebiet: Alles was sich als Graph abbilden lässt, kann mit deutlich besserer Performance in Graphen Datenbanken gespeichert und verwaltet werden, als in anderen Datenbanken. Alle Probleme die auf der Graphentheorie basieren, lassen sich leicht lösen, beispielsweise die Suche nach den kürzesten Wegen. Während bei relationalen Datenbanken dafür komplexe Joins verwendet werden müssen, können in Graphen Datenbanken durch einfaches Traversieren zwischen den Knoten diese Probleme schnell und einfach gelöst werden. Graphen Datenbanken eignen sich daher am besten für Datensätze, die sehr viele Beziehungen untereinander haben.

Nachteile

Ihre Spezialisierung auf Graphen ist allerdings auch ihr größter Nachteil. Sie eignen sich nicht dafür große Datensätze mit keinen oder wenigen Beziehungen untereinander zu verwalten, relationale Datenbanken können mit diesen Daten meist schneller und effizienter umgehen. Des Weiteren existieren noch keine standardisierte Abfragesprache, jeder Anbieter hat hier meist eigene Lösungen, was es nicht einfach macht, einen Anbieter zu wechseln.

Anbieter [97]

Die am häufigsten genutzten Graphen Datenbanken laut db-engines.com (Stand Juli 2017)

- Neo4j: 38,52 Punkte
- Microsoft Azure Cosmos DB: 7,71 Punkte
- OrientDB: 5,57 Punkte

Objektorientierte Datenbanken (vgl. [2 Kap. 8.5],[14 Kap. 2.11,7],[72-86])

Heutzutage werden sehr viele Programme mit dem Konzept der Objektorientierung programmiert, bekannte Sprachen sind dabei beispielsweise Java und C#. Daten werden dabei als Objekte betrachtet, mit denen gearbeitet werden kann. Relationale Datenbanken haben jedoch keine Möglichkeit diese Objekte einfach so zu speichern, da diese immer mit Tabellen arbeiten. Dieses Problem wird als Impedance Mismatch bezeichnet.

Impedance Mismatch (vgl. [70-71])

Impedance Mismatch (zu Deutsch: Objektrelationale Unverträglichkeit) beschreibt die Unverträglichkeit zwischen dem objektorientierten und dem relationalen Modell. Die Probleme sind dabei einerseits Unterschiede zwischen den Datentypen einer Datenbank und einer Programmiersprache. Der Datentyp String kann beispielsweise in einer Datenbank anders aufgebaut sein als in einer Programmiersprache und eventuell Beschränkungen in der Länge haben. Andererseits sind es vor allem die Unterschiede in den Datenstrukturen an sich. Objekte sind anders aufgebaut als Tabellen. Häufig müssen Daten von Objekten in verschiedene Tabellen aufgeteilt werden. Des Weiteren ist es nicht möglich objektorientierte Eigenschaften wie Vererbung, Objektidentität und das Objektverhalten in einer Relation zu speichern

Funktionsweise

Objektorientierte Datenbanken sind dabei eine Antwort auf das Problem des Impedance Mismatch. Diese Datenbanken speichern die Daten als Objekte gemäß der Objektorientierung. Objekte können somit direkt gespeichert werden, ohne dass ihre Daten erst aufgeteilt werden müssen. Dazu gehört auch die Objektidentifikation, es müssen für Objekte also keine zusätzlichen Primärschlüssel generiert werden, wie es bei relationalen Datenbanken der Fall ist. Dadurch, dass Objekte auch komplexe Datentypen halten können, sprich andere Objekte, lassen sich somit auch direkte Objektreferenzen speichern. Beziehungen über Primär- und Fremdschlüssele sind daher nicht notwendig. Auch lassen sich Vererbungshierarchien speichern und abbilden. Zusätzlich kann das Objektverhalten definiert und gespeichert werden, eine Eigenschaft, für die es im klassischen relationalen Modell kein Gegenstück gibt.

In vielen Objektorientierten Datenbanken wird als Abfragesprache OQL (Object Query Language) verwendet, eine Abfragesprache angelehnt an SQL. Häufig werden Anbindungen für die wichtigsten Objektorientierten Programmiersprachen angeboten.

Vorteile

Durch die komplette Speicherung der Daten als Objekte gemäß der Objektorientierung ist es deutlich einfacher Anwendungen zu programmieren, da keine Möglichkeit mehr gefunden werden muss, wie sich die Objekte am besten in Relationen abbilden lassen. Beim Arbeiten mit wenigen, aber sehr komplexen Objekten mit Verknüpfungen, sind Objektdatenbanken oft schneller als relationale, da keine großen Join Abfragen benötigt werden.

Nachteile

Der größte Nachteil objektorientierter Datenbanken ist ihre Verbreitung, bzw. das Fehlen davon. Obwohl die Entwicklung dieser Datenbanken bereits seit den 80er Jahren stattfindet, werden sie nicht sehr häufig genutzt und es gibt nur wenige Anbieter. Bedingt dadurch sind sie auch meist nicht so ausgereift, wie die älteren relationalen Datenbanken. Ihre Performance beim Arbeiten mit großen aber einfachen Datenmengen liegt deutlich unter der von relationalen Datenbanken.

Anbieter [87]

Die am häufigsten genutzten Objektorientierten Datenbanken laut db-engines.com (Stand Juli 2017)

- Caché: 2,72 Punkte
- Db40: 1,42 Punkte
- ObjectStore: 1,10 Punkte

Objektrelationale Datenbanken (vgl. [1 Kap. 6.6],[14 Kap. 7],[72-86])

Neben den Objektorientierten Datenbanken gibt es auch die sogenannten objektrelationalen Datenbanken. Auch diese haben zum Ziel das Problem des Impedance Mismatch anzugehen, allerdings streben sie eher eine Verminderung statt vollständiger Beseitigung des Problems an. objektrelationale Datenbanken versuchen die Vorteile von relationalen und objektorientierten Datenbanken zu vereinen. Dazu verwenden sie das relationale Grundgerüst und erweitern es um objektorientierte Features. Als Abfragesprache wird weiterhin das bekannte SQL genutzt. Die Forschung an objektrelationalen Datenbanken begann Anfang der 90er Jahren, erste kommerzielle Produkte gab es gegen Ende der 90er. Ihre Entwicklung wurde durch den SQL:1999 Standard und seine Structured Types vorangetrieben, welche es ermöglichen eigene komplexe Datentypen in Datenbanken zu definieren.

Funktionsweise

Objektrelationale Datenbanksysteme sind zum Großteil bestehende relationale Datenbanksysteme, die zusätzlich manche der folgenden objektorientierten Eigenschaften bieten:

- Komplexe Datentypen:

Neben den einfachen Datentypen wie Int und String sind auch komplexe wie Geldbeträge, Geometrie Daten und Dokumenttypen wie JSON und XML verwendbar.

- Benutzerdefinierte Datentypen:

Benutzer können eigene Datentypen spezifizieren und nutzen. Diese können aus einfachen, komplexen oder auch aus benutzerdefinierten Datentypen bestehen.

- Objekttabellen/Typisierte Tabellen:

Es lassen sich Tabellen anhand eines Datentyps erstellen. Die Tabelle hält dann Objekte/Datensätze des spezifizierten Typs. Die Attribute eines Datentyps werden dabei zu den Spalten der Tabelle.

- Objektidentifikation:

Objekte lassen sich durch einen Objektidentifikator identifizieren.

- Objektreferenzen:

Mithilfe der Objektidentifikatoren lassen sich Objekte einfach referenzieren.

Beziehungen über Primär- und Fremdschlüssel werden dadurch überflüssig.

- Methoden:

Es lassen sich Objektmethoden erstellen.

- Vererbung:

Es lassen sich Vererbungshierarchien zwischen Objekten herstellen.

Vorteile

Der Vorteil objektrelationaler Datenbanken ist, dass das Impedance Mismatch verringert wird und dennoch das relationale Modell verwendet werden kann. Es ist daher sehr einfach, bestehende relationale Systeme in objektrelationale Datenbanken zu migrieren, um die Vorteile der Objektorientierung zu nutzen.

Nachteile

Ein großer Nachteil der objektrelationalen Datenbanken ist der große Unterschied zwischen den einzelnen Anbietern. Nicht alle objektorientierten Features werden immer umgesetzt und auch generell gibt es große Unterschiede in der Umsetzung. Während Oracle beispielsweise fast alle objektorientierten Features umsetzt, bietet der Microsoft SQL Server nur benutzerdefinierte Datentypen an. Das Wechseln auf ein anderes objektrelationales Datenbankmanagementsystem ist daher in vielen Fällen nur schwer oder gar nicht möglich.

Anbieter [88]

Die am häufigsten genutzten relationalen Datenbanken mit objektorientierten Features laut db-engines.com (Stand Juli 2017)

- Oracle: 1374,88 Punkte
- Microsoft SQL Server: 1226,00 Punkte
- PostgreSQL: 369,44 Punkte

Fazit

NoSQL Datenbanken zeichnen sich durch große Vielfalt und Unterschiede aus. Es wurden die grundlegenden Theorien und Verfahren erläutert, die von NoSQL Datenbanken genutzt werden. Statt auf das ACID Transaktionskonzept setzen NoSQL Datenbanken meist auf BASE. Statt also die Konsistenz im CAP-Theorem als oberstes Ziel zu nehmen, begnügen sie sich meist mit eventueller, bzw. verzögerter Konsistenz. Dadurch, dass sie meist als verteilte Datenbanken im Cluster realisiert werden, sind sie optimiert für eine vertikale Skalierung. Mithilfe konsistenter Hashverfahren können sie ihre Daten optimal auf Shards aufteilen und auch Replikationen können damit effizient verteilt werden. Aufgrund der Verteilung setzen einige NoSQL Datenbanken auch auf Map-Reduce Verfahren, um Daten in angemessener Zeit verarbeiten zu können. Andere wiederum nutzen das In-Memory Verfahren und speichern ihre Daten im Arbeitsspeicher, um einen Geschwindigkeits-Gewinn beim Arbeiten mit den Daten zu erlangen.

Außerdem wurden die wichtigsten NoSQL-Arten erklärt und es wurde beschrieben, wie sie ihre Daten speichern. Neben den vier Kern NoSQL-Arten, Dokument-, Spalten-, Graph- und Key-Value-basiert wurden dabei auch Objekt und Objektrelationale Datenbanken erwähnt. Es hat sich gezeigt, dass alle Varianten sowohl Vorteile, als auch Nachteile besitzen und meist für bestimmte Anwendungsfälle besser geeignet sind als für andere. Des Weiteren wurde deutlich, dass auch innerhalb der Varianten Unterschiede zwischen den verschiedenen Anbietern existieren. Bei der Auswahl einer Datenbank sollte dementsprechend geprüft werden, welcher Anwendungsfall vorliegt und welche Besonderheiten die einzelnen Datenbanken der Anbieter haben, um eine entsprechend bestmögliche Datenbank auszuwählen.

Quellen NoSQL

- [1] Meier, Andreas; Kaufmann, Michael. "SQL- & NoSQL-Datenbanken", Springer Vieweg. (2016)
- [2] Edlich, Stefan; Friedland, Achim; Hampe, Jens; Brauer, Benjamin; Brückner, Markus. "NoSQL Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken", Carl Hanser Verlag GmbH & Co. KG. (2011)
- [3] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.Skalierbarkeit (abgerufen am 22.06.2017)
- [4] <http://www.se.uni-hannover.de/pub/File/kurz-und-gut/ws2011-labor-restlab/RESTLab-Skalierbarkeit-Oliver-Beren-Kaul-kurz-und-gut.pdf> (abgerufen am 22.06.2017)
- [5] <https://de.wikipedia.org/wiki/Skalierbarkeit> (abgerufen am 22.06.2017)
- [6] http://www.it-administrator.de/themen/server_client/fachartikel/204189.html (abgerufen am 22.06.2017)
- [7] <http://habacht.blogspot.de/2007/11/das-uneindeutig-unklare-duo-scale-out.html> (abgerufen am 22.06.2017)
- [8] http://trends-in-der-it.de/?Fachartikel/In-Memory-_Technologie (abgerufen am 23.06.2017)
- [9] <http://www.datenbanken-verstehen.de/lexikon/in-memory-datenbank/> (abgerufen am 23.06.2017)
- [10] http://winfwiki.wi-fom.de/index.php/Beschreibung_und_Einsatz_von_In-Memory-Computing#In-Memory-Datenbanken (abgerufen am 23.06.2017)
- [11] <https://de.wikipedia.org/wiki/In-Memory-Datenbank> (abgerufen am 23.06.2017)
- [12] Celko, Joe. "Joe Celko's Complete Guide to NoSQL", Elsevier Science. (2013)
- [13] Vaish, Gaurav. "NoSQL Starter", Packt Publishing. (2013)
- [14] Lake, Peter; Crowther, Paul. "Concise Guide to Databases", Springer Verlag. (2013)
- [15] <https://www.bg.bib.de/portale/dab/Grundlagen/acid.html> (abgerufen am 23.06.2017)
- [16] <http://www.itwissen.info/ACID-atomicity-consistency-isolation-durability-ACID-Paradigmen.html> (abgerufen am 23.06.2017)
- [17] http://www.dbs.ethz.ch/education/timi/WS_03_04/Vorlesungsunterlagen/1-Einfuehrung.pdf (abgerufen am 23.06.2017)
- [18] <https://de.wikipedia.org/wiki/ACID> (abgerufen am 23.06.2017)
- [19] <http://datenbanken-verstehen.de/blog/datenbankentwicklung/datenbanktransaktionen-acid-prinzip/> (abgerufen am 23.06.2017)
- [20] <https://dbs.uni-leipzig.de/buecher/DBSI-Buch/HTML/kap13-2.html> (abgerufen am 23.06.2017)

- 23.06.2017)
- [21] <http://www.searchenterprisesoftware.de/definition/AKID-ACID> (abgerufen am 23.06.2017)
 - [22] <https://db-engines.com/de/article/ACID> (abgerufen am 23.06.2017)
 - [23] <http://www.norcom.de/de/fachartikel/datenbanktechnologien-acid-base-cap-und-google-spanner> (abgerufen am 23.06.2017)
 - [24] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/CAP (abgerufen am 23.06.2017)
 - [25] <https://blog.codecentric.de/2011/08/grundlagen-cloud-computing-cap-theorem/> (abgerufen am 23.06.2017)
 - [26] <https://dzone.com/articles/better-explaining-cap-theorem> (abgerufen am 23.06.2017)
 - [27] <https://db-engines.com/de/article/CAP+Theorem> (abgerufen am 23.06.2017)
 - [28] <http://blog.nahurst.com/visual-guide-to-nosql-systems> (abgerufen am 23.06.2017)
 - [29] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.BASE (abgerufen am 23.06.2017)
 - [30] <https://db-engines.com/de/article/BASE> (abgerufen am 23.06.2017)
 - [31] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/MapReduce (abgerufen am 23.06.2017)
 - [32] <https://en.wikipedia.org/wiki/MapReduce> (abgerufen am 23.06.2017)
 - [33] <https://de.wikipedia.org/wiki/MapReduce> (abgerufen am 23.06.2017)
 - [34] <http://www.pal-blog.de/entwicklung/perl/einfach-erklaert-mapreduce-tutorial.html> (abgerufen am 23.06.2017)
 - [35] https://dbs.uni-leipzig.de/file/seminar_0910_koenig_ausarbeitung.pdf (abgerufen am 23.06.2017)
 - [36] https://dbs.uni-leipzig.de/file/seminar_0910_findling_K%C3%B6nig.pdf (abgerufen am 23.06.2017)
 - [37] <https://docs.mongodb.com/manual/core/map-reduce/> (abgerufen am 23.06.2017)
 - [38] <http://scienceblogs.de/dlaxs-rake/2012/11/18/big-data-1-einfaches-word-count-beispiel-in-python/?all=1> (abgerufen am 23.06.2017)
 - [39] <http://www.searchenterprisesoftware.de/definition/MapReduce> (abgerufen am 23.06.2017)
 - [40] <https://nirajrules.wordpress.com/2013/05/31/big-data-nosql-and-mapreduce/> (abgerufen am 23.06.2017)
 - [41] https://de.slideshare.net/j_singh/nosql-and-mapreduce (abgerufen am 23.06.2017)
 - [42] Dean, Jeffrey; Ghemawat, Sanjay. "MapReduce: Simplified Data Processing on Large Clusters", Google, Inc. (2004)
 - [43] <https://db-engines.com/de/article/MapReduce> (abgerufen am 23.06.2017)
 - [44] <https://dba.stackexchange.com/questions/52632/difference-between-sharding-and-replication-on-mongodb> (abgerufen am 24.06.2017)
 - [45] <http://blog.zuehlke.com/skalierung-von-datenbanken/> (abgerufen am 24.06.2017)

- [46] <https://www.quora.com/What-is-the-difference-between-replication-partitioning-clustering-and-sharding> (abgerufen am 24.06.2017)
- [47] <https://www.gi.de/service/informatiklexikon/detailansicht/article/partitionierung-von-datenbanktabellen.html> (abgerufen am 24.06.2017)
- [48] <https://www.techopedia.com/definition/17/clustering-databases> (abgerufen am 24.06.2017)
- [49] <https://de.wikipedia.org/wiki/Rechnerverbund> (abgerufen am 24.06.2017)
- [50] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.Replikation (abgerufen am 24.06.2017)
- [51] <http://www.itwissen.info/Replikation-replication.html> (abgerufen am 24.06.2017)
- [52] <http://www.datenbanken-verstehen.de/lexikon/replication-server/> (abgerufen am 24.06.2017)
- [53] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.Sharding (abgerufen am 24.06.2017)
- [54] <https://db-engines.com/de/article/Sharding> (abgerufen am 24.06.2017)
- [55] <https://www.it-visions.de/glossar/alle/6301/Sharding.aspx> (abgerufen am 24.06.2017)
- [56] <https://www.toptal.com/big-data/consistent-hashing> (abgerufen am 25.06.2017)
- [57] <http://theory.stanford.edu/~tim/s16/l/l1.pdf> (abgerufen am 25.06.2017)
- [58] https://en.wikipedia.org/wiki/Consistent_hashing (abgerufen am 25.06.2017)
- [59] https://de.wikipedia.org/wiki/Konsistente_Hashfunktion (abgerufen am 25.06.2017)
- [60] <http://www.paperplanes.de/2011/12/9/the-magic-of-consistent-hashing.html> (abgerufen am 25.06.2017)
- [61] <http://www.mikeperham.com/2009/01/14/consistent-hashing-in-memcache-client/> (abgerufen am 25.06.2017)
- [62] <http://www.tom-e-white.com/2007/11/consistent-hashing.html> (abgerufen am 25.06.2017)
- [63] <http://www.tomkleinpeter.com/2008/03/17/programmers-toolbox-part-3-consistent-hashing/> (abgerufen am 25.06.2017)
- [64] <http://michaelnielsen.org/blog/consistent-hashing/> (abgerufen am 25.06.2017)
- [65] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/ConsistentHashing (abgerufen am 25.06.2017)
- [66] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.NoSQL (abgerufen am 27.06.2017)
- [67] <http://www.datenbanken-verstehen.de/lexikon/nosql/>
- [68] <http://nosql-database.org/> (abgerufen am 27.06.2017)
- [69] <https://db-engines.com/de/article/NoSQL> (abgerufen am 27.06.2017)
- [70] <http://www.agiledata.org/essays/impedanceMismatch.html> (abgerufen am 28.06.2017)
- [71] <http://www.odbms.org/wp-content/uploads/2013/11/023.01-Shusman-The->

- [Impedance-Mismatch-2002.pdf](#) (abgerufen am 28.06.2017)
- [72] https://dbs.uni-leipzig.de/file/seminar_1112_tran_ausarbeitung.pdf (abgerufen am 28.06.2017)
 - [73] <http://www.sts.tu-harburg.de/teaching/ws-98.99/DBIS/6-dbis.pdf> (abgerufen am 28.06.2017)
 - [74] http://pi.informatik.uni-siegen.de/lehre/1999w/1999w_proseminar/ausarbeitungen/thema7/Objektrelationale_Datenbanken.pdf (abgerufen am 28.06.2017)
 - [75] <http://wwwbayer.in.tum.de/lehre/WS2001/DBS-kossmann/folienD.pdf> (abgerufen am 28.06.2017)
 - [76] <http://www.itwissen.info/Objektrelationale-Datenbank-object-relational-database-ORD.html> (abgerufen am 28.06.2017)
 - [77] <http://www.itwissen.info/Objektorientierte-Datenbank-object-oriented-database-OODB.html> (abgerufen am 28.06.2017)
 - [78] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.Objektdatenbank (abgerufen am 28.06.2017)
 - [79] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.Objektrelationale-Datenbank (abgerufen am 28.06.2017)
 - [80] <http://www.datenbanken-verstehen.de/datenbank-grundlagen/datenbankmodell/objektorientiertes-datenbankmodell/> (abgerufen am 28.06.2017)
 - [81] <https://de.wikipedia.org/wiki/Objektdatenbank> (abgerufen am 28.06.2017)
 - [82] <https://db.in.tum.de/~grust/teaching/ss06/DB2/db2-02.pdf> (abgerufen am 28.06.2017)
 - [83] <http://www-lehre.inf.uos.de/~dbs/2005/skript/node154.html> (abgerufen am 28.06.2017)
 - [84] https://de.wikipedia.org/wiki/Objektrelationale_Datenbank (abgerufen am 28.06.2017)
 - [85] https://en.wikipedia.org/wiki/Object-relational_database (abgerufen am 28.06.2017)
 - [86] https://en.wikipedia.org/wiki/Object_database (abgerufen am 28.06.2017)
 - [87] <https://db-engines.com/de/ranking/object+oriented+dbms> (abgerufen am 01.07.2017)
 - [88] <https://db-engines.com/de/ranking/relational+dbms> (abgerufen am 01.07.2017)
 - [89] <https://db-engines.com/de/article/Graph+DBMS> (abgerufen am 29.06.2017)
 - [90] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.Graphendatenbank (abgerufen am 29.06.2017)
 - [91] http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.Graphenmodell (abgerufen am 29.06.2017)
 - [92] https://en.wikipedia.org/wiki/Graph_database (abgerufen am 29.06.2017)
 - [93] <https://de.wikipedia.org/wiki/Graphdatenbank> (abgerufen am 29.06.2017)

- [94] <http://www.datenbanken-verstehen.de/lexikon/graphdatenbanken/> (abgerufen am 29.06.2017)
- [95] <http://www.itwissen.info/Graphen-Datenbank-graph-database.html> (abgerufen am 29.06.2017)
- [96] <https://neo4j.com/developer/graph-database/> (abgerufen am 29.06.2017)
- [97] <https://db-engines.com/de/ranking/graph+dbms> (abgerufen am 01.07.2017)
- [98] <http://www.datenbanken-verstehen.de/lexikon/spaltenorientierte-datenbanken/> (abgerufen am 30.06.2017)
- [99] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/SpaltenorientierteDatenbank (abgerufen am 30.06.2017)
- [100] https://de.wikipedia.org/wiki/Spaltenorientierte_Datenbank (abgerufen am 30.06.2017)
- [101] https://en.wikipedia.org/wiki/Column-oriented_DBMS (abgerufen am 30.06.2017)
- [102] <http://www.itwissen.info/columnar-database-Spaltenorientierte-Datenbank.html> (abgerufen am 30.06.2017)
- [103] <http://eliteinformatiker.de/2011/08/07/zeilenorientierte-und-spaltenorientierte-datenbanken> (abgerufen am 30.06.2017)
- [104] <http://www.searchenterprisesoftware.de/definition/Spaltenorientierte-Datenbank> (abgerufen am 30.06.2017)
- [105] <https://www.gi.de/service/informatiklexikon/detailansicht/article/spaltenorientierte-datenbanken.html> (abgerufen am 30.06.2017)
- [106] <http://database.guide/what-is-a-column-store-database/> (abgerufen am 30.06.2017)
- [107] <https://db-engines.com/de/ranking/wide+column+store> (abgerufen am 01.07.2017)
- [108] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/DokumentenorientierteDatenbank (abgerufen am 01.07.2017)
- [109] https://de.wikipedia.org/wiki/Dokumentenorientierte_Datenbank (abgerufen am 01.07.2017)
- [110] https://en.wikipedia.org/wiki/Document-oriented_database (abgerufen am 01.07.2017)
- [111] <http://eliteinformatiker.de/2011/05/18/nosql-document-store-couchdb-mongodb> (abgerufen am 01.07.2017)
- [112] <https://db-engines.com/de/article/Document+Stores> (abgerufen am 01.07.2017)
- [113] <http://database.guide/what-is-a-document-store-database/> (abgerufen am 01.07.2017)
- [114] <http://insights.dice.com/2012/06/04/nosql-document-storage-benefits-drawbacks/> (abgerufen am 01.07.2017)
- [115] <https://db-engines.com/de/ranking/document+store> (abgerufen am 01.07.2017)
- [116] http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/KeyValueSysteme (abgerufen am 01.07.2017)

am 02.07.2017)

- [117] <https://de.wikipedia.org/wiki/Schl%C3%BCssel-Werte-Datenbank> (abgerufen am 02.07.2017)
- [118] https://en.wikipedia.org/wiki/Key-value_database (abgerufen am 02.07.2017)
- [119] <http://www.datenbanken-verstehen.de/lexikon/key-value-datenbanksysteme/> (abgerufen am 02.07.2017)
- [120] <https://db-engines.com/de/article/Key-Value+Stores> (abgerufen am 02.07.2017)
- [121] <http://eliteinformatiker.de/2011/06/01/nosql-key-value-datenbanken-memcachedb-project-voldemort-redis> (abgerufen am 02.07.2017)
- [122] <http://database.guide/what-is-a-key-value-database/> (abgerufen am 02.07.2017)
- [123] <https://db-engines.com/de/ranking/key-value+store> (abgerufen am 01.07.2017)

Verwendete Programme für die Abbildungen:

- UMLet: <http://www.umlet.com/>
- Microsoft Word: <https://products.office.com/de-de/word>

Single Page Application

Autor: Alexander Schwietert

Single Page Application

Eine Single Page Application ist eine Web-Applikation, die lediglich aus einem einzigen HTML Dokument besteht und soll dadurch eine Nutzererfahrung ähnlich der einer Desktop-Applikation bieten. Der komplette Seiteninhalt (HTML, Javascript und CSS) wird dabei entweder beim initialen Laden der Seite übermittelt oder einzelne logische Seiten innerhalb der SPA dynamisch nachgeladen, in Abhängigkeit mit den Aktionen des Nutzers. Mit SPAs wird die Grundlage für eine Rich-Client bzw. Fat-Client-Architektur geboten, in der ein Großteil der Ausführung einer Webanwendung auf Clientseite geschieht und somit die Serverlast reduziert wird.

Eigenschaften

- Der Sitzungszustand wird bei dem Client gespeichert, der Server stellt hingegen lediglich noch die Nutzdaten zur Verfügung. Ein erneutes Laden der SPA ist mit einem Neustart einer Anwendung gleichzusetzen.
- Serverseitige Funktionalitäten daher Zustandslos
- "offline-friendly", Nutzungsdaten können im local storage zwischengespeichert werden, die SPA kann so auch bei Verbindungsverlust weiter laufen
- WebClient fungieren als unabhängige Einheit mit verschiedenen Services und kann somit selbstständig auf Benutzerinteraktionen reagieren. Dadurch wird der Datenverkehr zum Server verringert
- WebClient und Serverseite können separat entwickelt werden, lediglich die Schnittstellen für die Services müssen vorher definiert werden.

Entwurfsmuster

Entwurfsmuster beschreiben einen abstrakten Lösungsansatz für Problemstellungen in der Softwarearchitektur und -entwicklung, der für ein individuelles Problem entsprechend wiederverwendet werden kann.

Model-View-Controller

Das MVC Muster beschreibt die Trennung einer Software-Architektur in drei Komponenten. Das Datenmodell (Model), die Präsentation (View) und die Programmsteuerung / Logik (Controller). (Lahres 2006)

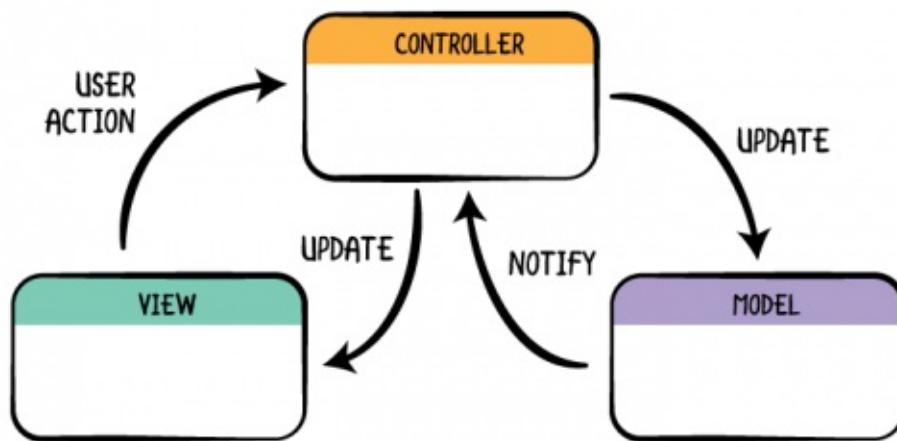


Abbildung 1: MVC (Peres 2016)

Model

Das Model enthält die darzustellenden Daten und ist von den anderen beiden Schichten unabhängig. Das Model publiziert entsprechende Änderungen an seinen Daten, welche von den lauschenden Komponenten registriert werden. Dies geschieht nach dem sogenannten Beobachter-Muster.

View

Die View Komponente subscribed auf die Daten des Models und updated seine Sichten entsprechend der Daten, sollten sie geändert werden. Mit der View Komponente werden die Benutzerinteraktion realisiert. Diese kennt ebenfalls den Controller, ist jedoch nicht für die Weiterverarbeitung der Daten zuständig.

Controller

Der Controller enthält in der Regel die Geschäftslogik (diese kann auch direkt im Model liegen, je nach Programmiersprache und Implementierung des MVC) und kann mehrere Views verwalten, wobei zu jeder View eine Steuerung existiert. Der Controller sorgt dafür, dass die Benutzerinteraktionen in der View ihre entsprechende Wirkung haben, verändert gegebenenfalls die View und gibt z.B. eingegebene Daten aus der View weiter an das Model.

Model-View-Presenter

Das MVP-Muster ist aus dem MVC-Muster heraus entstanden und setzt auf einen neuen Ansatz, um das Model komplett von der View zu trennen und über einen sogenannten Presenter zu verbinden. Dies soll eine bessere Testbarkeit als auch eine bessere Übersicht bei der Implementierung zur Folge haben. (Fowler 2006a)

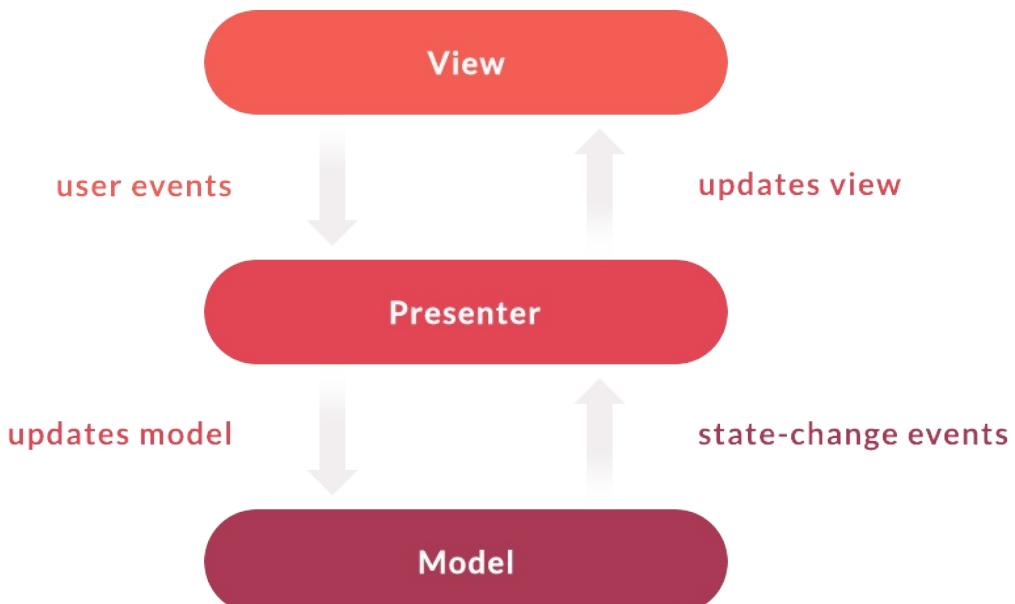


Abbildung 2: MVP (Ziomacki 2016)

Model

Das Modell stellt die Logik der Ansicht bzw. die Geschäftslogik dar und kennt dabei weder die View noch den Presenter. Es muss jedoch alle Funktionalitäten implementiert haben um die View betreiben zu können. Die Steuerung des Models erfolgt durch den Presenter.

View

Die View kennt ebenfalls weder das Model noch den Presenter und ist nur für die Ein- und Ausgabe von Daten zuständig, sowie die allgemeine Darstellung der Software. Die Steuerung der View erfolgt durch den Presenter.

Presenter

Der Presenter ist das Bindeglied zwischen View und Model, welche jeweils ihre Schnittstellen definieren, die anschließend vom Presenter miteinander verknüpft werden. Die Schnittstellen beschreiben dabei den genauen Aufbau der anderen zwei Komponenten. Dies resultiert in einer vollkommenen Austauschbarkeit von View und Model.

Supervising Controller

In einer Abwandlung des MVP-Musters kann die View zusätzlich für die Datensynchronisation zuständig sein, wodurch der Presenter sich lediglich um alle anderen Abläufe zwischen Model und View kümmert. Der Presenter kann zum data-binding hierbei Klassen zur Verfügung stellen, die View als auch Model bekannt sind, wodurch weiterer Synchronisationsaufwand entfällt und bei veränderten Daten die View direkt ihre Sicht aktualisiert. (Fowler 2006b)

Model-View-ViewModel

Das MVVM-Muster ist eine weitere abgewandelte Variante des MVC. Es ist eine Spezialisierung des MVP-Musters und kommt ebenfalls ohne Controller aus, da auch hier Zustand und Verhalten die Presenter bzw. ViewModel Komponenten regeln. Die Testbarkeit wird verbessert und die Arbeitsteilung zwischen UI-Designer und Programmierern gefördert, wodurch separate Teams gleichzeitig an einer Software entwickeln können. (Kühnel 2013)

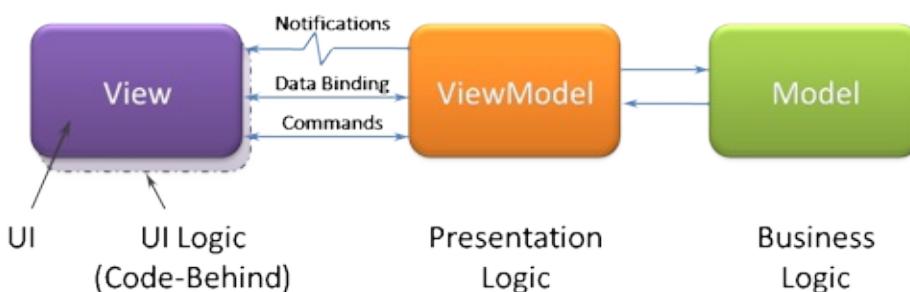


Abbildung 3: MVVM [Hill2009]

Model

Das Model hält die komplette Geschäftslogik sowie die dem Nutzer präsentierten Daten inne, welche nach Manipulation des Nutzers aktualisiert und validiert werden können.

View

Die View enthält alle Elemente zur Präsentation von Software und Daten. Sie bindet sich an Funktionen und Objekte des ViewModel um Daten darzustellen und Interaktionen der Nutzer weiterzugeben.

ViewModel

Das ViewModel dient als Bindeglied zwischen View und Model, wobei das ViewModel die View nicht kennt. Es stellt der View öffentliche Methoden und Eigenschaften zur Verfügung, über die die View Daten holen kann und Ereignisse in der View weiter an das ViewModel gibt. Das ViewModel wiederum ruft Methoden und Elemente des Models aus um Informationen auszutauschen.

Flux

Flux ist eine Architektur die erstmals 2013 von Facebook in Zusammenhang mit react.js vorgestellt wurde. Bei react bestehen Komponenten aus HTML, JavaScript Code als auch CSS und es herrscht somit keine Trennung von View und Model, wie man es aus den vorherigen Mustern kennt. (*Deutsch 2015*)

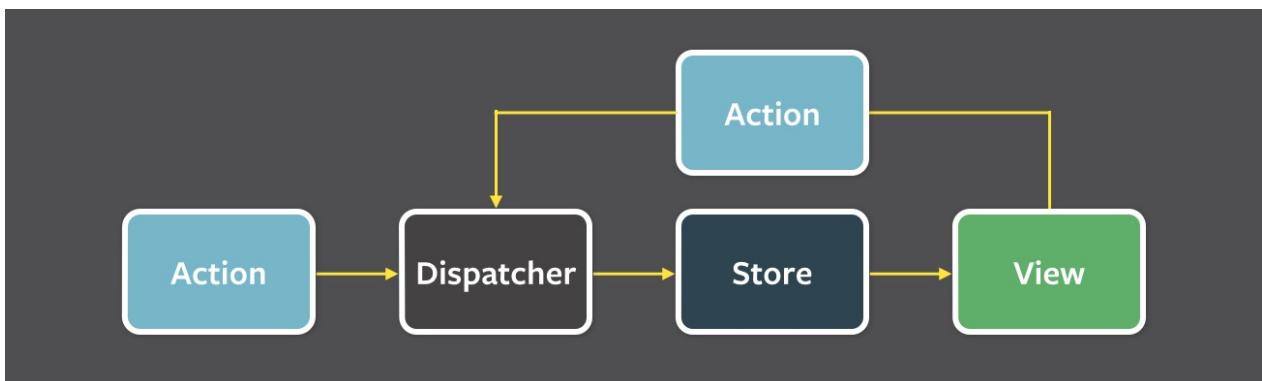


Abbildung 4: Flux (Zeigermann 2015)

Action

Eine Action ist ein normales JavaScript Objekt, welches aus seinem Payload (beliebig viele Attribute / Daten) sowie dem festen Attribut "type" besteht, welches in der Regel ein konstanter String ist. Über das type-Attribut können die Stores erkennen, ob es sich um eine für sie relevante Aktion handelt.

Sogenannte ActionCreators werden aus den Views (React Komponenten) heraus aufgerufen, um eine Action zu erstellen und diese zum Dispatcher weiter zu reichen, welches sie den jeweiligen Stores zuordnet. Die Views wiederum sind mit den Stores verbunden und werden bei Veränderungen der Daten im Store durch Actions neu gerendert.

Dispatcher

Der Dispatcher ordnet lediglich die Action Objekte einem Store zu, dabei behandelt er immer nur ein Objekt gleichzeitig.

Store

In Stores werden die Logik einer Applikation verwaltet und fungieren als Datenquellen für die Komponenten. Sie empfangen Actions, welche die Daten der Stores verändern können, woraufhin ein Store seine React Komponenten benachrichtigt.

View

Die React Komponenten subscriben auf ihre Stores und rendern sich bei Veränderungen gegebenenfalls neu. Es wird zwischen einfachen und smarten Komponenten unterschieden.

Einfache Komponenten sind abstrakter und leichter wieder zu verwenden, erhalten ihre Daten und selbst ausgelöste Actions als Properties.

Smarte Komponenten wiederum haben Funktionen um auf Stores zu subscriben und Actions zu dispatchen.

Da der Datenfluss von den Views über Actions und Dispatcher zu den Stores verläuft, spricht man auch von einem **Single-Directional-Dataflow**.

Angular

Angular (auch Angular2 oder Angular2+ genannt) ist ein open-source, auf JavaScript / Typescript basiertes Softwareframework, zur Entwicklung von Single Page Applications und der Nachfolger von AngularJS. Die Community-getriebene Entwicklung, bestehend aus verschiedenen Einzelpersonen und Unternehmen, wird vom Angular Team bei Google angeführt.

Grundlegende Architektur

Eine Angular-Applikation ist modular aufgebaut (siehe Abbildung 1: Angular Architektur). Ein wesentlicher Bestandteil sind die sogenannten Komponenten, diese beinhalten die in JavaScript oder Typescript geschriebene Logik, die wiederum mit einem HTML Template und optional einer (S)CSS Datei verknüpft wird. Das besondere hierbei ist das sogenannte 2-way-databinding, wodurch Variablen und Funktionen innerhalb der Komponente direkt mit den Elementen im HTML Markup verknüpft werden und Änderungen sofort wechselseitig übertragen und verarbeitet werden. Eine Komponente ist vergleichbar mit einer klassischen HTML Seite einer nicht Single Page Application. (Angular 2017a)

Globale Funktionalitäten werden wiederum in Service Modulen geschrieben, welche beim Start der SPA einmal als Singleton initialisiert werden und deren Funktionen und Variablen fortan von jeder Komponenten verwendet werden können.

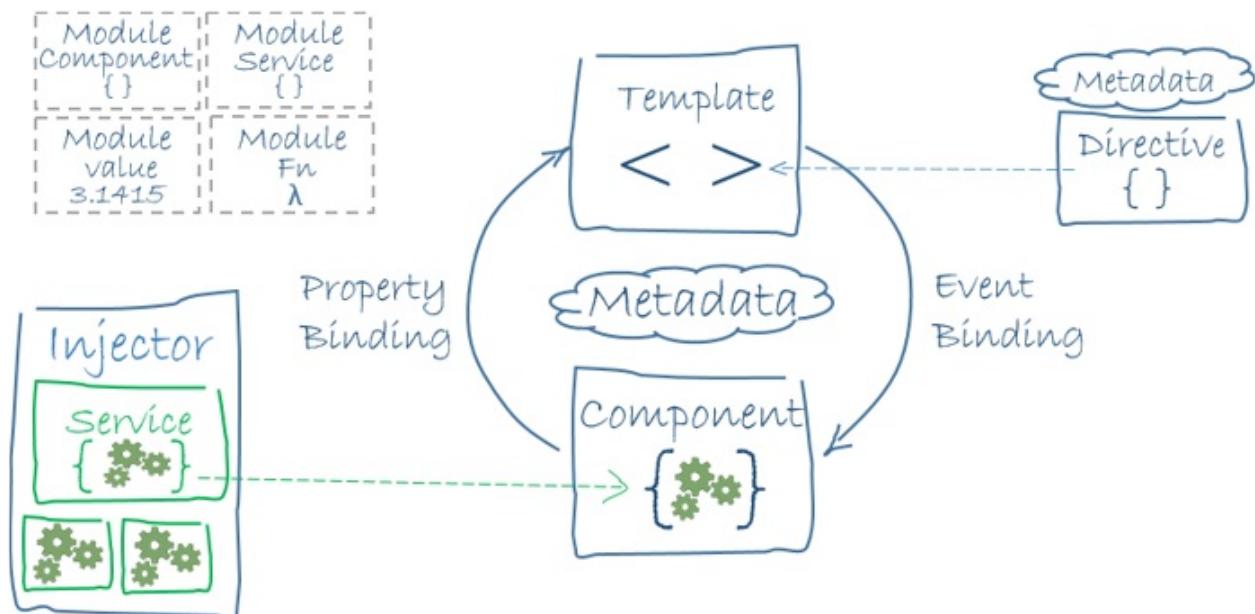


Abbildung 1: Angular Architektur (Angular 2017a)

Module

Die Module werden als Angular Module oder NgModule bezeichnet. Jede Applikation besteht aus mindestens einem Modul, dem sogenannten root module, in der Regel **AppModule** genannt.

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:   [ AppComponent ]
})
export class AppModule { }
```

Code-Ausschnitt 1: AppModule

In diesem Modul werden alle anderen notwendigen Module geladen, dazu gehören die Module bzw. Bibliotheken die Angular selber stellt, weitere Software die z.B. über NPM hinzugefügt wurde, sowie die eigens geschriebenen Komponenten und Services der Applikation (siehe Code-Ausschnitt 1: AppModule).

- **@NgModule** ist ein Decorator, der alle Module der Applikation als ein MetaData-Objekt zusammenfasst.
- **Imports** listet alle Module, deren exportierte Klassen und Funktionalitäten in den Komponenten gebraucht werden (z.B. das RouterModule, dessen Funktionen zum navigieren zwischen Komponenten nötig ist)
- **Providers** listet die Service Module, die initial instanziert werden und in allen anderen Modulen aufgerufen werden können
- **Declarations** listet die View-Klassen der Applikation, die vom User geschriebenen Komponenten, Direktiven und Pipes.
- **Exports** listet Deklarationen, die in den Templates anderer Module aufgerufen werden können
- **bootstrap** definiert die root Komponente, der Ausgangspunkt der Applikation und somit die erste geladene Sicht, über die auf die weiteren Komponenten zugegriffen werden kann.

Beim Starten wird die SPA über das root module gebaut und aufgerufen, welches die SPA anschließend dem Browser präsentiert. Dies geschieht nach gegebenen Konventionen in einer übergeordneten Datei, wie der "main.ts" (siehe Code-Ausschnitt 2: main.ts Beispiel).

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app/app.module';

platformBrowserDynamic().bootstrapModule(AppModule);
```

Code-Ausschnitt 2: main.ts Beispiel

Komponenten

Eine Komponente kontrolliert eine View innerhalb der SPA und ist daher mit einem HTML Template verknüpft. Eine Komponente kann dabei auf Funktionen der zuvor im root module deklarierten und importierten Module zugreifen, sowie eigenen Attribute und Funktionen definieren (siehe Code-Ausschnitt 3: Beispielkomponente).

```
@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css']
})

export class SearchComponent implements OnInit {

  searchString: String = "";
  results: String[] = [];

  constructor(private _searchService: SearchService, private route: ActivatedRoute) { }

  ngOnInit() {
    //stuff happens when components gets initiated
  }

  startSearch() {
    //...
  }
}
```

Code-Ausschnitt 3: Beispielkomponente

In einem Konstruktor werden Services für den lokalen Zugriff innerhalb der Komponente übergeben, mit Interfaces wie OnInit können wiederum die Funktionalitäten der Komponenten erweitert werden.

MetaData

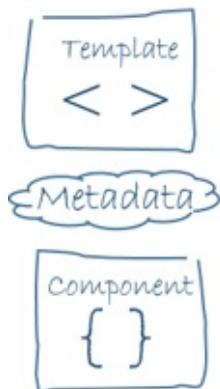


Abbildung 2: MetaData-Verknüpfung zwischen Template und Komponente (Angular 2017a)

Dem **@Component** Decorator können verschiedene Attribute mitgegeben werden, unter anderem ein Verweis auf eine HTML und eine CSS Datei. Der **selector** ist der HTML-Tag, über den die Komponenten innerhalb einer anderen Komponente aufgerufen werden könnte. Erst diese Angaben machen aus der Klasse SearchComponent eine Komponente im Angular Kontext.

Input und Output Attribute

Komponenten können andere Komponenten innerhalb ihres Templates aufrufen und so in einer Eltern-Kind-Beziehung zueinanderstehen. Beim Aufruf des Kindes über einen selector können Input-Attribute übergeben werden. Gleichzeitig können Output-Events des Kindes an Funktionen der Eltern-Komponenten gebunden werden (siehe Code-Ausschnitt).

(Angular 2017b)

```
<result-app [result]="resultElement" (deleteResult)="deleteR($event)">
</result-app>
```

Code-Ausschnitt 4: Aufruf einer Kind-Komponente im Template

In der Kind-Komponente werden diese Attribute mit einem **@Input** oder **@Output** deklariert. Ein Output-Event wird manuell innerhalb einer Funktion getriggert und kann ein oder mehrere Objekte mitgeben (siehe Code-Ausschnitt).

```
@Input() result: Result;
@Output() deleteResult = new EventEmitter<Result>();
```

Code-Ausschnitt 5: Input und Output Parameter einer Kind-Komponente

Im Falle der beiden Code-Ausschnitte wird zuerst die Variable `resultElement` der Eltern-Komponente dem Kind mitgegeben, welche es in der eigenen Variable `result` speichert. Wenn das Kind das Event `deleteResult` wirft, gibt es ein Objekt vom Typ `Result` mit, welches wiederum die `deleteR` Funktion in der Eltern-Komponente auruft.

Lifecycle Hooks

Hook	Beschreibung / Wann
ngChanges()	Wird noch vor OnInit das erste Mal aufgerufen und weiterhin immer dann, wenn eine 2-way gebundene Variabel geändert wird
ngOnInit()	Wird initial einmalig aufgerufen
ngDoCheck()	Erkennt Änderungen die Angular selber nicht erkennt, wird in Intervallen aufgerufen oder manuell getriggert
ngAfterContentInit()	Einmalig nach dem ersten DoCheck(), wenn alle externen Inhalte in die View geladen wurden
ngAfterContentChecked()	Nach ContentInit und jedem erneuten DoCheck()
ngAfterViewInit()	Einmalig nach dem ersten ContentChecked, wenn die eigene View der Komponente sowie deren Kinder gecheckt wurden
ngAfterViewChecked()	Nach AfterViewInit und jedem erneuten AfterContentChecked()
ngOnDestroy()	Bevor Komponente zerstört wird, um Observables zu unsubscriben und ein Speicherleck zu vermeiden

Tabelle 1: Lifecycle Hooks einer Komponente

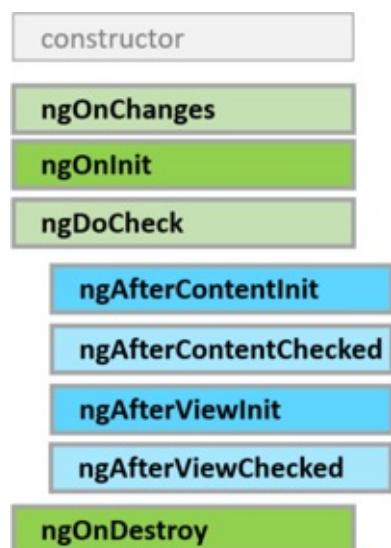


Abbildung 3: Reihenfolge der Lifecycle Hooks (Angular 2017c)

Template

Das Template kann entweder in einer separaten Datei definiert werden oder direkt als Attribut des Decorators, in Form eines Strings, in die Komponente geschrieben werden.

```
<md-input-container>
  <input mdInput [(ngModel)]="searchString" [placeholder]="'SEARCH.SEARCH' | translate"
  [type]="text" [mdAutocomplete]="auto">
</md-input-container>

<result-app *ngFor="let searchresult of results" [result]="searchresult"></result-app>
```

Code-Ausschnitt 6: Auszug aus HTML Template mit data-binding

result-app ist in diesem Fall der selector einer anderen Komponente, die in einer Schleife mehrfach aufgerufen und in der View generiert wird (siehe Code-Ausschnitt 6: Auszug aus HTML Template mit data-binding). Das in Klammern stehende **[result]** ist wiederum eine Input-Variable innerhalb der result-Komponente.

Result-App ist in diesem Fall eine Kind-Komponenten von der SearchComponent und wird innerhalb des Template der Eltern-Komponenten aufgerufen. Dieses Verhalten ist hierarchisch organisiert (siehe Abbildung 4: Hierarchie der Komponenten)

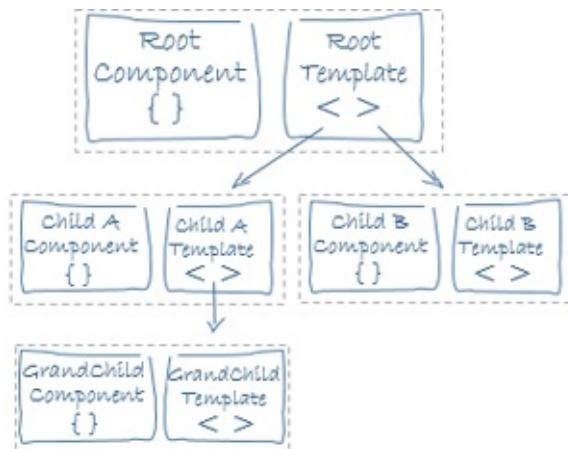


Abbildung 4: Hierarchie der Komponenten (Angular 2017a)

Structural Directives

Strukturelle Direktiven werden genutzt um den DOM zu manipulieren, Elemente hinzuzufügen oder zu entfernen. Folgende Elemente sind die, die am häufigsten verwendet werden.

***ngIf** wird an eine Boolean-Variable oder ein Statement gebunden, das entweder true oder false ist. Das Element und alle Kind-Elemente werden entsprechend dem DOM hinzugefügt oder entfernt.

```
<div *ngIf="isActive"></div>
```

Code-Ausschnitt 7: *ngIf Beispielcode

***ngFor** wird verwendet um über eine Liste oder Array zu iterieren und entsprechende Elemente im DOM auszugeben. Das HTML-Element an dem die Liste hängt wird entsprechend häufig erzeugt. Optional kann der Index des Elementes an eine Variable gebunden werden.

```
<div *ngFor="let result of results;let i=index;"> </div>
```

*Code-Ausschnitt 8: *ngFor Beispielcode*

***ngSwitch** entspricht der Logik seines JavaScript Pendants und fügt jenes Element dem DOM hinzu, welches dem Switch-Statement entspricht.

```
<div [ngSwitch]="result.price">
  <div *ngSwitchCase="50">Ist okay.</div>
  <div *ngSwitchCase="100">Viel zu teuer!</div>
</div>
```

*Code-Ausschnitt 9: *ngSwitch Beispielcode*

Attribute Directives

Diese Direktiven funktionieren als klassische HTML-Attribute, die Variablen aus der Komponente mit den Attributen verknüpfen und so das Verhalten des HTML-Elements beeinflussen. Neben ngModel (welches im folgenden Kapitel genauer beschrieben wird) wird auf 2 relevante Direktiven eingegangen. (*Angular 2017b*)

ngClass fügt ein komplettes Set an potenziellen Klassen hinzu, indem eine Liste aus Schlüssel-Wert Paaren bestimmt welche Klassen aktuell aktiv sein sollten für das jeweilige Element.

```
<div [ngClass]="currentClasses">This div is initially saveable, unchanged, and special
</div>
```

Code-Ausschnitt 10: ngClass Beispielcode

ngStyle funktioniert nach der gleichen Methode, verschiedene Styles können wiederum durch weitere Variablen beeinflusst werden, die beim Setzen der Schlüssel-Wert Paare abgerufen werden.

```
<div [ngStyle]="currentStyles"> </div>
```

```

this.currentStyles = {
  'font-style': this.canSave ? 'italic' : 'normal',
  'font-weight': !this.isUnchanged ? 'bold' : 'normal',
  'font-size': this.isSpecial ? '24px' : '12px'
};

```

Code-Ausschnitt 11: *ngStyle* Beispielcode

Data binding

Durch das data-binding werden Elemente und Ereignisse aus DOM und Komponente verknüpft. Dies kann in eine oder in beide Richtung geschehen (siehe Abbildung 5: Data Binding).

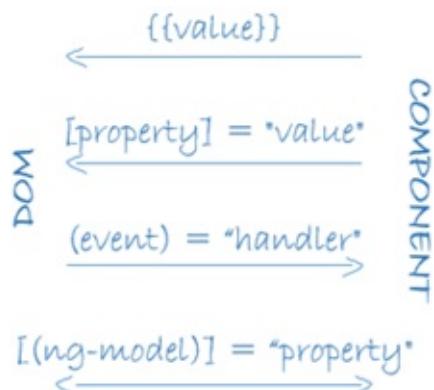


Abbildung 5: Data Binding (Angular 2017a)

1-Way-Data-Binding

```

<li></li>
<result-app [result]="result"></result-app>
<li (click)="search(seachstring)"></li>

```

Code-Ausschnitt 12: 1-Way-Data-Binding Beispielcode

- **geschweifte Klammern** werden dafür genutzt um den Wert einer Variable direkt im HTML anzuzeigen
- **[Attribute]** eines HTML Elements in eckigen Klammern werden Werte einer Variable aus der Komponente zugeschrieben, z.B. `[class] = "klassenvariable"`
- **(event)** wiederum kann eine Funktion innerhalb der Komponenten aufrufen und ggf. Variablen übergeben, alternativ kann auch die Funktion direkt in das HTML geschrieben werden z.B. `(click) = "variable='false'"`

2-Way-Data-Binding

Über **[(ngModel)]** können Variablen z.B. mit einem Input-Feld verknüpft werden, die Variable ändert sich entsprechend im Template als auch in der Komponente, wenn der Nutzer eine Eingabe macht. Zusätzlich kann über das **ngModelChange**-Event eine Funktion an die Änderung des gebundenen Objektes gehängt werden.

Angular verarbeitet alle data-bindings pro **JavaScript-Event-Circle** und traversiert dabei von der root Komponenten bis hinunter zum letzten Kind. Das 2-way-data-binding ist daher auch besonders für die Verknüpfung von Eltern und Kind-Komponenten wichtig, wo Objekte als Referenzen übergeben werden können. Sollte ein übergebenes Objekt in der Kind-Komponente verändert werden, gilt dies auch für das Objekt in der Eltern-Komponente (siehe auch Abbildung 6: Eltern-Kind-Data-Binding).

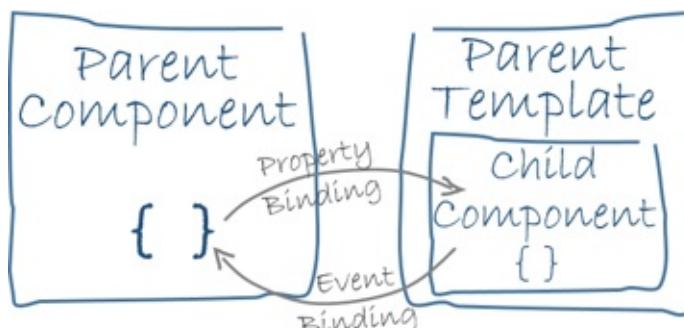


Abbildung 6: Eltern-Kind-Data-Binding (Angular 2017a)

Pipes

Pipes transformieren einen gegebenen Input und werden direkt im Template mit dem Pipe Operator "|" aufgerufen. Dieser kann direkt hinter einem Value Aufruf stehen oder hinter einer strukturellen Direktive, wie einem *ngFor. Die Pipe kann dabei beispielsweise als Filter fungieren (siehe Code-Ausschnitt). (Angular 2017d)

```

<div>Value Aufruf: {{title | uppercase}}</div>
<div>Value Aufruf: {{title | uppercase | lowercase}}</div>
<div *ngFor="let Offer of Offers | filterList: filterBy">
  
```

Code-Ausschnitt 13: Pipes Operator Beispielcode

Hinter dem Pipe Operator wird der Name der Pipe genannt, wobei auch mehrere Pipes hintereinander gereiht werden können. Zusätzlich kann durch Doppelpunkte die zu übergebenen Parameter definiert werden. Falls dies nicht geschieht, wird die Variable vor dem Operator übergeben.

Built-in

Pipe	Format	Beschreibung
DatePipe	date_expression \	date[:format]
LowerCasePipe	value \	lowercase
UpperCasePipe	value \	uppercase
CurrencyPipe	number_expression \	currency:currencyCode:symbolDisplay:digitInfo
PercentPipe	number_expression \	percent:digitInfo

Tabelle 2: Built-In Pipes

Eigene Pipes

Eigens geschriebene Klassen können mit @Pipe zu einer Pipe deklariert werden. Dabei wird der Name definiert, der später hinter dem Pipe Operator angegeben wird, und optional, ob die Pipe pure oder impure ist. Standardmäßig werden Pipes als pure definiert.

```

@Pipe({
  name: 'filterResults',
  pure: true
})
export class SortListPipe implements PipeTransform {

  transform(array: Array<any>, filterby: string): Array<any> {
  }
}

```

Code-Ausschnitt 14: Eigene Pipe Komponente

Aufgerufen wird eine Methode, die das PipeTransform-Interface implementiert, welche beliebig viele Input-Parameter (mindestens einen) verarbeiten kann. Hier kann zum Beispiel ein Array übergeben werden, dessen Objekte nach einem bestimmten Wert gefiltert werden sollen. Anschließend wird das neue Array zurück gegebenen und in der View der Komponente dargestellt.

Pure Pipes werden lediglich aufgerufen, wenn sich der triviale Inputparameter ändert (String, Boolean, Integer etc) oder ein komplett neues Objekt oder Array eingespeist wird. Die Pipe wird jedoch nicht getriggert, sollte ein Attribut eines Objektes geändert oder einem Array ein Element hinzugefügt werden.

Impure Pipes hingegen reagieren auf jede Veränderung, weswegen sie grade bei komplexeren Aufgaben und größeren Inputs die Performance negativ beeinflussen können.

Services

Services sind Klassen, die an sich nichts Angular-Spezifisches beinhalten (siehe Code-Ausschnitt 15: Beispiel des Search Service). Es ist jedoch guter Stil alle nicht trivialen Aufgaben in diese Service-Klassen auszulagern und ihre Funktionen innerhalb der Komponenten zu nutzen, statt die komplette Logik in die Komponenten zu schreiben.

```

@Injectable()
export class SearchService {

  constructor(private http: Http, private _router: Router) { }

  searchOffer(params:any) {
    //....
  }
}

```

Code-Ausschnitt 15: Beispiel des Search Service

Dependency injection

Mit der dependency injection wird eine neue Instanz einer Klasse erzeugt, mit samt allen ihren Abhängigkeiten. Dies sind in der Regel Services. Alle zu injizierenden Klassen werden dem Konstruktor einer Komponente als Parameter übergeben.

Wenn Angular eine Komponenten aufruft werden zuerst ihre Abhängigkeiten überprüft, ein Injektor hält in einem Container alle bereits initialisierten Klassen bereit (siehe Abbildung 7: Injektor). Sollte eine benötigte Klasse noch nicht in diesem Container vorhanden sein, wird eine neue Instanz erstellt.

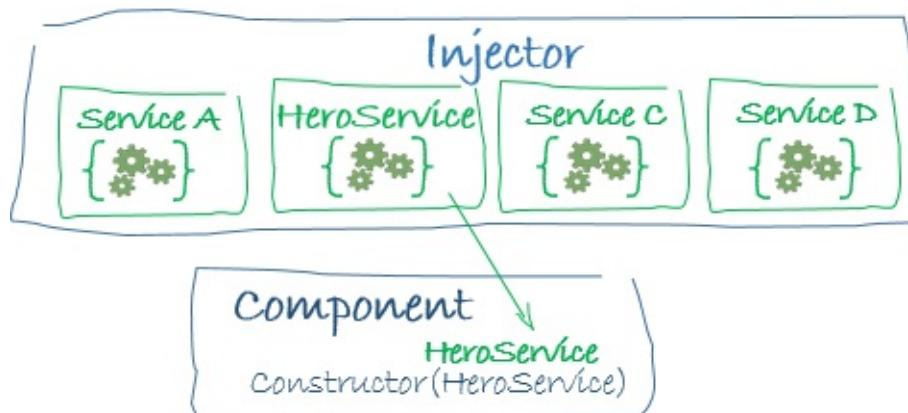


Abbildung 7: Injektor (Angular 2017a)

Um eine neue Instanz zu erstellen, muss zuvor ein Provider für die Klasse definiert worden sein. Ein Provider kann einen Service erstellen oder zurückgeben. Dieser kann, wie bereits zuvor erwähnt, im root module angegeben werden oder als MetaData im @Component Decorator. Wenn der Provider im root module definiert ist, werden alle Komponenten auf die gleiche Instanz zugreifen.

Server Kommunikation

Angular bietet unter der **@angular/http** Bibliothek sein eigenes HTTP Modul, welches optional über das root module in eine SPA integriert werden kann. Dabei werden die Methoden GET, POST, PUT und DELETE zur Verfügung gestellt, sowie Möglichkeiten Header, Options und Body eines Requests zu definieren. (Angular 2017e)

```
searchOfferExt(params:any) {
    return this.http.get(Path + "/api/offer")
        .map((r: Response) => r.json() as Offer[]);
}
```

Code-Ausschnitt 16: GET-Request aus Service Funktion

Als Rückgabewert wird ein sogenannter Observable erwartet. In diesem Beispiel wird innerhalb der Funktion eines Services die Antwort des Servers bereits auf einen bestimmten Datentypen gemaped.

Observable

Ein Observable ist ein Fluss an Daten auf den man subscriben kann, um fortan über mögliche Events informiert zu werden. Klassisch wird auf den Erfolg oder Misserfolg einer Datenübertragung reagiert. Der Aufruf dieser Funktion ist asynchron.

```
this._searchService.searchOfferExt(params).subscribe(  
  data => {  
    this.resultOffers = data;  
  },  
  error => {  
    console.log("Error during search");  
  }  
);
```

Code-Ausschnitt 17: Subscription auf Methode des Services und Observable

Routing

Der Router in Angular erlaubt das Navigieren von einer View zu einer anderen, basierend auf den Aktionen des Nutzers. Der Trigger liegt dabei innerhalb der Komponente der aktuellen View, die den Nutzer auf eine neue Route weiterleitet und dabei optional auch Parameter mitgeben kann. Der Router kann jedoch auch klassisch eine URL auslesen und so die Route erkennen, die der Nutzer nehmen möchte, und eine entsprechende View generieren. (*Angular 2017f*)

Der Router ist nicht Teil der @angular/core Bibliothek, sondern besitzt sein eigenes Package **@angular/router** und kann optional dem root module einer Applikation hinzugefügt werden. Der **Router Service** bietet die entsprechenden Funktionen, um aus einer Komponente heraus zu navigieren.

Aufbau

Eine SPA besteht nur aus einer einzigen Seite, physisch ist dies die **index.html**, die auf dem Server liegt. Über einen base-Tag wird diese als Ausgangspunkt für den Router deklariert.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>SPA-Example</title>
  <base href="/">
</head>
<body>
  <app-root>Loading...</app-root>
</body>
</html>
```

Code-Ausschnitt 18: Index-Seite der SPA

Im app-root-Tag wird schließlich die komplette SPA geladen und die oberste View aufgebaut, dieses ist nach gegebenen Konventionen die App-Komponente "app.component", die dem root module zum bootapen übergeben wurde.

Routing Module

Die Struktur der Applikation wird dem Router über ein separat geschriebenes Routing-Modul mitgeteilt (siehe Code-Ausschnitt 19: Ausgelagertes Routing Modul). Dieses wird anschließend in das root modul eingefügt, optional kann dieser Teil auch direkt in das root module geschrieben werden.

```

const routes: Routes = [
{
  path: '',
  redirectTo: '/main/start',
  pathMatch: 'full'
},
{ path: 'login', component: LoginComponent },
{
  path: 'main',
  component: MainComponent,
  canActivate: [LoginService],
  children: [
    { path: 'start',      component: StartpageComponent },
    { path: 'search/:input', component: SearchComponent },
    {
      path: '',
      redirectTo: '/main/start',
      pathMatch: 'full'
    },
  ]
},
];
];

@NgModule({
  imports: [ RouterModule.forRoot(routes) ],
  exports: [ RouterModule ]
})
export class RoutingModule { }

```

Code-Ausschnitt 19: Ausgelagertes Routing Modul

- Jede **Route** stellt dabei einen Endpunkt dar, zu dem navigiert werden kann und zu dem eine entsprechende URL innerhalb der SPA existiert. Routen können verschachtelt werden, sodass manche Views nur über andere erreichbar sind.
- Durch **redirectTo** können z.B. bei keiner oder Falscheingabe einer Route, ein allgemeiner Fallback-Startpunkt für den Nutzer festgelegt werden. Dazu ist zusätzlich das **pathMatch** Attribut notwendig.
- Mit Hilfe von **canActivate** können zusätzlich Services eingesponnen werden, die über eine bestimmte Methode zurückgeben, ob der Nutzer zu einer bestimmten Route navigieren darf.
- Über einen Doppelpunkt werden die **Parameter* definiert, die eine Route bzw. die Komponente dahinter erwartet.
- Es ist auch möglich **Wildcards** über Pfadangaben wie `***` zu erteilen oder andere Formen eines RegEx Statements zu verwenden.

Router Outlet

Das Router-Outlet ist ein HTML-Tag, welches sich typischerweise im Template der obersten Komponente wiederfindet. Es kann jedoch an beliebiger Stelle eingebracht werden. In diesem Element werden die Views geladen, zu denen der Nutzer traversieren möchte.

```
<router-outlet></router-outlet>
```

Code-Ausschnitt 20: Router Outlet

Navigation

Um einen Navigationspunkt festzulegen kann entweder die RouteLink-Direktive verwendet werden oder die Funktion des Router Service innerhalb einer Komponente verwendet werden. Dabei können auch entsprechende Parameter gesetzt werden.

```
<a routerLink="/main/search" routerLinkActive="active">Search</a>
```

```
this._routerService.navigate(['/main/search', searchString]);
```

Code-Ausschnitt 21: Navigation mit Router

Status

Der Status des Routers kann über das Attribut **RouterState** aus jeder Komponente heraus abgerufen werden. Nach jeder erfolgreichen Navigation baut der Router dabei einen Baum aus **ActivatedRoute** Objekten auf, der den aktuellen Status des Routers darstellt. Die Activatedroute Objekte bieten dem Nutzer Methoden zum traversieren über den Baum.

Parameter Subscription

Eine Komponente kann auf seine eigene Route subscriben, wodurch bei jeder erfolgreichen Navigation die Parameter auf der Route erneut übergeben werden und diese anschließend verarbeitet werden können.

Die Komponente selber wird nicht zerstört, wenn der Router auf eine andere View navigiert, daher ist eine initiale Subscription, die für den kompletten Lifecycle bestehen bleibt, notwendig.

```
constructor(private route: ActivatedRoute) { }

ngOnInit() {
  this.subscription = this.route.params.subscribe((params: Params) => {
    this.searchString = params['input'];
  });
}
```

Code-Ausschnitt 22: Parameteraufruf

Quellen

1. Marlon Brüntje . Single-Page-Application - individuelle Web-Anwendung für Unternehmen. <http://www.flyacts.com/blog/single-page-application-individuelle-web-anwendung-fuer-unternehmen>, 2015. Aufgerufen: 25.06.2017 14:00
2. Rui Peres. Model View Controller in iOS: A modern approach. <https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach>, 2016. Aufgerufen: 01.07.2017 13:00
3. Bernhard Lahres, Gregor R. Rheinwerk Computing :: Praxisbuch Objektorientierung – 8.2 Die Präsentationsschicht: Model, View, Controller (MVC). http://openbook.rheinwerk-verlag.de/oo/oo_06_moduleundarchitektur_001.htm, 2006. Aufgerufen: 01.07.2017 12:45
4. Martin Fowler. Supervising Controller. <https://martinfowler.com/eaaDev/SupervisingPresenter.html>, 2006a. Aufgerufen: 01.07.2017 13:20
5. Martin Fowler. GUI Architecture. <https://martinfowler.com/eaaDev/uiArchs.html>, 2006b. Aufgerufen: 01.07.2017 13:10
6. Piotr Ziomański. Model-View-Presenter Architecture in Android Applications | Macoscope Blog. <http://macoscope.com/blog/model-view-presenter-architecture-in-android-applications/>, 2016. Aufgerufen: 01.07.2017 13:40
7. Andreas Kühnel. Rheinwerk Computing :: Visual C# 2012 - 28 WPF-Commands. http://openbook.rheinwerk-verlag.de/visual_csharp_2012/1997_28_005.html, 2013. Aufgerufen: 01.07.2017 14:10
8. David Hill. The ViewModel Pattern. <https://blogs.msdn.microsoft.com/dphill/2009/01/31/the-viewmodel-pattern>, 2009. Aufgerufen: 01.07.2017 14:30
9. Sebastian Deutsch. Die Flux Architektur und React. <http://reactjs.de/posts/die-flux-architektur-und-react>, 2015. Aufgerufen: 01.07.2017 16:00
10. Oliver Zeigermann. Flux – Ein Versprechen an UI-Architekten. <http://www.embarc.de/flux-architektur-webanwendungen>, 2015. Aufgerufen: 01.07.2017 17:00
11. Angular. Angular - Architecture Overview. <https://angular.io/guide/architecture>, 2017a. Aufgerufen: 02.07.2017 12:00

12. Angular. Angular - Template Syntax. <https://angular.io/guide/template-syntax>, 2017b.
Aufgerufen: 02.07.2017 15.00
13. Angular. Angular - Lifecycle Hooks. <https://angular.io/guide/lifecycle-hooks>, 2017c.
Aufgerufen: 02.07.2017 16:23
14. Angular. Angular - Pipes. <https://angular.io/guide/pipes>, 2017d. Aufgerufen: 02.07.2017
17:15
15. Angular. Angular - HTTP. <https://angular.io/guide/http>, 2017e. Aufgerufen: 02.07.2017
18:00
16. Angular. Angular - Router. <https://angular.io/guide/router>, 2017f. Aufgerufen: 02.07.2017
20:00

Hybride_AppEntwicklung

Autor: Benjamin Schmidt

1. Einführung

Wenn man Apps entwickeln will, dann kann man auf einige Technologien zurückgreifen. Diese Technologien helfen dem Entwickler bei der Entwicklung von Apps. Als erstes sollte man klären, welche App-Variante man entwickeln will. Anhand dieser App-Variante wählt man dann die dafür passende Technologie aus. Ich werde 3 Arten von Apps vorstellen, welche Vorteile und Nachteile diese untereinander haben. Ich werde mich danach dann nur noch mit den hybriden Apps beschäftigen und dafür die zu verwendende Technologie vorstellen.

2. Unterschiede zwischen Web-Apps, Native Apps und Hybride Apps

2.1 Web-Apps

Web-Apps werden mit JavaScript, HTML und CSS entwickelt. Web-Apps laufen im Browser, da Web-Apps Mobile/Responsive Seiten sind. Sie passen sich jeder Größe genau an und sind plattformunabhängig, da jedes Smartphone einen eigenen Browser mit sich bringt. Sie können aber nicht im App Store veröffentlicht werden und unterstützen keine nativen Gerätefunktionen wie z.B. die Benutzung der Kamera oder das Auslesen der Kontakte. [1]

2.2 Native Apps

Native Apps werden mit plattformabhängigen Programmiersprachen wie Java für Android, Objective-C für IOS usw. entwickelt. Dadurch können die Apps im App Store der jeweiligen Plattform angeboten und heruntergeladen werden. Native Apps unterstützen die nativen Gerätefunktionen z.B. die Benutzung der Kamera im Vergleich zu Web-Apps. Sie sind aber nicht plattformunabhängig wie Web-Apps, da Sie für jede Plattform (Android, IOS usw.) neu entwickelt werden. Denn jede Plattform hat ihre eigene Programmiersprache. Die Wartung ist hoch, denn jede einzelne App muss getestet und gewartet werden und Updates müssen einzeln angepasst werden. Es gibt viele verschiedene Geräte z.B. Smartphones und Tablets für die auch einzelnen Apps realisiert werden. [2]

2.3 Hybride Apps

Hybride Apps kombinieren die Vorteile beider vorherigen App Varianten in einer. Hybride Apps werden genau wie Web-Apps mit JavaScript, HTML und CSS entwickelt. Sie können aber im App Store veröffentlicht und heruntergeladen werden wie Native Apps. Sie laufen in einem spezifischen Browser der jeweiligen Plattform, unterstützen aber auch native Gerätefunktionen der gewählten Plattform. Sie haben nicht so eine gute Performanz wie Native Apps, bei Spielen kann es daher zu Leistungseinbrüchen kommen, da empfiehlt sich lieber native Apps zu entwickeln. Ein großer Vorteil von hybriden Apps ist, wenn sie einmal entwickelt und getestet wurden, können Sie für jede Plattform erstellt werden. Man spart

sich die Zeit mehrere Apps für jedes System einzeln zu entwickeln. Dadurch ist die App leichter zu warten und auch Updates können besser eingespielt werden, da sie auf einer App basieren. [3]

Die folgende Tabelle zeigt die Unterschiede für die einzelnen App-Varianten im Vergleich zu den anderen.

App-Varianten	Web-Apps	Native Apps	Hybride Apps
Performanz der App	Sehr gut	Sehr gut	Beim Start dauert es, da die App erst geladen werden muss.
Plattformabhängigkeit	Nein	Ja	Ja
App-Store Veröffentlichung	Kann nicht im App-Store veröffentlicht werden.	Kann im App-Store veröffentlicht werden.	Kann im App-Store veröffentlicht werden.
Unterstützt Gerätefunktionen	Es werden nur einige Gerätefunktionen mit der Nutzung von Web-APIs unterstützt.	Es werden alle Gerätefunktionen unterstützt.	Es werden alle Gerätefunktionen unterstützt, durch eigene und externe Plug-Ins.

In der Tabelle sieht man in der Spalte für hybride Apps, dass es nur Leistungseinbrüche bei der Performanz der App gibt. Das merkt man aber auch beim direkt Start einer hybriden App sehr genau.

Hier sieht man eine Grafik, welche die internen Abläufe zeigen. Man sieht, dass eine Native App und eine Web-App sich darin unterscheiden, dass die Web-App in einem Browser ausgeführt wird. Der Browser wird vom Betriebssystem aufgerufen. Das ist auch der Grund, dass man keine Gerätefunktionen bei Web-Apps verwenden kann, da der Browser vom System abgekapselt ist. Wäre das nicht der Fall, dann könnte man mit einer einfachen Webseite auf alle Informationen eines PCs zugreifen und manipulieren. Das hat Sicherheitsgründe. Für einen Angreifer wäre das ein leichtes Spiel sich alle Kontakte und sonstige Informationen von einem Smartphone zu beschaffen. Bei nativen Apps ist das möglich Gerätefunktionen zu nutzen, da native Apps auf dem Betriebssystem installiert werden. Sie greifen direkt auf die System internen Prozesse und Daten zu. Eine hybride App muss beides kombinieren, dazu wird ein Browser simuliert, welches auf dem Betriebssystem installiert wird. Dieser spezifische Browser besteht aus einer Web View zum Anzeigen des HTML-Codes und einer nativen Schnittstelle. Die native Schnittstelle erlaubt native Zugriffe auf die Gerätefunktionen und stellt die native UI zur Verfügung, damit die hybride App wie eine native App aussieht.

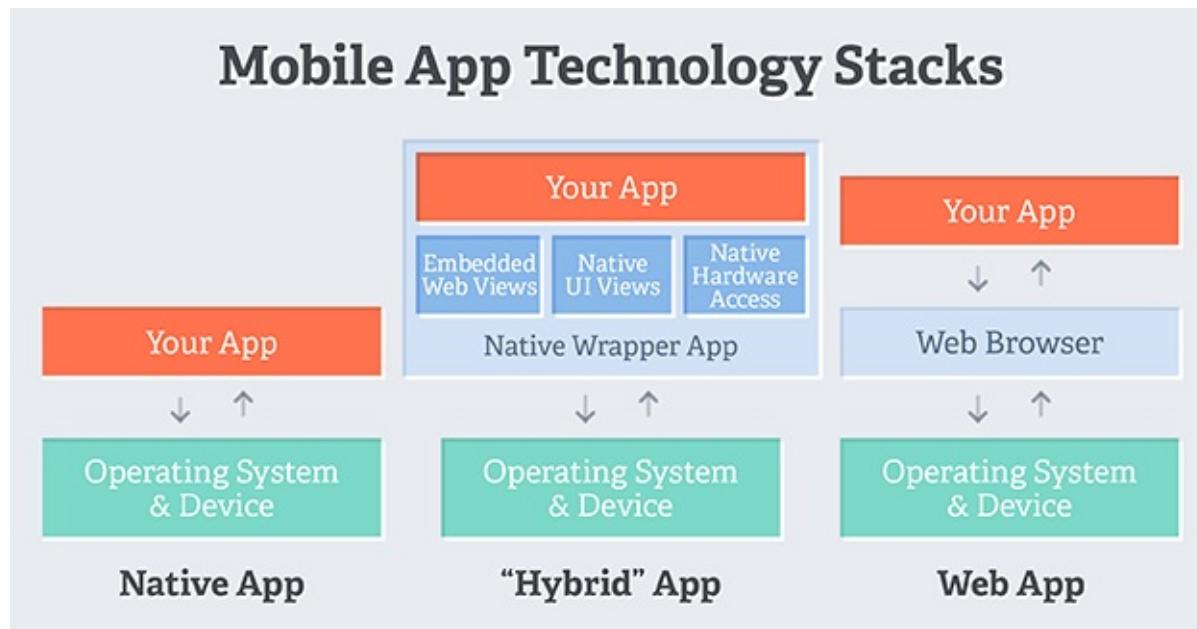


Abbildung 1: Der interne Ablauf einer App. [8]

3. PhoneGap/Cordova

3.1 Allgemein

PhoneGap/Cordova ist ein Framework für die Entwicklung von hybriden Applikationen, welches auf einem Smartphone/Mobile Device benutzt werden kann.

PhoneGap/Cordova wurde von der Firma Nitobi Software entwickelt. Diese Firma wurde am 04. Oktober 2011 von Adobe Systems aufgekauft. Seit dem gehört PhoneGap/Cordova Adobe Systems und Adobe Systems entwickeln PhoneGap/Cordova auch weiter.

PhoneGap baut auf Apache Cordova auf. Ursprünglich war Apache Cordova PhoneGap wurde aber aus markenrechtlichen Gründen umbenannt in Apache Callback und später dann in Apache Cordova. PhoneGap/Cordova unterstützt folgende Betriebssysteme: IOS, Android, WebOS, Symbian OS, Blackberry, Windows Phone, Windows 8. Die Hybriden Apps die man mit PhoneGap/Cordova entwickelt benutzen JavaScript, HTML und CSS als Entwicklungssprachen. Dabei wird die Programmlogik mit JavaScript realisiert, das Layout mit HTML und das Design mit CSS. PhoneGap/Cordova ist lizenziert unter der Apache Lizenz 2.0. [4]

3.2 Architektur

PhoneGap/Cordova besteht aus folgenden zwei Teilen: der Cordova.js und Nativen Engine. Die Cordova.js stellt eine API für plattformabhängige Zugriffe auf die nativen Gerätefunktionen des jeweiligen Systems bereit. Die Native Engine verarbeitet die nativen Befehle und ermöglicht die Abfrage von Sensoren. Für jedes System existiert eine eigene Native Engine, damit der native Code für jedes System separat verarbeitet werden kann. Es existieren viele verschiedene Plug-Ins, sie stellen native Gerätefunktionen für die verschiedenen Plattformen bereit z.B. Kamerasteuerung. Es existieren externe Plug-Ins für Gerätefunktionen, die noch nicht vollständig unterstützt werden. Diese Plug-Ins wurden von der Community entwickelt und werden auch von ihnen weiterentwickelt. Für jedes Plug-In muss eine JavaScript Datei existieren und eine Klasse für den nativen Code für jede unterstützte Plattform, damit die Gerätefunktion native genutzt werden kann, da das Plug-In über JavaScript angesprochen wird und mit den nativen Plug-In der passenden Plattform kommuniziert. Die HTML, CSS und JS Dateien werden in die WebView geladen. Und die WebView ladet die PhoneGap/Cordova Plug-Ins, danach wird der native Code auf dem jeweiligen System ausgeführt. [5] [9]

In Abbildung 2 kann man die Architektur von PhoneGap/Cordova sehen. In der Abbildung wird die Schnittstelle zwischen der App und dem mobilen Betriebssystem gezeigt. Die Plug-Ins kommunizieren mit dem mobilen Betriebssystem über eine eigene API und auch die WebView kommuniziert mit der API vom mobilen Betriebssystem. Die Plug-Ins kommunizieren mit der WebView, da die Plug-Ins in Javascript geschrieben sind. Der Entwickler benutzt die JavaScript-Funktionen, um die Sensoren und Service vom mobilen Betriebssystem zu verwenden.

Der HTML5-Code und das Cordova.js werden in der WebView geladen. Danach werden die PhoneGap/Cordova Plug-Ins für die passende Native Engine geladen. Aus der WebView heraus wird der Browser geladen und die App wird dort angezeigt. Der Browser interpretiert den HTML5-Code und zeigt ihn an. PhoneGap/Cordova unterdrückt die Bedienelemente vom Browser, sodass der Nutzer nicht merkt, dass die App im Browser läuft. Der Browser fängt den PhoneGap/Cordova -Aufruf der eigenen JavaScript-Datei ab und ruft den passenden nativen Aufruf der jeweiligen Plattform auf über die Plug-Ins, die dann mit dem System des Mobile Device interagieren. Das Ergebnis kommt per JavaScript Callback an die Anwendung zurück und kann dann dargestellt oder verarbeitet werden mit dem Browser. [5] [9]

PhoneGap Architecture

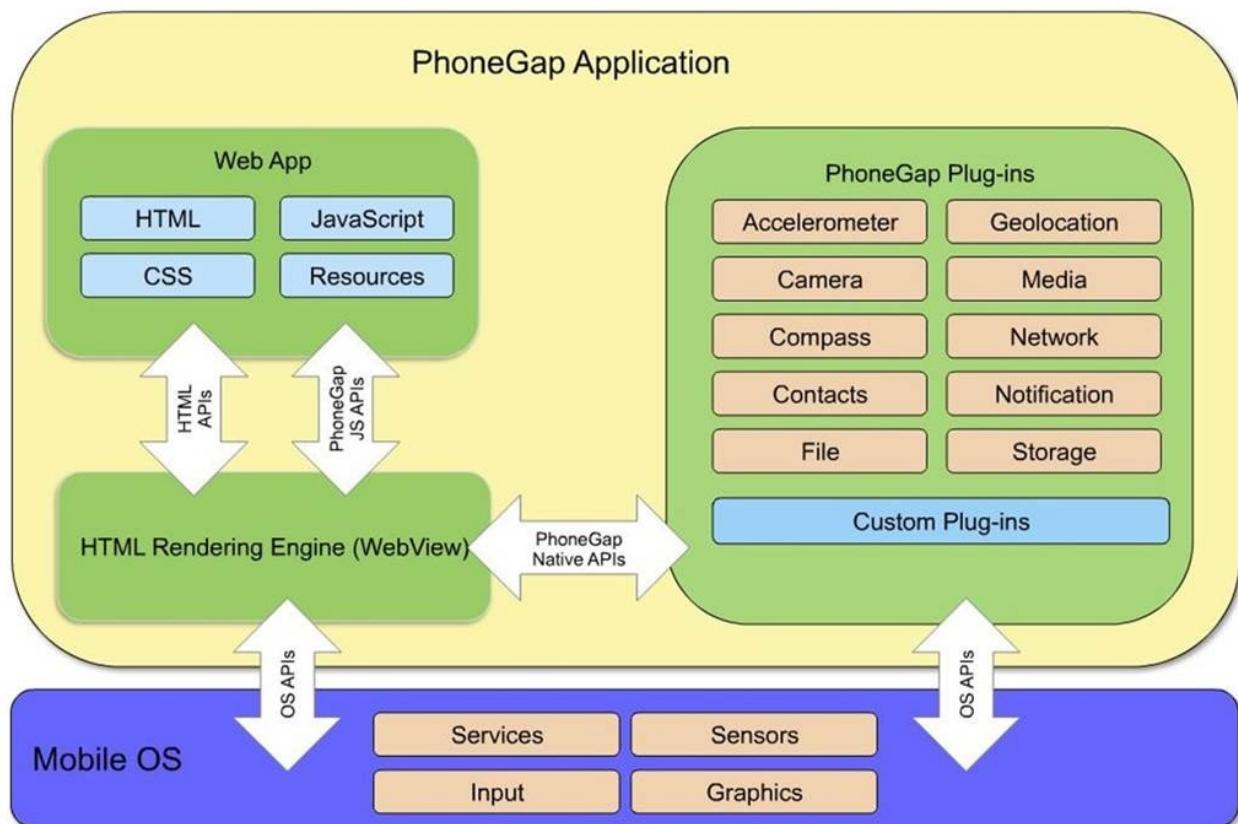


Abbildung 2: Allgemeine Architektur von PhoneGap/Cordova. [9]

4. Ionic

4.1 Allgemein

Ionic ist ein Frontend-Framework für das Erstellen von hybriden Apps. Ionic legt einen besonderen Wert auf eine Native UI, sodass die UI-Elemente ein natives Aussehen haben. Für die Architektur von Ionic wird Angular 2 verwendet. Die hybride App wird mit PhoneGap/Cordova aufgerufen, dadurch können die nativen Gerätefunktionen benutzt werden, um zum Beispiel die Kamera zu benutzen. [6] [7]

4.2 Komponenten

Für die Erstellung der hybriden Apps benutzt Ionic verschiedene Frameworks als Komponenten.

4.2.1 SASS

SASS wird zur Erzeugung von Cascading Style Sheets genutzt, da SASS eine Stylesheet-Sprache ist. SASS generiert CSS mit einem Präprozessor. Der Vorteil von SASS gegenüber CSS ist, dass der Entwickler auf Vererbung, Variablen usw. verwenden kann, wie bei einer Programmiersprache. Wenn man SASS kompiliert, dann erhält der Entwickler am Ende den CSS Code. Ionic verwendet SASS, um das Design der App festzulegen. Das Design kann über SASS Variablen verändert werden und der Entwickler kann eigene hinzufügen. Es existiert für Android, IOs und andere eigene Variablen, um für jedes Betriebssystem ein individuelles Design festzulegen. [7]

4.2.2 Apache Cordova/PhoneGap

Ionic benutzt zum Anzeigen und installieren der App das Framework PhoneGap/Cordova. Das Framework PhoneGap/Cordova stellt eine API bereit. Mit dieser API kann man auf die Gerätefunktionen zugreifen. PhoneGap/Cordova wird weiter oben genauer beschrieben. [7]

4.2.3 AngularJS und Angular 2

AngularJS und Angular 2 sind ein JavaScript-Framework. AngularJs wird in Ionic 1 verwendet und ab Ionic 2 wird nur noch Angular 2 verwendet. Die Programmlogik wird in Ionic mit Angular umgesetzt, da Ionic auf Angular basiert. Durch die Verwendung von

Angular hat Ionic eine MVC-Architektur. Mit Angular lassen sich Single-Page-Applikationen entwickeln. Angular benutzt für die Funktionalität Direktiven und diese können in der View verwendet werden. [7]

4.2.4 Node.js

Node.js ermöglicht es JavaScript serverseitig zu verwenden und Node.js stellt einen eigenen Anwendungsserver. Andere Anwendungen können auf diesen Server zugreifen. So kann Node.js als Schnittstelle zwischen der Datenbank und einer Single-Page-Applikation genutzt werden. Ein Vorteil von Node.js ist, dass seine Architektur ressourcensparend ist und man sie ständig mit neuen Modulen erweitern kann. Damit man Ionic benutzen kann muss man Node.js installieren, da Ionic über den Browser auf Node.js zugreift. [7]

In der folgenden Abbildung 3 werden die einzelnen Komponenten und deren Beziehungen untereinander dargestellt. Man sieht, dass für die Hardware also die Gerätefunktionen PhoneGap/Cordova verwendet wird. PhoneGap/Cordova stellt die Schnittstelle zu der Hardware her. Für die Architektur wird Angular 2 bzw. AngularJS verwendet. Angular stellt auch die Schnittstelle zwischen der Ionic-UI Komponenten und PhoneGap/Cordova da. Die Ionic-UI Komponenten werden mit SASS gestaltet und verwenden HTML5 und CSS3. Über Service und Direktiven kann auf die einzelnen Elemente zugegriffen werden.

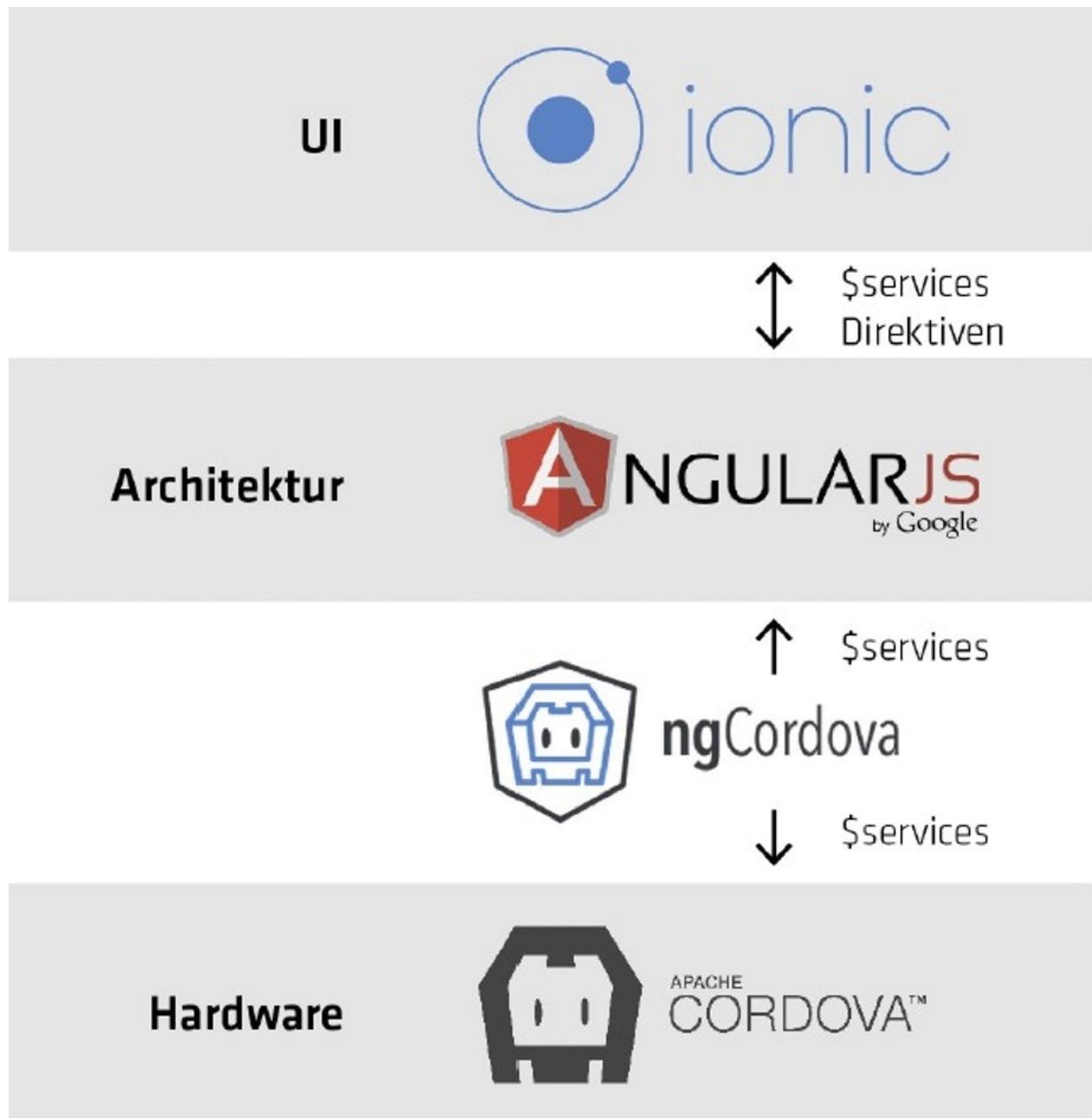


Abbildung 3: Allgemeine Architektur von Ionic. [10]

5. Fazit

Wenn man sich in die hybride App Entwicklung eingearbeitet hat, dann kann man sehr schnell sehr gute Ergebnisse erzielen. Es lassen sich damit leicht einfache Apps entwickeln, welche nur eine Kommunikation mit einer Datenbank herstellen wollen oder nur Inhalte anzeigen wollen. Wenn aber ein Entwickler eine komplexere Anwendung entwickeln will, dann sollte der Entwickler immer native entwickeln, da bei der hybriden App-Entwicklung immer Plug-Ins verwendet werden. Diese müssen vom Entwickler selber geschrieben werden, wenn kein bereits fertiges Plug-In existiert. Das bringt den Entwickler aber nichts, da er dann Plug-Ins für Android und IOs mit Nativen Code entwickeln müsste. Der Entwickler müsste auch noch den Code ständig weiterentwickeln, da kann der Entwickler gleich Native entwickeln. Ein Entwickler müsste sich mit der Funktionsweise von PhoneGap/Cordova auskennen. Für kleine Anwendung ist die hybride App Entwicklung sehr gut geeignet. Für große Entwicklungen sollte man die App Native umsetzen, da man einzelne Änderungen im Design und der Funktionslogik der App besser anpassen kann. Ein Entwickler sollte auch darauf achten das hybride Apps in der Performanz Probleme haben.

Mit PhoneGap/Cordova lassen sich gut hybride Apps umsetzen. Es gibt auch sehr viele Plug-Ins für PhoneGAp/Cordova, diese werden auch ständig verbessert und erweitert. Wenn ein Entwickler besonderen Wert auf die UI legt, dann ist Ionic eine gute Wahl, da dieses Framework seinen Schwerpunkt auf native UI setzt. Man hat bei Ionic auch eine MVC-Architektur zur Verfügung und viele lästige JavaScript Aufgaben werden einen Entwickler abgenommen, da Angular 2 viele nützliche Module bereits mitbringt.

Quellen

Literaturquellen

[1] Web-Apps

<http://de.wikipedia.org/wiki/Web-App> nachgeschaut am 01.07.2017

[2] Native Apps

http://de.wikipedia.org/wiki/Mobile_App#Native_Mobile_Apps nachgeschaut am 01.07.2017

[3] Hybride Apps

<http://de.wikipedia.org/wiki/Hybrid-App> nachgeschaut am 01.07.2017

[4] Allgemeine Daten über PhoneGap

<http://de.wikipedia.org/wiki/PhoneGap> nachgeschaut am 01.07.2017

[5] Marcus Ross (13.08.2013)

<http://www.heise.de/developer/artikel/Cross-Plattform-Apps-mit-PhoneGap-entwickeln-1934535.html> nachgeschaut am 01.07.2017

[6] Ionic Framework

<http://ionicframework.com/> nachgeschaut am 01.07.2017

[7] Entwicklung einer Ionic App

<http://t3n.de/magazin/entwicklung-ionic- 237246/> nachgeschaut am 01.07.2017

Bilderquellen

[8] Interner Ablauf einer App

<http://developer.telerik.com/wp-content/uploads/2015/11/hybrid-native-web.png>
nachgeschaut am 01.07.2017

[9] Architektur von PhoneGap/Cordova

<https://html5hu.files.wordpress.com/2011/10/phonegap-architecture-by-ibm-29-july-2011-modules-instead-plug-ins.jpg> nachgeschaut am 01.07.2017

[10] Architektur von Ionic

<http://img.t3n.sc/magazin/wp-content/uploads/2014/11/ionic-app-entwicklung1.jpg?auto=compress%2Cformat&fm=jpg&ixlib=php-1.1.0&q=65&w=920&s=71ff64b0587f19a3c06c263339799839> nachgeschaut am 01.07.2017

Realtime Datenbanken

Autor: Tolga Aydemir

Einleitung

Realtime Datenbanken sind mehr als nur Datenspeicher mit einem Datenbank Management System (wie zum Beispiel mongoDB or MySQL). Realtime Datenbanken haben nicht nur den Aspekt der Verarbeitung der Ereignisse (insert, update, delete) in Echtzeit inne, sondern auch die Kommunikation (Verbreitung) eben dieser Ereignisse in Echtzeit zu allen angemeldeten Client-Anwendungen. Das heißt, Realtime Datenbanken sind eine Erweiterung herkömmlicher Datenbanken um die Echtzeit-Verbreitung der eintretenden Ereignisse. Alle (herkömmlichen) Datenbanken können die Ereignisse *sofort* verarbeiten und eine darauffolgende Abfrage würde den neuen Stand der Datenbank wiedergeben. Allerdings ist es für einige Szenarien, wie Börsenkurse oder Chats, durchaus von Bedeutung, dass jede Aktualisierung allen Teilnehmern in Echtzeit zur Verfügung gestellt wird, ohne, dass eine Client-Anwendung erst eine Anfrage an die Datenbank starten muss. Weitere Optimierungen der Datenbank hinsichtlich dem Verzicht auf unnötige Schemata, Regeln, Verarbeitung der Daten oder die Nutzung von JSON-Dokumenten können die Verarbeitungsgeschwindigkeit der Datenbank bzw. auch der Client-Anwendung weiter steigern - aber das trifft auch auf Datenbanken ohne die Echtzeit-Verbreitung der Ereignisse zu.

Technische Aspekte

Die Echtzeit-Verbreitung der Ereignisse bzw. der aktualisierten Daten findet im Idealfall durch [WebSockets](#) statt. Als Ersatz bei älteren Client-Anwendungen kann auch das sog. *long polling* ausreichen.

Echtzeit in Zusammenhang mit der Kommunikation über das Internet bedeutet: "so schnell wie technisch möglich". Latenzen und Paketverluste können während der Kommunikation auftreten. Bei sehr zeitkritischen Anwendungen kann es notwendig sein, Zeitstempel der Datenbank mit den Nachrichten zu schicken, damit die Client-Anwendungen die Nachrichten besser einordnen und darauf reagieren können.

Um eine Datenbank zu einer Realtime Datenbank zu erweitern bedarf, es mindestens zwei weiterer Komponenten:

- Backend-Server mit Unterstützung für WebSockets
- Publish/Subscribe Server

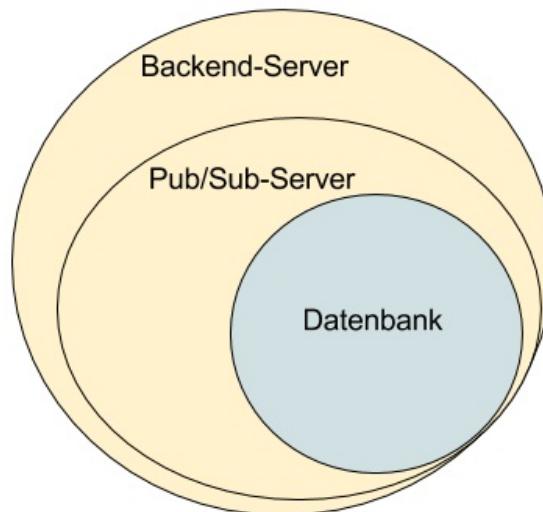


Abbildung: Veranschaulichung Erweiterung einer beliebigen Datenbank um eine Echtzeit-Komponente

Ereignisse in der Datenbank werden von dem Publish/Subscribe Server erkannt und über die WebSockets propagiert. Ein Publish/Subscribe Server, ist ein Server, der zwischen dem Backend-Server und der Datenbank sitzt und alle Veränderungen der Datenbank entgegennimmt und zum Abonnieren über ein WebSocket zur Verfügung stellt. Als Beispiel für einen Publish/Subscribe Server sei hier Redis genannt. Aber anstatt eine beliebige SQL- oder noSQL-Datenbank um diese durchaus komplexen Komponenten zu erweitern und so zu konfigurieren, dass diese auch gut skalieren, können vorgefertigte Dienste genutzt werden, die nicht nur diese Komponenten zur Verfügung stellen, sondern auch Bibliotheken

für diverse Web-Technologien, wodurch die Implementierung der Echtzeit-Komponente für die eigene Anwendung deutlich erleichtert wird.

Ein durchaus bekannter Dienst, der diese Komponenten bündelt und als *eine* Realtime Datenbank vermarktet, ist *Firebase Database*.

Vor- und Nachteile

Die Nachteile von Realtime Datenbanken gegenüber herkömmlichen Datenbanken hängen von der Implementierung und dem Optimierungsgrad der Datenbank ab. Wenn das Szenario sehr zeitkritisch ist, dann wird die Implementierung und Optimierung der Datenbank darauf hinauslaufen, dass die Vorteile (Relationen, Querys, Schemata) von klassischen RDBMS nicht zum tragen kommen werden, da die Vorteile durch Rechenkraft erkauft werden und dies kostet Zeit. Aber der Aspekt der Echtzeit-Verbreitung der Ereignisse selbst muss *keinen* Nachteil mitbringen, wenn das Szenario nicht die Definition "so schnell wie technisch möglich" hat. Dann kann, wie in Kapitel Technische Aspekte erläutert, jede Datenbank zu einer Realtime Datenbank erweitert werden.

Ein Vorteil von Realtime Datenbanken ist es, vereinfachte Schnittstellen zu haben, um an Datensätze zu gelangen, die sich stetig verändern bzw. erweitern. Anstatt, dass eine Client-Anwendung immer wieder mit einer Anfrage an eine REST-Schnittstelle arbeitet, um nach neuen Daten zu fragen und diese darzustellen, kann es in modernen Rich Client Applikationen ausreichen, wenn diese sich einmalig an einem WebSocket registrieren und die Daten ohne weitere Verarbeitung anzeigen.

Dies kann dann so ausgebaut werden, dass ein `three-way-data-binding` vorliegt. In modernen Rich Client Applikationen wird häufig das `two-way-data-binding` genutzt, um den Status der Client-Anwendung im Model (MVC-Pattern) aus der View heraus zu verändern. Mit dem `three-way-data-binding` ergibt sich nun die Möglichkeit, dass die Realtime Datenbank eine Aktualisierung bekommt (von einer *beliebigen* Quelle aus; sei es eine andere Client-Anwendung oder einer externen Schnittstelle), dieses Ereigniss wird dann zu allen Client-Anwendungen propagiert und dadurch wird der Status der jeweiligen Client-Anwendung im Model verändert und diese Veränderung spiegelt sich dann auch in der Veränderung der View wieder. Durch die vereinfachte Schnittstelle der Realtime Datenbank (zum Beispiel Firebase) kann das in wenigen Zeilen Code ermöglicht werden.

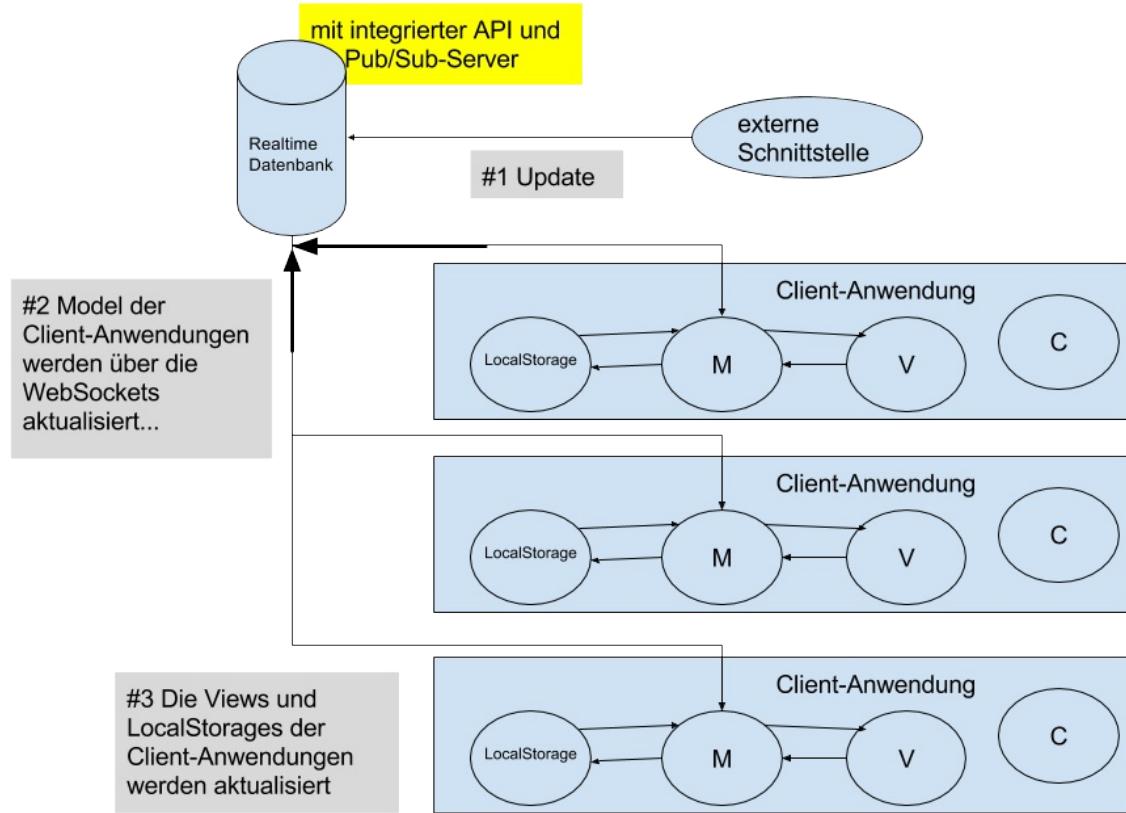


Abbildung: Szenario Four-Way-Data-Binding mit mehreren Client-Anwendungen

In der Abbildung wird das three-way-data-binding zum four-way-data-binding erweitert. Die vierte Komponente ist in dem Fall das *LocalStorage*, das dazu genutzt wird, um die Daten auch offline speichern und erreichen zu können. Wenn das Szenario die Nutzung des LocalStorages eines WebBrowsers nicht zulässt, kann auch eine andere lokale Datenbank genutzt werden. Diese lokale (offline) Datenbank muss unbedingt synchron zur Realtime Datenbank sein. Jede Veränderung der Realtime Datenbank muss synchronisiert werden, sodass die Abfragen der Client-Anwendung sich direkt an die lokale Datenbank wenden und nicht an die Realtime Datenbank.

Einsatzmöglichkeiten

Die Einsatzmöglichkeiten einer Realtime Datenbank ergeben sich immer aus der Anforderung der Echtzeit-Kommunikation. Ob die Kommunikation zwischen Menschen in einer Chat-Applikation abläuft oder ob voll automatisierte Maschinen ihre Positionen oder Arbeitsschritte miteinander synchronisieren, ist dabei irrelevant. Es wird verschiedene Implementierungen und Optimierungsstufen der Datenbanken geben, um sich dem Szenario anzupassen, aber die Idee eine Nachrichtenquelle zu abonnieren, um jegliche Aktualisierungen in Echtzeit zu bekommen, ist dabei die Gemeinsamkeit der Szenarien. Um explizit einige Beispiele zu nennen, die von einer Realtime Datenbank profitieren (können): Chat-Applikationen, Flottensysteme (LKW, Taxi, Schiffe, Bahn), Wetterdaten und Maschinen/Roboter, die miteinander Daten teilen müssen.

Quellen

- [1] <https://firebase.google.com/products/database/>
- [2] https://en.wikipedia.org/wiki/Real-time_database
- [3] <https://medium.com/@leetheguy/firebase-pros-and-cons-ce37c766190a>

Kommunikation in verteilten Systemen

Autor: Timo Rolfsmeier (Einführung, RPC, REST, GraphQL und Fazit) bzw. Tolga Aydemir (WebSocket)

Einführung

Ein verteiltes System ist als ein Verbund von Rechenknoten definiert, der zur Erreichung eines übergeordneten Ziels dienen soll. Die Instanzen innerhalb des Verbunds sind jeweils für einzelne Teifunktionen verantwortlich und kommunizieren zwecks Informationsaustausch über ein Netzwerk. Die Gründe, warum verteilte Systeme errichtet werden, sind vielfältig. Zum einen ist die Flexibilität zu nennen. Ein verteiltes System kann dynamisch um neue Rechenknoten erweitert bzw. reduziert werden, ohne dass die bestehenden Instanzen hiervon betroffen sind. Weiterhin erlauben verteilte Systeme die Trennung von Rechenknoten über Orts- und Organisationsgrenzen hinweg. Es kann somit dynamisch auf die Informationen von entfernten Hard- und Softwarekomponenten zugegriffen werden, ohne diese auf dem eigenen System redundant zu implementieren bzw. tiefere Kenntnisse von ihnen zu haben. Auch die Performanz spielt für diese Architekturform eine zentrale Rolle. Ein Rechenknoten kann beispielsweise mehrfach instanziert werden, um Aufgaben zu parallelisieren und somit für eine bessere Lastverteilung zu sorgen. Schließlich reduziert sich auch das Risiko für einen Dienstausfall bzw. Systemfehler, wenn Rechenknoten redundant und dezentralisiert vorliegen. (vgl. Schill und Springer 2012, S. 3-6)

Im Gegensatz zu einer monolithischen Architektur ist ein verteiltes System auf eine interne Kommunikation angewiesen. Da Rechenknoten sowie deren Speicher physikalisch voneinander getrennt sind, müssen Informationen innerhalb von Nachrichten übermittelt werden. Um nicht zu einem Flaschenhals für das Gesamtsystem zu werden, ist der Datenaustausch möglichst performant, sicher und flexibel zu gestalten. Innerhalb dieses Kapitels werden verschiedene Methoden und Ansätze vorgestellt, welche eine Lösung bezüglich dieser Problematik bieten. Konkret wird dabei auf die Technik RPC (Remote Procedure Call), das Netzwerkprotokoll WebSocket, den Architekturstil REST (Representational State Transfer) und die Abfragesprache GraphQL eingegangen.

RPC

RPC (Remote Procedure Call) ist eine Basistechnik zur Kommunikation innerhalb von heterogenen Anwendungslandschaften. Grundgedanke von RPC ist es, Teile eines Kontrollflusses an externe Rechenknoten mit anderen Adressräumen auszulagern (vgl. Schill 2012, S.46). Das Prinzip wurde 1976 veröffentlicht und befindet sich seit 1988 im RFC-Standard. Im gleichen Jahr erfolgte die erstmalige Implementierung von RPC durch das ONC RPC (Open Network Computing RPC), welches von Sun Microsystems entwickelt wurde.

Ablauf

Der Ablauf eines entfernten Methodenaufrufs gliedert sich in zwei verschiedene Abschnitte. Bevor der Client einen externen Server mit der Abarbeitung seines Kontrollflusses beauftragen kann, ist ein passender Service zu finden. Der Bindevorgang sowie der nachgelagerte Methodenaufruf werden im Folgenden nach (Schill 2012, S.47-51) beschrieben.

Bindevorgang

In komplexen Netzwerken hat ein Client typischerweise keine Kenntnis von der Gesamtheit der erreichbaren Serverknoten. Es liegen ihm dementsprechend keine Informationen darüber vor, welche Services von Servern angeboten werden bzw. mit welcher Syntax und

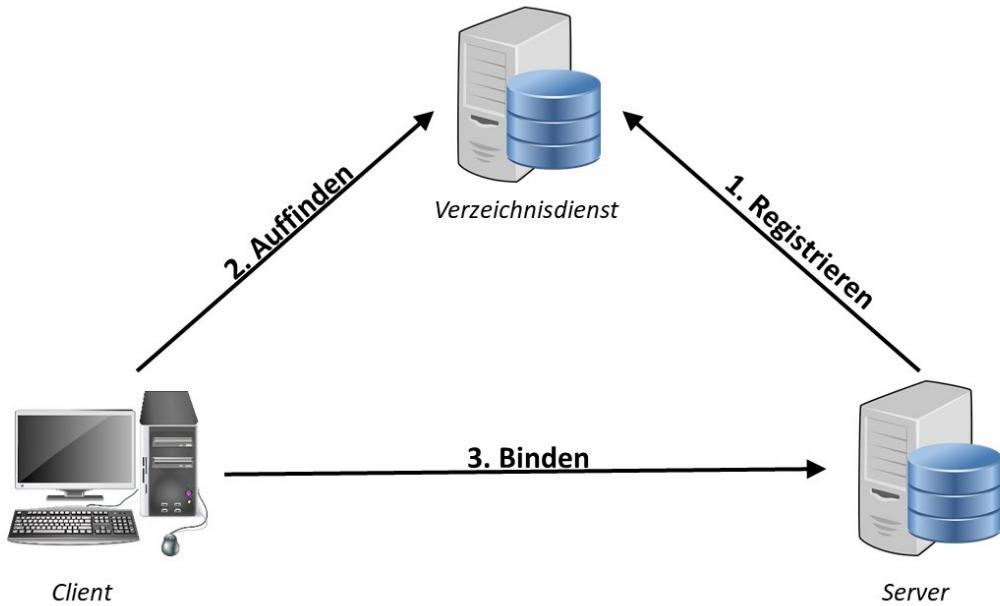


Abbildung 1: Bindevorgang mithilfe von Verzeichnisdienst

Methodenaufruf

Nachdem ein Client Kenntnis von einem Dienst hat, kann er einen entfernten Methodenaufruf auf diesem durchführen. Initial ruft der Client die benötigte Methode auf seinem lokalen System auf (siehe Abbildung 2). Dieser Aufruf wird nun an einen sogenannten **Stub** weitergeleitet, welcher ebenfalls auf dem Client befindlich ist. Ein Stub wird mithilfe der Schnittstellenbeschreibung des Servers erstellt und umfasst alle darin deklarierten Methoden. Das lokale Programm hat keine Kenntnis vom entfernten Server, sondern ausschließlich vom Stub. Dieser kann somit als ein clientseitiger Stellvertreter für die Serverfunktion angesehen werden. Der aufgerufene Stub wandelt nun alle an den Server zu übermittelnden Daten in ein für die Netzwerkübertragung geeignetes Format um, was als **Marshalling** bezeichnet wird. Hierunter befinden sich Angaben über die aufzurufende Methode sowie ggf. Methodenparameter. Anschließend erfolgt eine Übermittlung an den Server. Auf diesem befindet sich nun wiederum ein Stellvertreter, welcher **Skeleton** genannt wird. Dieser repräsentiert das aufrufende Objekt innerhalb des Servers. Nach dem Deserialisieren (**Unmarshalling**) der empfangenden Daten, ruft der Skeleton die definierte Funktion des lokalen Server-Programms auf. Das berechnete Ergebnis wird nun auf umgekehrtem Weg wieder an das lokale Programm des Clients übermittelt, wo die Prozessausführung fortzusetzen ist.

sogenannten **Stub** weitergeleitet, welcher ebenfalls auf dem Client befindlich ist. Ein Stub wird mithilfe der Schnittstellenbeschreibung des Servers erstellt und umfasst alle darin deklarierten Methoden. Das lokale Programm hat keine Kenntnis vom entfernten Server, sondern ausschließlich vom Stub. Dieser kann somit als ein clientseitiger Stellvertreter für die Serverfunktion angesehen werden. Der aufgerufene Stub wandelt nun alle an den Server zu übermittelnden Daten in ein für die Netzwerkübertragung geeignetes Format um, was als **Marshalling** bezeichnet wird. Hierunter befinden sich Angaben über die aufzurufende Methode sowie ggf. Methodenparameter. Anschließend erfolgt eine Übermittlung an den Server. Auf diesem befindet sich nun wiederum ein Stellvertreter, welcher **Skeleton** genannt wird. Dieser repräsentiert das aufrufende Objekt innerhalb des Servers. Nach dem Deserialisieren (**Unmarshalling**) der empfangenden Daten, ruft der Skeleton die definierte Funktion des lokalen Server-Programms auf. Das berechnete Ergebnis wird nun auf umgekehrtem Weg wieder an das lokale Programm des Clients übermittelt, wo die Prozessausführung fortzusetzen ist.

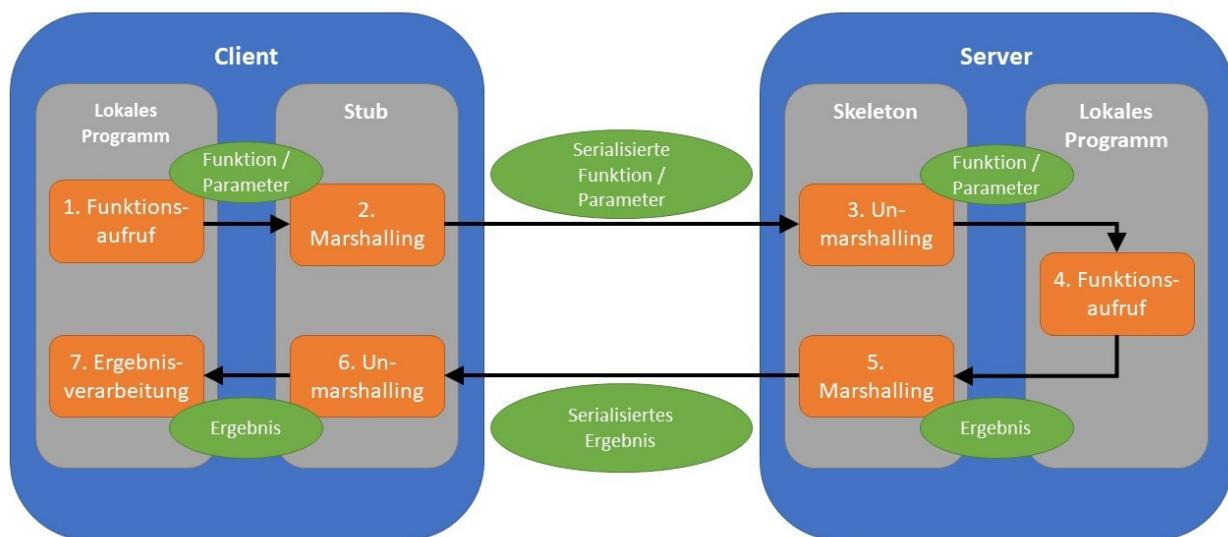


Abbildung 2: Schematische RPC-Kommunikation

Quelle: In Anlehnung an (Schill 2012, S. 47)

Wichtige Punkte:

Wichtige Punkte bezüglich der Verwendung von RPC werden im Folgenden nach (Schill 2012, S. 51-55) dargestellt.

Parameterübergabe: Client und Server werden in den meisten Fällen verschiedene Rechenknoten sein, weshalb sie auch über verschiedene Adressräume verfügen. Dies kann zu Problemen führen, wenn Funktionen per Call-by-Reference aufgerufen werden. Das

Client. Hat der Server seine Berechnung abgeschlossen, wird dieses Objekt auf Clientseite durch das tatsächliche Ergebnis ersetzt. Der Vorteil eines Future bzw. Promise ist, dass für den Client zu jedem Zeitpunkt ersichtlich ist, ob dieses bereits das gewünschte Resultat beinhaltet oder noch als Platzhalter dient. Der Client kann hierdurch seine Programmprozedur nach Aufruf einer entfernten Methode weiterführen und befindet sich nicht in einem blockierenden Zustand.

Fehlersemantik: Eine RPC-Kommunikation besitzt drei verschiedene Schwachpunkte. Sowohl der Client als auch der Server sind nicht vor Ausfällen geschützt. Weiterhin kann es zu Nachrichtenverlusten innerhalb des Netzwerks kommen. Diese Ausfallrisiken sind in Tabelle 1 verschiedenen klassifizierten RPC-Systemen gegenübergestellt. Jedes Paar von Systemklasse und Fehlerart sind zwei Zahlen zugeordnet. Die Ausführungen beschreiben, wie oft die vom Client angeforderte Operation auf dem Server ausgeführt wurde. Das Ergebnis definiert, wie oft der Client ein Resultat vom Server tatsächlich erhalten hat.

Systemklasse / Fehlerart	Fehlerfreier Ablauf	Nachrichtenverluste	Zus. Ausfall Server	Zus. Ausfall Client
Maybe	Ausführung: 1; Ergebnis: 1	Ausführung: 0/1; Ergebnis: 0/1	Ausführung: 0/1; Ergebnis: 0/1	Ausführung: 0/1; Ergebnis: 0/1
At-Least-Once	Ausführung: 1; Ergebnis: 1	Ausführung: >=1; Ergebnis: >=1	Ausführung: >=0; Ergebnis: >= 0	Ausführung: >=0; Ergebnis: 0
At-Most-Once	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1	Ausführung: 0/1; Ergebnis: 0/1	Ausführung: 0/1; Ergebnis: 0
Exactly-Once	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1

Tabelle 1: Verhalten von RPC-Systemklassen bei verschiedenen Fehlerarten

Quelle: In Anlehnung an (Schill 2012, S. 54)

Ein System mit der Fehlerklasse **Maybe** vollzieht keine Fehlerbetrachtung. Eine Anfrage wird einmalig versendet, unabhängig davon, ob der Client ein Resultat erhält oder nicht. Ein System mit der Fehlerklasse **At-Least-Once** reagiert auf ausbleibende Antworten seitens des Servers, indem der Client weitere Anfragen derselben Art stellt, bis schließlich eine Antwort erhalten wurde. Wurden Antworten aber beispielsweise lediglich aufgrund eines überlasteten Netzwerks noch nicht zugestellt, aber dennoch vom Server bearbeitet, kann die

Systemklasse / Fehlerart	Fehlerfreier Ablauf	Nachrichtenverluste	Zus. Ausfall Server	Zus. Ausfall Client
Maybe	Ausführung: 1; Ergebnis: 1	Ausführung: 0/1; Ergebnis: 0/1	Ausführung: 0/1; Ergebnis: 0/1	Ausführung: 0/1; Ergebnis: 0/1
At-Least-Once	Ausführung: 1; Ergebnis: 1	Ausführung: >=1; Ergebnis: >=1	Ausführung: >=0; Ergebnis: >= 0	Ausführung: >=0; Ergebnis: 0
At-Most-Once	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1	Ausführung: 0/1; Ergebnis: 0/1	Ausführung: 0/1; Ergebnis: 0
Exactly-Once	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1	Ausführung: 1; Ergebnis: 1

Tabelle 1: Verhalten von RPC-Systemklassen bei verschiedenen Fehlerarten

Quelle: In Anlehnung an (Schill 2012, S. 54)

Ein System mit der Fehlerklasse **Maybe** vollzieht keine Fehlerbetrachtung. Eine Anfrage wird einmalig versendet, unabhängig davon, ob der Client ein Resultat erhält oder nicht. Ein System mit der Fehlerklasse **At-Least-Once** reagiert auf ausbleibende Antworten seitens des Servers, indem der Client weitere Anfragen derselben Art stellt, bis schließlich eine Antwort erhalten wurde. Wurden Antworten aber beispielsweise lediglich aufgrund eines überlasteten Netzwerks noch nicht zugestellt, aber dennoch vom Server bearbeitet, kann die wiederholte Anfrage für ungewollte Seiteneffekte sorgen. Dieses Problem vermeiden Systeme mit der Fehlerklasse **At-Most-Once**. Aufrufwiederholungen werden hier vom Server explizit erkannt und nicht behandelt. Die stärkste Klasse wird als **Exactly-Once** bezeichnet. Bei einem Server- bzw. Client-Ausfall werden beide Systeme auf den Zustand gebracht, der vor der Anfrage herrschte. Die Anfrage wird anschließend erneut durchgeführt. Somit ist die Konsistenz der Systeme zueinander gewährleistet.

gRPC

gRPC ist ein öffentlich zugängliches RPC-Framework, das 2015 von Google veröffentlicht wurde. Mithilfe von gRPC ist möglich, verschiedensprachige Microservices bzw. Endgeräte miteinander zu verknüpfen sowie Client-Bibliotheken zu erstellen. (Google 2017)

```
// Proto-Datei

service Greeter {
    // Berechne Preis von speziellem Auto
    rpc CalculatePrice (Auto) returns (int32) {}
}

// Datentyp Auto
message Auto {
    required Hersteller hersteller = 1;
    required int32 id = 2;
    optional string modell = 3;
    optional AntriebType type = 4 [default = VORDERRAD];
}

enum AntriebType {
    ALLRAD = 0;
    VORDERRAD = 1;
    HINTERRAD = 2;
}

message Hersteller {
    required string number = 1;
    optional string standort = 2;
}
```

Code-Beispiel 1: Protocol Buffer

Nachdem die Proto-Datei erstellt wurde, ist diese zu kompilieren. Anschließend erhält man einen nicht mehr lesbaren Protocol Buffer.

Verwendung

Mithilfe des Protocol Buffers generiert gRPC nun automatisch Quellcode. Es werden Getter- und Setter-Methoden für die Message Types erstellt sowie Funktionen zum (De-)Serialisieren. Weiterhin liegen nach Kompilierung Stub- sowie Skeleton-Objekte für Client und Server vor. Die Server-Entwickler können sich hierdurch maßgeblich auf die Geschäftslogik konzentrieren, anstatt auf Implementierungsdetails von RPC. Ein weiterer Vorteil von gRPC besteht darin, dass Services und Message Types generell sprachunabhängig sind und sich somit für verschiedene Systeme nicht unterscheiden. Lediglich die Art der Kompilierung gibt vor, in welcher Sprache die Methoden und Objekte erstellt werden. Hierdurch ist es ohne Aufwand möglich, zwischen heterogenen Systemen zu kommunizieren. Bisher liegen Bibliotheken für zehn verschiedene Programmiersprachen vor. Weiterhin ermöglicht gRPC die einfache Anbindung von vorgefertigten Modulen bezüglich Authorisierung, Lastenverteilung und Health Checking.

```
// Proto-Datei

service Greeter {
    // Berechne Preis von speziellem Auto
    rpc CalculatePrice (Auto) returns (int32) {}
}

// Datentyp Auto
message Auto {
    required Hersteller hersteller = 1;
    required int32 id = 2;
    optional string modell = 3;
    optional AntriebType type = 4 [default = VORDERRAD];
}

enum AntriebType {
    ALLRAD = 0;
    VORDERRAD = 1;
    HINTERRAD = 2;
}

message Hersteller {
    required string number = 1;
    optional string standort = 2;
}
```

Code-Beispiel 1: Protocol Buffer

Nachdem die Proto-Datei erstellt wurde, ist diese zu kompilieren. Anschließend erhält man einen nicht mehr lesbaren Protocol Buffer.

Verwendung

Mithilfe des Protocol Buffers generiert gRPC nun automatisch Quellcode. Es werden Getter- und Setter-Methoden für die Message Types erstellt sowie Funktionen zum (De-)Serialisieren. Weiterhin liegen nach Kompilierung Stub- sowie Skeleton-Objekte für Client und Server vor. Die Server-Entwickler können sich hierdurch maßgeblich auf die Geschäftslogik konzentrieren, anstatt auf Implementierungsdetails von RPC. Ein weiterer Vorteil von gRPC besteht darin, dass Services und Message Types generell sprachunabhängig sind und sich somit für verschiedene Systeme nicht unterscheiden. Lediglich die Art der Kompilierung gibt vor, in welcher Sprache die Methoden und Objekte erstellt werden. Hierdurch ist es ohne Aufwand möglich, zwischen heterogenen Systemen zu kommunizieren. Bisher liegen Bibliotheken für zehn verschiedene Programmiersprachen vor. Weiterhin ermöglicht gRPC die einfache Anbindung von vorgefertigten Modulen bezüglich Authorisierung, Lastenverteilung und Health Checking.

WebSocket

Das WebSocket-Protokoll wurde entwickelt, um bidirektionale Verbindungen zwischen einer Client-Anwendung und einer Host-Anwendung in einer möglichst effizienten Art und Weise zu ermöglichen.

Vor der Entwicklung dieses Protokolls musste das sog. *polling* (über HTTP) verwendet werden, um eine Art bidirektionaler Kommunikation zu ermöglichen, wobei die Kommunikation immer von der Client-Anwendung aus gestartet werden muss. Polling kommt aus dem Englischen und bedeutet übersetzt: befragen oder abfragen. Das heißt, die Client-Anwendung befragt immer wieder die Host-Anwendung nach Neuigkeiten und die Host-Anwendung gibt eine Antwort bzw. eine leere Antwort zurück. Mit dem sog. *long polling* gibt es eine etwas effizientere Möglichkeit des einfachen pollings.

Wieder startet die Client-Anwendung eine Anfrage. Die Host-Anwendung antwortet aber nur, wenn es auch eine Neuigkeit gibt. Ansonsten wird die HTTP-Verbindung so lange offen gehalten, bis eine Neuigkeit entsteht. Erst dann sendet die Host-Anwendung über die bereits etablierte HTTP-Verbindung die Nachricht an die Client-Anwendung und die Verbindung ist damit beendet. Die Client-Anwendung wird dann wieder eine neue HTTP-Verbindung zur Host-Anwendung öffnen und damit auf Neuigkeiten warten.

Die Nutzung des pollings bzw. long pollings erlaubt zwar eine Art der bidirektionalen Kommunikation, es ergeben sich jedoch einige Nachteile. In dem RFC 6455 Dokument der IETF zur Standardisierung des WebSocket-Protokolls werden die folgenden Punkte genannt:

- Die Host-Anwendung muss für jede Client-Anwendung mehrere TCP-Verbindungen aufrechterhalten. Dabei entfällt immer mindestens eine Verbindung auf die polling-Verbindung zur Client-Anwendung und für jede weitere Nachricht durch die Client-Anwendung entsteht eine neue Verbindung.
- Es besteht allgemein ein hoher Anteil an ungenutzten Daten beim Informationsaustausch (Overhead durch HTTP-Header), da die Header-Informationen immer hin und her gesendet werden. Insbesondere in einem Szenario, wie einer Chat-Anwendung, kann die Größe des Headers die Größe der zu transportierenden Nachricht übersteigen.
- Die Client-Anwendung muss die ausgehenden Verbindungen mappen, um die Antworten der Host-Anwendung richtig interpretieren zu können.

Das WebSocket-Protokoll kann diese Nachteile durch die Nutzung einer einzigen TCP-Verbindung umgehen und bricht dennoch nicht mit der HTTP-Infrastruktur, sodass auch die WebSocket-Kommunikation über die Ports 80 und 443 oder aber über HTTP-Proxy

Verbindungen erfolgen kann.

Nach einem sog. *handshake* der Teilnehmer (Client- und Host-Anwendung) kann der bidirektionale Nachrichtenaustausch beginnen.

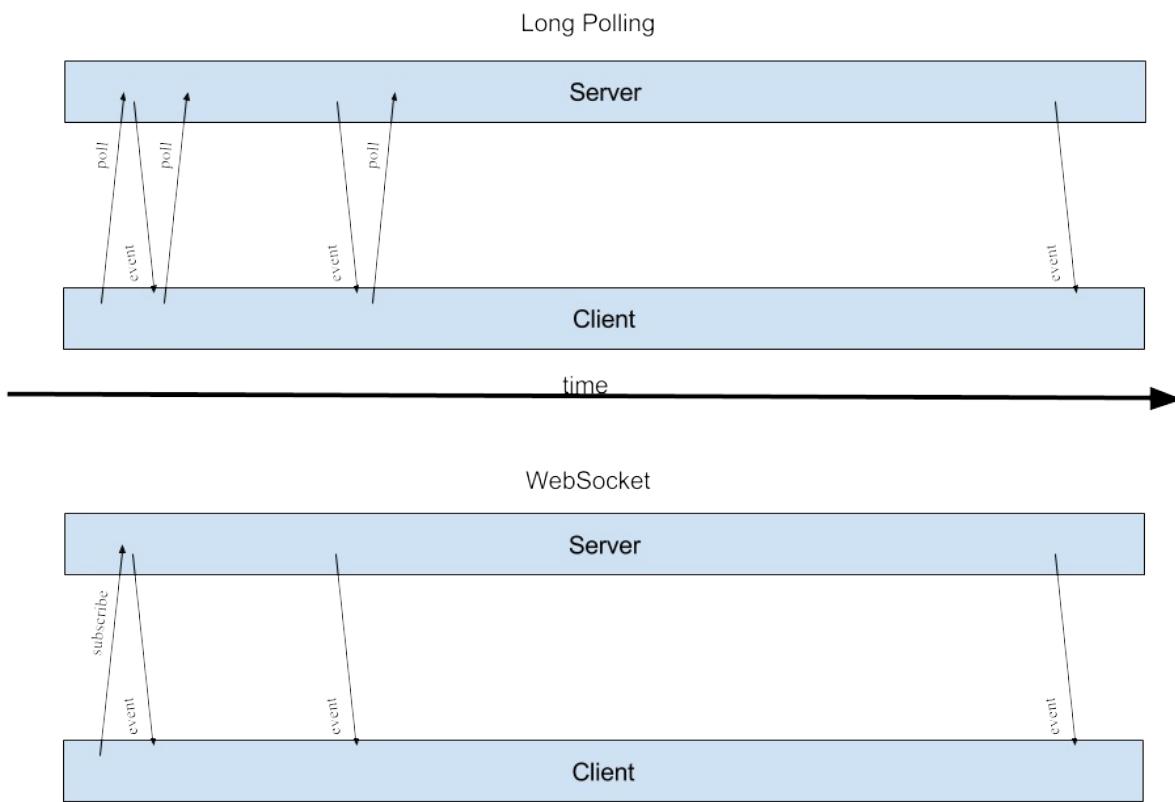


Abbildung A: Veranschaulichung Long Polling und WebSocket Kommunikation

REST

Representational State Transfer (REST) ist ein Architekturstil für verteilte Systeme, der von Roy Fielding innerhalb seiner im Jahr 2000 abgeschlossenen Dissertation "Architectural Styles and the Design of Network-based Software Architectures" entworfen wurde. REST beschreibt auf abstrakte Weise, wie das World Wide Web idealerweise aufzubauen ist. Die lose Kopplung einzelner Komponenten sowie ihre einheitliche Kommunikation untereinander sind nach Fielding die wesentlichen Voraussetzungen eines effizienten und dynamisch skalierbaren WWW. REST ist in seiner ursprünglichen Form dabei gänzlich unabhängig von Techniken sowie Werkzeugen und spezifiziert lediglich einen theoretischen Ansatz. [vgl. Pautasso et al. 2014, S. 1]

Konzepte

Constraints

Entgegen den meisten Behauptungen ist nicht jede Applikation, die Daten in Abhängigkeit des angefragten URI übermittelt, konform hinsichtlich des REST-Standards. Damit sich ein Service als RESTful bezeichnen darf, muss er Fielding zufolge eine Menge von Constraints erfüllen (vgl. Fielding 2000, S. 76-85). Diese werden im Folgenden erläutert. Zur groben Übersicht dient Abbildung 3.

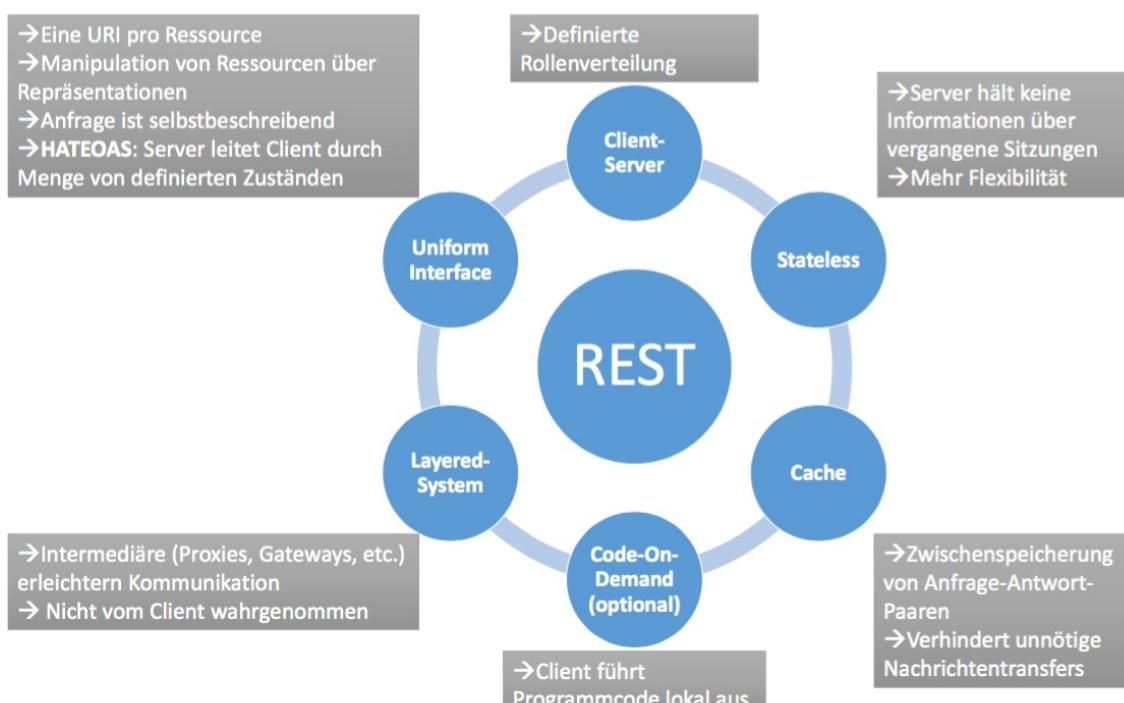


Abbildung 3: Veranschaulichung REST-Constraints

- **Client-Server:** REST kann als hybrider Architekturstil bezeichnet werden, da er auf bereits bestehende Paradigmen, wie beispielsweise das Client-Server-Modell zurückgreift. Für REST ist eine strikte Trennung zwischen der Geschäftslogik des Dienstes und der Schnittstelle des Benutzers fundamental. Die hierdurch erreichte lose Kopplung ermöglicht sowohl die Nutzung von Services unabhängig des verwendeten Endgeräts als auch die getrennte Weiterentwicklung von Client- bzw. Serverkomponenten.
- **Stateless:** RESTful Services müssen zustandslos sein. Die Server-Komponente darf demnach keine Informationen über ihre jeweiligen Clients halten. Dieses Constraint impliziert, dass jeder Aufruf eines Clients sämtliche für die Verarbeitung erforderlichen Informationen beinhalten muss. Der Client hat demnach die Verantwortung seine relevante Daten selber zu verwalten und dem Server bei jedem Aufruf geeignet mitzuteilen. Der Vorteil dieses Verfahrens liegt darin, dass die Performanz des Servers erheblich gesteigert wird. Dieser muss keine aufwändigen Lese- und Schreiboperationen zur Verwaltung der Benutzerdaten durchführen, sondern kann Anfragen ausschließlich anhand ihrer Semantik beantworten. Im Fall eines Serverausfalls bzw. bei unzureichender Lastverteilung ermöglicht das zustandslose Verfahren weiterhin die Umleitung von Clients an eine weitere Server-Instanz, ohne dass diese Kenntnisse bezüglich der Benutzerhistorien haben muss.
- **Cache:** Einen weiteren Beitrag zur Optimierung des World Wide Web leistet Fielding zufolge der Gebrauch von Caching. Ziel ist es, den Server zu entlasten und damit die Performanz von Datenaustauschen zu verbessern. Hierzu platziert sich in der Regel eine weitere Server-Instanz zwischen Client und dem eigentlichen Server bzw. der Client führt das Caching lokal durch. Jede Antwort, die vom Server zurück an den Client versendet wird, ist entweder als cacheable oder non-cacheable zu deklarieren. Eine Antwort, die cacheable ist, wird vom Intermediär inklusive der dazugehörigen Anfrage abgespeichert. Jede weitere Anfrage stellt die Client-Instanz nun an den Intermediär, der diese mit seinen abgespeicherten Anfragen vergleicht. Im Fall einer Übereinstimmung wird die dazugehörige Antwort direkt an den Client geliefert, ohne dass der Server in die Kommunikation involviert ist. Befindet sich die Anfrage nicht im Cache, wird sie zwecks normaler Verarbeitung an den Server weitergeleitet. Caching soll sicherstellen, dass der Server niemals die gleiche Antwort zweimal generieren bzw. übermitteln muss.
- **Uniform Interface:** Einheitliche Schnittstellen zwischen Client- und Server-Instanzen sind eine wesentliche Abgrenzung von REST zu anderen Architekturstilen. Sie verbessern die Nachvollziehbarkeit von Nachrichtenaustauschen für Außenstehende und ermöglichen den Aufruf von Services ohne genauere Kenntnisse der jeweiligen Implementierung. Aufgrund der Allgemeinheit des Constraints vollzieht Fielding eine Verfeinerung in vier Unter-Constraints.
 - *Identification of resources:* Jede Ressource eines Dienstes wird über einen

eindeutigen URI abgebildet.

- *Manipulation of resources through representations*: Wenn der Client Kenntnis von der Repräsentation einer Ressource besitzt, kann er diese nutzen, um die Ressource auf Serverseite zu manipulieren. Hält der Client beispielsweise eine JSON-Repräsentation einer Ressource Bestellung, kann er hiermit die Ressource serverseitig ändern (vorausgesetzt er besitzt die nötigen Zugriffsrechte). Weitere Benutzer, die sich eben diese Bestellung beispielsweise als XML-Objekt repräsentieren lassen, erhalten nun auch die manipulierte Version.
- *Self-descriptive messages*: Jede einzelne Nachricht enthält alle Informationen, die für ihre Verarbeitung nötig sind. Anfragen spezifizieren die Ressource, auf die sich beziehen, und die auszuführende Methode auf dieser Ressource. Im Fall, dass weitere Daten mitgesendet werden, muss zudem der Datentyp deklariert werden. Antworten hingegen sind als cacheable bzw. non-cacheable zu markieren.
- *HATEOAS*: Hypermedia as the engine of application state (HATEOAS) ist ein zentrales Prinzip von REST. Der Kerngedanke dahinter ist, dass der Client Ressourcen nicht selbstständig über URLs abfragt bzw. Kenntnisse von einer API hat, sondern für die gesamte Interaktionsdauer vom Server geführt wird. Nimmt der Client initial Kontakt mit dem Server auf, erhält er eine Antwort, in der spezifiziert ist, welche weiteren Anfragen möglich sind. Tätigt der Client eine dieser Anfragen, erhält er wiederum eine neue Menge von Anfragen, die aus seinem aktuellen Status getätigkt werden können. Diese Prozedur wiederholt sich bis zum Ende der Kommunikation. Eine Antwort seitens des Server enthält somit nicht nur das Ergebnis der eigentlichen Anfrage, sondern auch die nächsten potentiellen Interaktionsmöglichkeiten. Diese Möglichkeiten liefert der Server in der Regel über src- bzw. href-Attribute innerhalb des zurückgelieferten HTML-Objekts aus. Weiterhin kann er dem Client ein Formular senden, in dem dieser nur eingeschränkte Auswahlmöglichkeiten besitzt, was seine potentiellen Aktionen ebenfalls eingrenzt. HATEOAS baut somit auf dem Prinzip des endlichen Automatens auf. Ein Client gelangt in einen Initialzustand, für den eine Menge von Transitionen vom Server festgelegt sind. Mit jeder Transition erhält der Client die Informationen seines neuen Zustands sowie neue Transitionsmöglichkeiten vom Server vorgelegt. Diese Prozedur ist aus dem folgenden Beispiel (siehe Abb. 4) ersichtlich. Der Client gelangt initial auf die Homepage des RESTful Dienstes Versandhandel. Von diesem Zustand ermöglicht der Server ihm ausschließlich die Transition in den Status "Alle Produkte anzeigen". Gelangt er in diesen Status, kann er ein spezifisches Produkt anfragen bzw. eine Rücktransition vollziehen. Lässt er sich ein spezifisches Produkt anzeigen, kann er dieses erwerben bzw. eine Bestellung abschicken oder zurück auf die Startseite wechseln. Jeder Status beinhaltet demnach die tatsächlich angefragten Informationen, wie beispielsweise die Darstellung sämtlicher Produkte, als auch die möglichen Transitionen zu neuen

Status. Welche Status und Transitionsmöglichkeiten existieren, gibt alleine der Server vor. Hierdurch entstehen diverse Vorteile. Die Komplexität auf Seiten des Clients wird erheblich verringert, da dieser keine API verinnerlichen muss, sondern ausschließlich vom Server gelenkt wird. Hierdurch wird auch das Risiko von inkorrekteten Anfragen gesenkt. Weiterhin kann der Server flexibler agieren. Er stellt dem Client ausschließlich die Ressourcen zur Verfügung, die im aktuellen Moment tatsächlich angefragt werden können. Ist ein Produkt nicht mehr im Sortiment, kann sich der Client dieses auch nicht mehr anzeigen lassen. Im Fall, dass der Server ausgelastet ist, kann er den Client dynamisch an eine weitere Server-Instanz verweisen, ohne dass dieser Kenntnis davon hat.

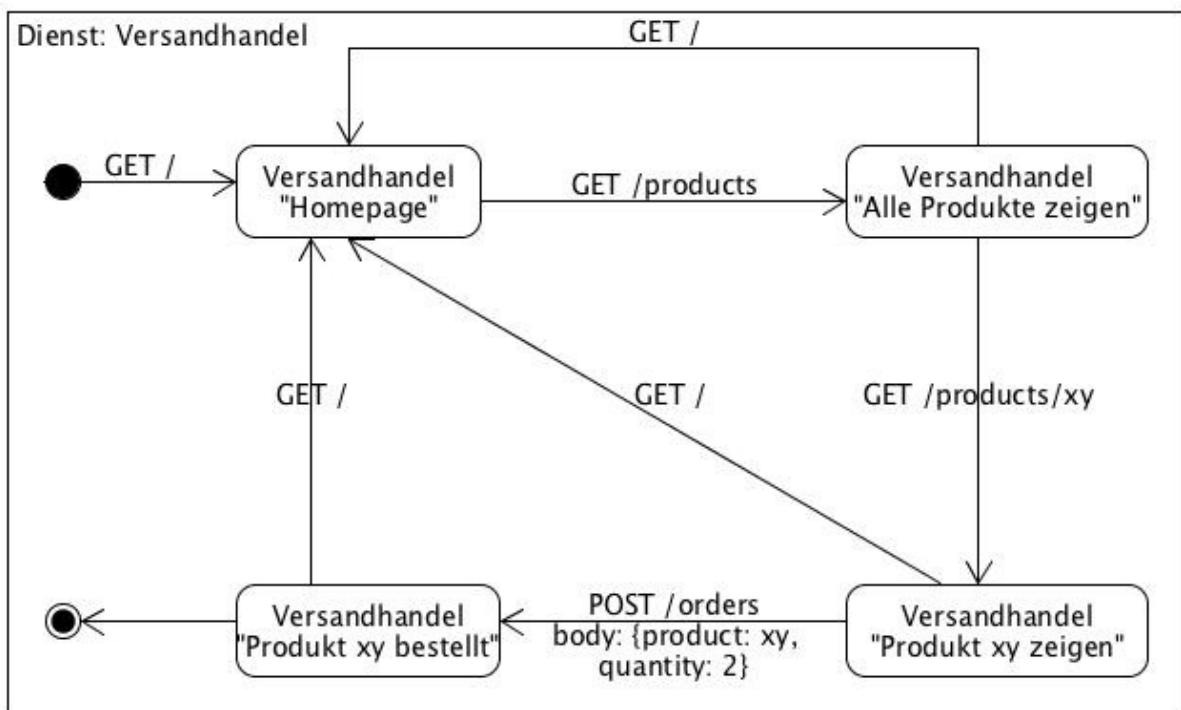


Abbildung 4: HATEOAS am Beispiel Versandhandel

- **Layered-System:** Ein geschichtetes System soll die lose Kopplung innerhalb des World Wide Web weiter vorantreiben. Das Ziel ist es, die Komplexität von Diensten zu verringern, ohne weitere Änderungen an ihnen vorzunehmen. Hierzu werden weitere Schichten zwischen Client und Server gesetzt. Hierunter können Proxy-, Gateway- oder andere Server-Instanzen verstanden werden, die diverse Funktionen, wie beispielsweise Caching, Vermittlung oder Übersetzung von Anfragen, übernehmen. Diese Intermediäre sind so einzusetzen, dass für den Client keine zusätzlichen Anstrengungen bezüglich des Kommunikationsaufbaus entstehen. Er weiß also zu keiner Zeit, ob er direkt mit dem Server oder mit einem Intermediär kommuniziert, da ihre Schnittstellen identisch sind.

- **Code-On-Demand:** Unter Code-On-Demand ist die Übermittlung von Programmcode vom Server an den Client zu verstehen. Sowohl Client als auch Server können hierdurch entlastet werden, da Operationen nun lokal beim Anwender aufgerufen werden können, ohne dass eine Übermittlung von Eingabe-Parametern bzw. Programmergebnissen stattfinden muss. Fielding sieht es jedoch als optionales Constraint an, da es die Nachvollziehbarkeit einer Kommunikation verringert, wenn der Client teilweise seine eigenen Berechnungen durchführt.

Ressourcen und Repräsentationen

Der Austausch von Informationen zwischen Client und Server ist die Grundlage von verteilten Systemen. REST führt hier eine weitere Differenzierung durch und teilt Informationen in Ressourcen und Repräsentationen ein. Ressourcen bilden die Ausgangsbasis und stellen in der Regel Objekte des realen Lebens dar. Beispielhaft für eine Ressource ist ein Mensch oder ein Auto. Auch nicht greifbare Objekte können Ressourcen sein, wie zum Beispiel eine Bestellung. Eine Ressource kann weiterhin durch eine beliebige Anzahl an Repräsentationen beschrieben werden. Eine Repräsentation der Ressource Auto ist beispielsweise ein JSON-Objekt, in dem alle technischen Daten des Autos aufgelistet sind. Auch ein Foto im JPEG-Format, auf dem das Auto abgebildet ist, kann als Repräsentation verstanden werden. Der Vorteil der Abgrenzung von Ressourcen und Repräsentationen ist die daraus resultierende Flexibilität. Der Server speichert alle Daten, die er einer Ressource zuordnen kann in einer Datenbank ab. In Abhängigkeit davon, welche Darstellung der Client bevorzugt, kann er aus diesen Daten dynamisch eine geeignete Repräsentation generieren und übermitteln. Unabhängig davon, welche Repräsentation angefordert wird, bleibt die Aufrufroutine für eine Ressource jedoch immer gleich. Lediglich die übermittelten Informationen ändern sich. HTTP erlaubt beispielsweise, den bevorzugten Typ einer angefragten Ressource in dem Headerfeld der Anfrage mitzusenden. (vgl. Tilkov et al. 2015, S. 35-36)

(Tilkov et al. 2015, S. 36-39) teilen Ressourcen weiterhin in verschiedene Unterkategorien ein, um die Übersichtlichkeit eines RESTful Services zu steigern:

- **Primärressourcen:** Unter Primärressourcen sind Ressourcen im eigentlichen Sinn zu verstehen. Jedes Objekt, welches ein elementarer Bestandteil des Dienstes ist, ist dieser Kategorie zuzuordnen. Ein Dienst, der dem Verkauf von Autos dient, wird somit höchstwahrscheinlich eine Primärressource Auto aufweisen. Jedoch kann auch eine Bestellung oder ein Kunde hier als Primärressource angesehen werden.
- **Subressourcen:** Subressourcen sind die elementaren Bestandteile übergelagelter Ressourcen. Die Primärressource Auto könnte beispielsweise die Subressourcen Motor und Lenkrad umfassen.
- **Listenressourcen:** Listenressourcen kommen immer dann zum Einsatz, wenn man

nicht auf eine einzelne Primär- bzw. Subressource zugreifen möchte, sondern auf die Gesamtheit aller Ressourcen des Typs. Die Listenressource Autos kann beispielsweise alle Primärressourcen Auto umfassen, welche aktuell zum Verkauf angeboten werden.

- **Filter:** Eine nach einem bestimmten Kriterium gefilterte Listenressource ist als Filterressource anzusehen. Beispielhaft dafür seien alle zu verkaufenden Autos mit mehr als 150 PS.
- **Paginierung:** Eine Paginierung teilt eine Listenressource auf einzelne Seiten auf, sodass eine übersichtliche Darstellung erfolgen kann.
- **Projektionen:** Werden nur bestimmte Informationen einer Ressource benötigt, kann eine Projektion erfolgen. Beispielsweise wird für die Listenressource Autos nur das Bild und das Modell des jeweiligen Autos für die Darstellung benötigt. Die Informationen bezüglich der Leistung müssen zwecks Performanz somit nicht übermittelt werden.
- **Aggregationen:** Sollen bestimmte Attribute verschiedenartiger Ressourcen in einer einzelnen Antwort übermittelt werden, ist eine Aggregation durchzuführen.
- **Aktivitäten:** Schritte innerhalb einer Verarbeitung sind den Aktivitäten zuzuordnen. Hierbei handelt es sich um nicht greifbare Tätigkeiten, die dennoch als Ressourcen abgelegt werden. Kauft man beispielsweise ein Auto bei einem Dienst, kann initial eine neue Ressource "SCHUFA-Prüfung" angelegt werden, die die Kreditwürdigkeit feststellen soll.

URLs

Ein Universal Resource Identifier (URI) definiert den Zugriffsort einer einzelnen Ressource in einem Netzwerk. Der Identifikator muss dabei nach einem standardisierten Schema (siehe Abb. 5) aufgebaut sein, welches fünf Teile spezifiziert (vgl. Berners-Lee 2005, S. 16-24).

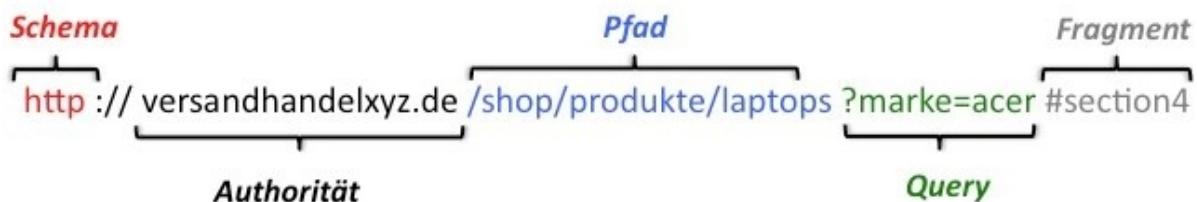


Abbildung 5: Beispielhafter Aufbau eines HTTP-URI

- **Schema:** Unter dem Schema kann das Protokoll verstanden werden, mit dem die jeweilige Ressource anzufragen ist. Hierzu sind neben Hypertext Transfer Protocol (HTTP) und File Transfer Protocol (FTP) auch beispielsweise Lightweight Directory Access Protocol (LDAP) und Teletype Network (Telnet) zu zählen. Für REST wird jedoch in der Regel HTTP verwendet.

- **Authorität:** Die Authorität bezeichnet die Organisation, der die angebotene Ressource unterliegt. In manchen Fällen ist dieser Authorität zusätzlich eine Portnummer nach- bzw. der die Ressource anfragende Benutzer vorgestellt.
- **Pfad:** Der Pfad beschreibt, an welcher Stelle sich die benötigte Ressource innerhalb der Organisation befindet und ist typischerweise hierarchisch aufgebaut. Einzelne Hierarchieebenen werden dabei mit einem Schrägstrich voneinander getrennt.
- **Query:** Optional ist der Pfad um eine Abfrage zu erweitern. Mit einem Kaufmanns-Und sind beliebig viele Schlüssel-Wert-Paare anzuhängen, die vom Server ausgelesen und in die Verarbeitung miteinbezogen werden.
- **Fragment:** Ein Fragment wird ausschließlich auf Seite des Clients aufgelöst und nicht per Anfrage an den Server übertragen. Die konkrete Verwendung des Fragments ist dabei abhängig vom verwendeten Protokoll bzw. von dem Medientyp der Antwort. Wird beispielsweise ein HTTP-URI mit Fragment aufgerufen, der ein HTML-Objekt liefert, scrollt der Browser nach Darstellung der kompletten Webseite automatisch zu dem von dem Fragment beschriebenen HTML-Tag.

Innerhalb von REST sind URLs unerlässlich. Jede Ressource muss über einen definierten URI erreichbar sein. Weiterhin darf sich maximal ein URI auf eine Ressource beziehen. Diese Auflagen garantieren einen eindeutigen und konsistenten Zugriff.

Verben

Die Semantik einer Anfrage an einen RESTful Service ergibt sich nicht ausschließlich aus der Lokalität der angeforderten Ressource, sondern auch aus der Methode, die darauf angewendet werden soll. HTTP bietet eine Vielzahl an Methoden, die potentiell für REST benutzt werden können. In der Praxis hat sich jedoch nur eine Teilmenge an Methoden etabliert, die auf die grundlegenden Datenbankoperationen (Create, Read, Update und Delete) abgebildet werden können. Die Methoden werden im Folgenden nach (Tilkov et al. 2015, S. 53-59) beschrieben.

- **GET:** Die grundlegendste HTTP-Methode ist GET. Ein mit GET angefragter Service liefert eine Repräsentation der durch den URI beschriebenen Ressource an den Client zurück. Der Typ der übermittelten Repräsentation kann bereits bei der HTTP-Anfrage festgelegt werden. Im Fall, dass der Server die deklarierte Repräsentation nicht liefern kann, übermittelt er eine andere Repräsentation seiner Wahl.
- **PUT:** Eine bestehende Ressource kann von einem Client auf dem Server manipuliert werden. Dafür muss der Client geeignete Zugriffsrechte und eine Repräsentation der Ressource besitzen. Die veränderte Repräsentation wird nun im Body einer PUT-Anfrage an den URI der Ressource versendet. Im Header wird der Typ der übermittelten Repräsentation spezifiziert, sodass der Server die Daten geeignet

verarbeiten kann.

- **POST:** Mit dem POST-Befehl teilt der Client dem Server eindeutig mit, dass eine neue Ressource angelegt werden soll. Äquivalent zum PUT-Befehl wird eine Repräsentation im Body der Anfrage mitgeliefert und im Header beschrieben. Ein Hauptunterschied zwischen PUT und POST ist, dass der URI einer PUT-Anfrage auf eine Primär- bzw. Subressource verweist, während der URI eines POST-Befehls eine Listenressource beschreibt, in die das neue Element einzufügen ist.
- **DELETE:** Soll eine Ressource gelöscht werden, ist der DELETE-Befehl mit dem dazugehörigen URI zu verwenden. In der Regel wird die Ressource jedoch nicht physikalisch aus dem Speicher des Servers entfernt, sondern lediglich als nicht mehr existent gekennzeichnet.

Weiterhin können die beschriebenen Methoden hinsichtlich diverser Faktoren kategorisiert werden. Ein Faktor ist die Sicherheit. Ist eine Methode als sicher einzustufen, kann sie ausgeführt werden, ohne dass dies Auswirkungen auf die gespeicherten Daten des Servers hat. Ein GET-Befehl manipuliert beispielsweise keine serverseitigen Informationen, weshalb er ohne Bedenken genutzt werden kann. Ein PUT-Befehl ist im Gegensatz dazu unsicher, da durch ihn potentiell weitreichende Manipulationen durchgeführt werden können. Idempotenz liegt dann vor, wenn eine Methode beliebig oft auf denselben URI angewendet werden kann, und lediglich die erste Anfrage Auswirkungen hat. Führt man beispielsweise den PUT-Befehl mit gleichem URI und gleichem Body zweimal durch, wirkt nur die erste Anfrage. Die zweite Anfrage wird zwar vom Server abgearbeitet, hat aber keine Auswirkungen, da das Update der Ressource bereits geschehen ist. Ein zweimaliges Posten derselben Anfrage führt im Gegensatz zur Anlage zwei neuer Ressourcen und ist somit nicht idempotent. Die Kenntnis über die Idempotenz einer Methode kann beispielsweise bei Verbindungsstörungen hilfreich sein. Wurde eine Anfrage abgeschickt, ohne dass eine Antwort eingegangen ist, können idempotente Anfragen ohne Sorge wiederholt ausgeführt werden. Selbst wenn der Server die erste Anfrage bereits verarbeitet hat und die Antwort lediglich aufgrund von Latenzproblemen noch nicht zugestellt wurde, hat das nochmalige Senden keine Auswirkungen. Wie schon angeschnitten, unterscheiden sich Methoden bezüglich ihrer Adressierung. Ein GET-Befehl wird immer direkt auf eine einzelne Ressource angewendet und identifiziert diese dementsprechend. Im Gegensatz dazu steht eine POST-Anfrage, die lediglich auf eine Liste angewendet wird und dementsprechend nicht auf eine konkrete Instanz einer Ressource. Letztlich ist eine Differenzierung der Methoden im Hinblick auf ihre Cache-Fähigkeit zu vollziehen. Ein GET-Befehl ist beispielsweise cache-fähig, da er ausschließlich Daten zurückliefert. Diese müssen nicht zwingend vom Server stammen, sondern können auch durch einen Intermediär bereitgestellt werden. Befehle, die Manipulationen auf dem Server einfordern, müssen zwangsweise an diesen zugestellt werden und sind dementsprechend nicht von einer zwischengeschalteten Instanz zu verarbeiten. Sie dienen jedoch der Invalidierung. Wird beispielsweise ein PUT auf eine

Ressource ausgeführt, die sich im Cache befindet, ist diese anschließend aufgrund der Manipulation aus dem Cache zu entfernen. Eine detaillierte Gegenüberstellung der HTTP-Methoden bezüglich der genannten Bewertungsfaktoren ist aus Tabelle 2 ersichtlich.

Methode	Sicher	Idempotent	Identifiziert einzige Ressource	Cache-fähig
GET	Ja	Ja	Ja	Ja
PUT	Nein	Ja	Ja	Nein
POST	Nein	Nein	Nein	Nein
DELETE	Nein	Ja	Ja	Nein

Tabelle 2: Vergleich der grundlegenden HTTP-Methoden

Quelle: In Anlehnung an (Tilkov et al. 2015, S. 59)

Caching

Wie sich im vorherigen Abschnitt bereits herausgestellt hat, ist Caching ein essentieller Bestandteil von REST. Die Zwischenspeicherung von Anfrage-Antwort-Paaren soll zum einen den Client unterstützen. Erhält dieser seine angeforderten Informationen, ohne dass sie auf Seiten des Servers aufwändig generiert werden mussten, kann er Zeit einsparen und seine Performanz steigern. Zum anderen wird auch der Server durch Caching entlastet, da ihn weniger Anfragen erreichen. Ohne zu tief in Implementierungsdetails zu gehen, kann ein einzelnes Caching-Verfahren für RESTful Services anhand von zwei Dimensionen charakterisiert werden, wobei auch hybride Ansätze möglich und praktikabel sind. (Tilkov et al. 2015, S. 127-133) definieren diese Dimensionen als Modell und Topologie.

Modell

Das Modell legt fest, von welcher Instanz das Caching initiiert wird (siehe Abb. 6). Beim **Expirationsmodell** erweitert der Server jede von ihm ausgelieferte Nachricht um den Cache-Control-Header. In diesem wird vermerkt, wie lange die übermittelte Repräsentation bzw. die dazugehörige Ressource aktuell ist. Der Server spezifiziert damit, dass sich die jeweilige Ressource in dem festgelegten Zeitraum (wahrscheinlich) nicht ändern wird und ermöglicht somit die zielführende Zwischenspeicherung des bereits ausgelieferten Ergebnisses. Nach Ablauf der Zeit sind Anfragen bezüglich der Ressource wieder konventionell an den Server zu richten, bis möglicherweise wieder ein Cache-Control-Header empfangen wurde. Der Vorteil dieses Modells liegt in seiner Effizienz. Es ist kein zusätzlicher Datenverkehr notwendig, um den Client auf die Gültigkeit von Repräsentationen bzw. Ressourcen aufmerksam zu machen. Hierunter leidet jedoch in manchen Fällen die Aktualität von zwischengespeicherten Antworten. Hat sich trotz Versprechen des Servers

eine Ressource während der Caching-Zeitspanne verändert, bleibt der Client davon uninformiert und greift weiterhin auf die veralteten Daten zu. In Abhängigkeit des Kontexts kann dies mehr oder weniger schwere Konsequenzen nach sich ziehen. Beim **Validierungsmodell** geht die Interaktion hingegen von Seiten des Clients aus. Die nochmalige Anfrage einer Ressource, von der er bereits eine Repräsentation besitzt, erweitert er um zusätzliche Informationen. Typischerweise ist dies ein Zeitstempel von der erstmaligen Anforderung. Der Server prüft nun, ob die Ressource seit dem angegebenen Zeitpunkt einer Änderung unterlag. Im Fall, dass die Ressource konstant blieb, ist die nochmalige Übertragung nicht nötig. Der Server teilt dem Client nun lediglich mit, dass er mit der früheren Repräsentation weiterarbeiten kann. Sollte jedoch eine Änderung aufgetreten sein, wird die neue Repräsentation im Body mitgeschickt. Anstatt eines Zeitstempels ist es weiterhin möglich, den Hashwert einer Repräsentation zu übergeben. Der Server kann auch hiermit feststellen, ob die clientseitige Repräsentation der Ressource noch immer mit der serverseiten Repräsentation übereinstimmt. Der Vorteil vom Validierungsmodell liegt darin, dass der Client stets mit aktuellen Repräsentationen arbeitet, auch wenn sie nicht erneut durch den Server übertragen wurden. Der Nachteil ergibt sich jedoch dadurch, dass die Einsparung der Kommunikationsaufwände im Vergleich zum vorherigen Modell geringer ausfällt. Der Server muss in jedem Fall die Anfrage des Clients verarbeiten. Effizienz wird lediglich dadurch gewonnen, dass die Antwort teilweise verschlankt wird bzw. keine Repräsentation auf Serverseite zu generieren ist.

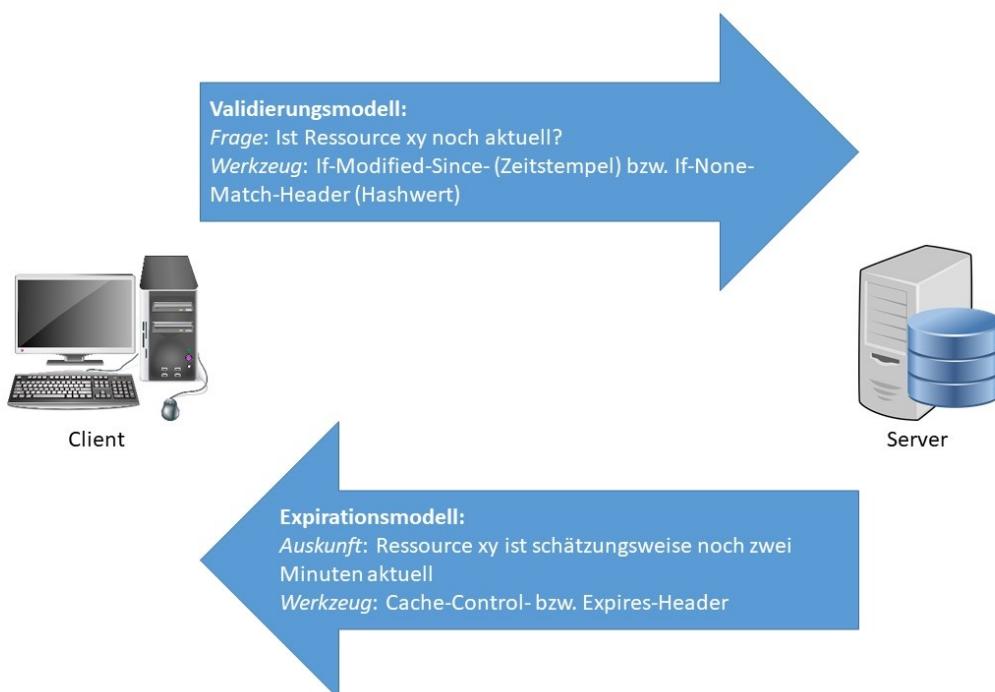


Abbildung 6: Veranschaulichung Caching-Modelle

Topologie

Die Topologie eines Caching-Verfahren definiert, an welcher Lokalität das Caching stattfindet bzw. wo der Zwischenspeicher angesiedelt ist (siehe Abb. 7). Beim **clientseitigen Caching** (typischerweise in jedem Webbrower implementiert) besitzt jeder Client, der mit einer Server-Instanz kommuniziert, seinen eigenen Cache. Der Vorteil liegt darin, dass auch verschlüsselte bzw. vertrauliche Anfragen und Antworten zu cachen sind, da keine Sicherheitsrisiken bezüglich weiterer Nutzer entstehen. Der **clientseitige Shared Cache** bzw. **Proxy-Cache** erweitert diese Topologie. Der Zwischenspeicher gilt nun für eine beliebige Anzahl an Benutzern und ist typischerweise auf einem zwischengelagerten Proxy-Server befindlich. Die Vielzahl der anfragenden Clients erhöht die Chance für das Caching einer später benötigten Repräsentation. Beim **Server-Caching** wird jede an den Server gestellte Anfrage zwischengespeichert und für alle nachkommenden Clients verfügbar gemacht. Zwar erhöht sich die Chance, dass eine Anfrage ohne Berechnung auf Serverseite beantwortet werden kann, jedoch steigen auch die Aufwände. Anfragen sind stets durch das gesamte Netzwerk hindurch zu leiten, bevor sie überprüfbar sind. In der Praxis werden alle Topologien miteinander verknüpft, um bestmögliche Caching-Resultate zu erzielen.

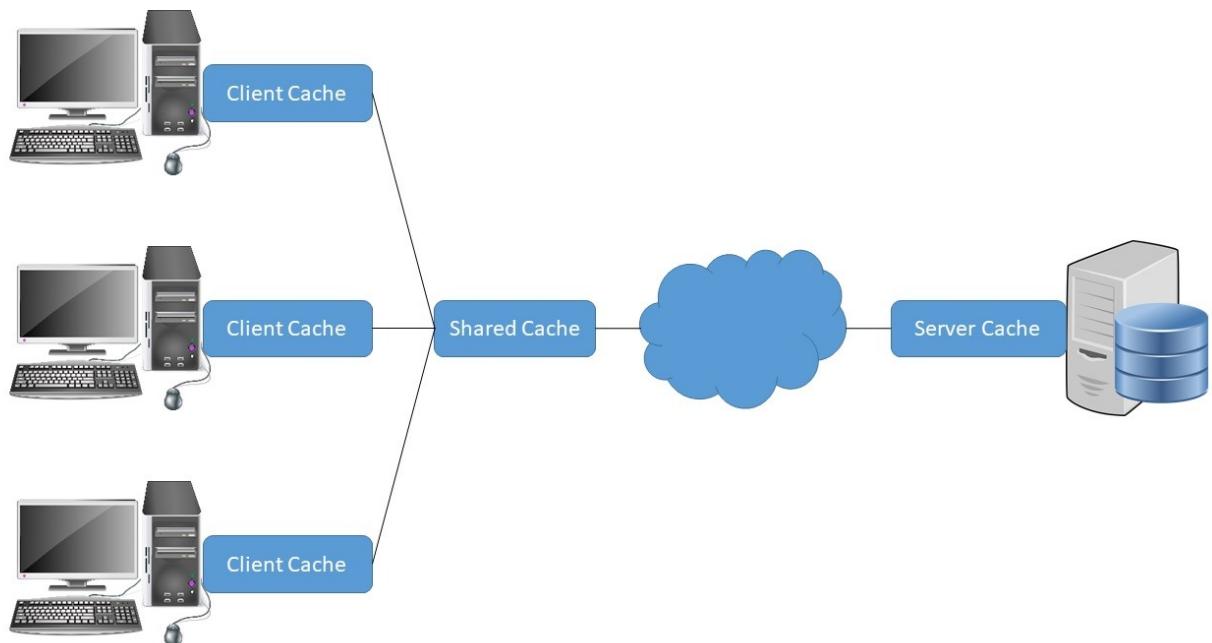


Abbildung 7: Veranschaulichung Caching-Topologien

Quelle: In Anlehnung an (Tilkov et al. 2015, S. 133)

Security

Sicherheit ist für verteilte Systeme und insbesondere RESTful Services ein zentrales Thema. Letztere bieten aufgrund ihrer meist öffentlich zugänglichen Schnittstelle eine breite Fläche für potentielle Angreifer und sind somit zu schützen. (Levin 2016) definiert dabei fünf verschiedene Maßnahmen, deren Umsetzung einen sicheren Dienst gewährleisten soll.

1. **Authorisierung:** Wie bereits erwähnt, sind RESTful Services oft öffentlich zugänglich. Dies sei gewollt, um einen Dienst möglichst vielen Nutzern zur Verfügung zu stellen und somit seine Popularität zu steigern. Mithilfe von HATEOAS leitet der Server den spezifischen Client durch eine Menge von für ihn verfügbaren Zuständen. Benutzer, die erweiterte Kenntnisse bezüglich der REST-Schnittstelle haben, können jedoch selbstständig Anfragen an Ressourcen stellen. Obwohl diese Herangehensweise vom Server ungewollt ist, kann er sie nicht unterbinden. Um trotzdem zu verhindern, dass jeder Benutzer potentiell dazu in der Lage ist, jede existierende Ressource einzusehen bzw. zu manipulieren, muss eine Authorisierung erfolgen. Der Client erweitert jede Anfrage um ein Authorisierungsobjekt (falls er ein solches besitzt), das in Abhängigkeit des angefragten Dienstes als API- bzw. Session-Key oder Benutzername-Passwort-Paar anzusehen ist. Der Server entscheidet anschließend, ob die beantragte Operation auf der Ressource zu bewilligen ist und führt diese ggf. aus. Welche Benutzergruppen für welche Aktionen befähigt sind, hängt hingegen allein von der Geschäftslogik ab.
2. **Eingabeverifikation:** Auch autorisierte Clients sind nicht zwingend vertrauenswürdig und können dem RESTful Service schaden. Insbesondere durch die Nutzung von POST- und PUT-Befehlen ergeben sich für sie diverse Angriffsmöglichkeiten. Eine simple HTTP-Anfrage, die per POST gestellt wurde, beinhaltet eine Vielzahl an Informationen. Der URI verrät, an welche Ressource die Operation gerichtet ist. Eine breite Anzahl von Headern erweitert diese Information um verschiedenartige Details, die sich sowohl auf die Syntax als auch auf die Semantik der Anfrage beziehen. Im Body kann weiterhin jede erdenkliche Zeichenfolge mitgeliefert werden. Schädlinge können diese Freiheit nutzen, um eine Anfrage derart zu formulieren, dass sie auf dem Server Schaden anrichtet. Cross-Site-Scripting und SQL-Injection sind hier nur zwei mögliche Angriffsmethoden. Um diesen entgegenzuwirken, sollte der Server initial den angefragten URI sowie die beigefügten Header validieren. Sind die diese nicht gültig bzw. entsprechen nicht dem typischen Schema, ist die gesamte Anfrage zu verwerfen. Weiterhin muss auch der Inhalt der Anfrage geeignet verarbeitet werden. Der Server sollte sich dabei nicht ausschließlich auf den vom Client angegebenen Content-Type verlassen, sondern den Body immer explizit mit diesem abgleichen. Bei fehlender Übereinstimmung ist die weitere Verarbeitung der Anfrage ggf. mit Gefahr verbunden. Für die weitere Nutzung ist der Body-Inhalt immer mit einem sicheren Parser zu verarbeiten, sodass weitere Risiken minimiert werden. Bei der Erstellung neuer oder der Manipulation von bestehenden Ressourcen sollte der Server schließlich stets strikte Wertebereiche für einzelne Attribute vorgeben, um Angriffsmöglichkeiten weiterhin einzuschränken.
3. **Ausgabeverifikation:** Angreifer können nicht nur dem Server schaden, sondern auch allen weiteren Clients, die mit diesem kommunizieren. Liefert der Server an diese eingeschleuste und schadhafte Inhalte aus, sind Sicherheitsrisiken die Folge. XML- oder JSON-Objekte sollten vom Server immer mit einem Serializer erzeugt werden, um

beispielsweise SQL-Injection zu verhindern. Der mitgelieferte Content-Type muss in jedem Fall mit dem Body-Inhalt der Antwort übereinstimmen, damit die Antwort auf Clientseite geeignet verarbeitet werden kann.

4. **Kryptographie:** Auch die ausgetauschten Nachrichten zwischen Client und Server sind zu schützen, sodass weder Lese- noch unbemerkte Schreiboperationen von Unbefugten während der Übermittlung möglich sind. Hierzu wird typischerweise auf das Protokoll TLS (Transport Layer Security) zurückgegriffen. TLS ist der Nachfolger von SSL (Secure Sockets Layer) und besteht aus den zwei Unterprotokollen TLS Handshake Protocol und TLS Record Protocol, welche den Verbindungsauflaufbau zwischen Client und Server bzw. deren Nachrichtenaustausch absichern sollen. Durch den Einsatz von TLS ist die Kommunikation für Dritte nicht einseh- oder manipulierbar. Weiterhin sind sensitive Daten, welche im Speicher des Servers befindlich sind, gegen Angriffe zu schützen. Kreditkartennummern, Adressen und andere benutzerspezifische Informationen dürfen ausschließlich verschlüsselt abgelegt werden. Dabei ist darauf zu achten, dass die zugehörigen Schlüssel nicht an der derselben Lokalität befindlich sind. Passwörter sind hingegen immer als Hashwerte ihrer selbst aufzubewahren.
5. **HTTP-Status-Codes:** HTTP-Status-Codes dienen dazu, den Client über den (Miss-)Erfolg seiner gesendeten Anfrage zu informieren. Wird bei der Verarbeitung differenziert auf Client-Anfragen eingegangen und spezifische HTTP-Status-Codes erzeugt, hilft dies nicht nur dem anfragenden Benutzer bei der Fehlersuche, sondern unterstützt auch die Identifizierung von möglichen Sicherheitsproblemen auf der Serverseite.

Richardson Maturity Model

Um die REST-Konformität eines Services zu bestimmen, dient das Richardson Maturity Model, welches von Leonard Richardson entwickelt wurde. Ein Service wird dabei einem spezifischen Level zugeordnet, wobei vier verschiedene Level existieren. Jedes Level beinhaltet dabei alle Forderungen der untergeordneten Level inklusive seiner eigenen Forderung. Tabelle 3 veranschaulicht die einzelnen Abstufungen nach (Fowler 2010).

Level	Bedeutung	Bemerkung
0	Der Service besitzt ausschließlich einen Endpunkt. Anfragen jeglicher Art werden an diesen URI gestellt.	Kein REST - mit RPC oder SOAP vergleichbar.
1 - Ressourcen	Der Service ordnet jeder existierenden Ressource einen eigenen URI zu.	Schwaches REST - kein Gebrauch der HTTP-Semantik. Lesende und manipulierende Operationen werden über denselben HTTP-Befehl abgebildet (beispielsweise POST zum Einsehen, Aktualisieren, Erstellen und Löschen von Ressourcen).
2 - Verben	Der Service versteht und unterscheidet die Semantik der verschiedenen HTTP-Verben und wendet diese dementsprechend auf seine Ressourcen an.	Mittelmäßiges REST - Clients müssen Kenntnis bezüglich der Schnittstelle besitzen, um Anfragen zu stellen.
3 - Hypermedia	HATEOAS - Server führt Client durch Menge von Zuständen, ohne dass dieser Wissen über die Schnittstelle besitzen muss.	REST

Tabelle 3: Stufen des Richardson Maturity Model

Anwendung

Best Practices

Fielding definiert mit dem Architekturstil REST Richtlinien, an denen sich Webservices idealerweise orientieren sollten. Diese Richtlinien sind jedoch absichtlich so grob gehalten, dass sie Raum für verschiedenartige Umsetzungen lassen. In der Vergangenheit hat sich jedoch nach (Sahni 2015) eine Menge von Ansätzen etabliert, mithilfe der RESTful Dienste ihr maximales Potential bezüglich Benutzer- und Entwicklerfreundlichkeit ausschöpfen können. Eine Auswahl dieser Ansätze wird im Folgenden vorgestellt.

Einheitliche Verwendung von URIs und HTTP-Methoden: Auch wenn der Nutzer nach HATEOAS keine direkte Kenntnis von der API des Dienstes haben sollte, sind die Routen nach einem konsistenten Schema zu bestimmen. Der Pfad eines URI ist immer hierarchisch

aufzubauen. Er beginnt mit der übergeordneten Ressource und wird immer weiter durch untergeordnete Ressourcen verfeinert. Weiterhin sind Ressourcen innerhalb des URI immer als Nomen zu formulieren, wobei es üblich ist, Listenressourcen im Plural anzugeben. Die Operationen, die auf den Ressourcen ausgeführt werden, sind ausschließlich über die HTTP-Methoden abzubilden (siehe Tabelle 4). Auch serverseitige Funktionen, welche nicht als eigene Ressource existieren, sind nach dem definierten Schema aufzurufen. Die Funktion *produktBewerten(int bewertung, int produktID)* könnte beispielsweise über eine POST-Anfrage an den URI <http://versandhandelxyz/produkte/{productID}/bewertungen> gestellt werden, wobei die neue Bewertung im Body befindlich ist. Ab einem gewissen Grad der Komplexität sollten dem URI Query-Parameter beigelegt werden. In Abhängigkeit davon, ob eine Paginierung, Sortierung, Filterung oder Projektion auf der anforderten Ressource durchzuführen ist, sind diese auszuwählen.

Ressource / Methode	GET	PUT	POST	DELETE
/produkte	Liefert Repräsentationen aller vorhandenen Produkte aus.	Aktualisiert alle Ressourcen in einer Anfrage (eher unüblich).	Erstellt ein neues Produkt, wobei Server die ID selbstständig vergibt.	Logisches Löschen aller Produkte.
/produkte/33	Liefert die Repräsentation eines speziellen Produktes aus.	Aktualisiert eine spezifische Ressource, falls diese existiert. Andernfalls tritt ein Fehler auf.	Fehler, da POST nicht auf existierende Ressourcen angewendet werden kann.	Logisches Löschen eines Produktes.

Tabelle 4: Typische Semantik von HTTP-Methoden bezüglich URLs

Benachrichtigung des Clients: Der Client sollte auf jede seiner Anfragen eine Antwort erhalten. Dabei ist es unwichtig, mit welcher HTTP-Methode die Anfrage verschickt wurde bzw. welche Ressource sie adressierte. Jede Antwort ist weiterhin mit einem geeigneten Statuscode zu versehen. Die HTTP-Statuscodes können grob in fünf verschiedene Kategorien eingeteilt werden (siehe Abb. 8). Diese Einteilung ist jedoch nicht ausreichend, um den Client geeignet über die Verarbeitung seiner Anfrage zu informieren. Beispielsweise macht es für ihn einen Unterschied, ob seine Anfrage fehlerhaft aufgebaut war (HTTP-Statuscode 400), die Ressource nicht gefunden wurde (404) oder die spezifizierte Methode nicht auf der Ressource ausgeführt werden darf (405). Dementsprechend ist der Statuscode möglichst differenziert auszuwählen, um dem Client ein Höchstmaß an Unterstützung zu bieten. Weiterhin ist es gängige Praxis, dem Client bei manipulierenden Anfragen (durch

POST und PUT) mit einer Repräsentation der erstellten bzw. aktualisierten Ressource zu antworten. Diese Repräsentation kann er direkt in sein System eingepflegen, ohne eine weitere Anfrage tätigen zu müssen.

1xx - Informational

2xx - Success

3xx - Redirection

4xx - Client Error

5xx - Server Error

Abbildung 8: Kategorisierung der HTTP-Statuscodes

Quelle: (Reich 2017)

Dokumentation der API: Insbesondere bei großen Webservices können APIs schnell komplex und unübersichtlich werden. Um die Entwickler zu unterstützen, ist eine einheitliche API-Dokumentation unerlässlich. Insbesonders die Kommunikation zwischen Frontend- und Backendentwicklern wird hierdurch vereinfacht. Idealerweise ist eine API-Dokumentation sowohl vom Menschen lesbar als auch vom Server zu interpretieren. Änderungen in dem zentralen Dokument haben dadurch nun gleichermaßen Auswirkungen auf die tatsächliche und die vom Benutzer wahrgenommene API.

API-Dokumentation mit Swagger

Swagger ist ein verbreitetes Open-Source-Framework, das zur Dokumentation und Generierung von REST-APIs genutzt wird. Das Framework beinhaltet dafür drei Grundfunktionalitäten, welche im Folgenden nach (Swagger 2017) erläutert werden.

Swagger Editor: Der Kern einer jeden Swagger-API ist eine Datei im YAML- bzw. JSON-Format, in denen die API nach einem von der OpenAPI Specification (OpenAPI Specification 2017) festgelegten Schema beschrieben ist. Die Generierung der Datei kann automatisch über Annotations im Quellcode oder manuell erfolgen. Für Letzteres stellt Swagger den sogenannten Swagger Editor bereit. Diese Applikation bietet diverse Hilfsfunktionalitäten bei der Erstellung der YAML- bzw. JSON-Dateien, wie beispielsweise Auto vervollständigung oder Live-Visualisierung.

Eine API (beispielsweise im YAML-Format) lässt sich dabei grob in drei verschiedene Abschnitte einteilen. Im Abschnitt *Tags* (siehe Code-Beispiel 2) werden die Ressourcen aufgeführt, auf die die API Zugriff gewährt. Beispielhaft seien hierfür Pet, Store und User.

```
tags:  
# 1. Ressource Pet  
- name: "pet"  
  description: "Everything about your Pets"  
  externalDocs:  
    description: "Find out more"  
    url: "http://swagger.io"  
# 2. Ressource Store  
- name: "store"  
  description: "Access to Petstore orders"  
# 3. Ressource User  
- name: "user"  
  description: "Operations about user"  
  externalDocs:  
    description: "Find out more about our store"  
    url: "http://swagger.io"
```

Code-Beispiel 2: Definition von Tags in YAML-API

Quelle: (Swagger 2017)

Im zweiten Abschnitt erfolgt die Definition der API-Routen (siehe Code-Beispiel 3). Für jede Route werden gültige HTTP-Operationen spezifiziert. Jedes Paar von URI und Methode ist weiterhin mit diversen Informationen anzureichern. Anfänglich wird beschrieben, auf welche Ressource bzw. Tag sich das Paar bezieht. Im Anschluss werden die Eingabe- bzw. Ausgabedaten der Route spezifiziert und mögliche Fehlermeldungen in Form von HTTP-Statuscodes aufgeführt und erläutert. Abschließend erfolgt die Definition von Nutzerrollen, denen eine Anfrage erlaubt ist.

```

paths:
  # Mit der Route "/pet" ...
  /pet:
    # ... und der Methode POST ...
    post:
      tags:
        # ... kann auf die Ressource Pet zugegriffen werden.
        - "pet"
      summary: "Add a new pet to the store"
      description: ""
      operationId: "addPet"
      consumes:
        - "application/json"
        - "application/xml"
      produces:
        - "application/xml"
        - "application/json"
      parameters:
        - in: "body"
          name: "body"
          description: "Pet object that needs to be added to the store"
          required: true
          schema:
            $ref: "#/definitions/Pet"
      responses:
        405:
          description: "Invalid input"
      security:
        - petstore_auth:
          - "write:pets"
          - "read:pets"
  # Mit der Route "/pet" und der Methode PUT ...
  put:
    tags:
      # ... kann auf die Ressource Pet zugegriffen werden
      - "pet"

  ...

```

Code-Beispiel 3: Definition von Routen in YAML-API

Quelle: (Swagger 2017)

Der letzte Teil des Dokuments wird durch die Definitionen gebildet (siehe Code-Beispiel 4). Die Repräsentationen, die vom Server entgegengenommen bzw. ausgeliefert werden, sind hier im Hinblick auf ihren Aufbau bzw. ihre Attribute zu spezifizieren.

```

definitions:
  Pet:
    type: "object"
    required:
      - "name"
      - "photoUrls"
    properties:
      id:
        type: "integer"
        format: "int64"
      category:
        $ref: "#/definitions/Category"
      name:
        type: "string"
        example: "doggie"
      photoUrls:
        type: "array"
        xml:
          name: "photoUrl"
          wrapped: true
          items:
            type: "string"
      tags:
        type: "array"
        xml:
          name: "tag"
          wrapped: true
          items:
            $ref: "#/definitions/Tag"
      status:
        type: "string"
        description: "pet status in the store"
        enum:
          - "available"
          - "pending"
          - "sold"
    xml:
      name: "Pet"

```

Code-Beispiel 4: Definition von Datenobjekten in YAML-API

Quelle: (Swagger 2017)

Swagger Codegen: Mithilfe von Swagger Codegen kann die bereits erstellte API-Spezifikation automatisch zur Generierung von Programmcode auf Server- sowie auf Clientseite genutzt werden. Für jedes Paar von Route und Methode wird auf dem Server eine prototypische Methode erstellt, die vom Entwickler lediglich mit der Programm- bzw.

Geschäftslogik zu befüllen ist. Gleichzeitig werden für Clients individuelle Programmhbibliotheken generiert, mit der sie unkompliziert auf die API zugreifen können. Bisher unterstützt Swagger Codegen über 40 unterschiedliche Programmiersprachen.

Swagger UI: Das Gegenstück zu Swagger Codegen bildet Swagger UI. Hiermit ist es möglich, die vorliegende YAML- bzw. JSON-Spezifikation für Benutzer geeignet zu visualisieren. Mithilfe einer grafischen Oberfläche sind Ressourcen, URLs, Methoden und weitere Informationen schnell einsehbar. Einzelne Elemente können zwecks übersichtlicherer Darstellung ausgeblendet werden. Weiterhin ermöglicht Swagger UI dem Benutzer das Testen von API-Routen. Durch das Betätigen von speziellen Schaltflächen können ohne programmatische Aufwände Anfragen an den Server gesendet sowie dessen Antworten eingesehen werden. Eine Fehlersuche ist somit einfach und effizient möglich. Letztlich stellen Swagger Codegen und Swagger UI bei gemeinsamer Nutzung sicher, dass die Implementierung einer API zu jedem Zeitpunkt konsistent zu ihrer Dokumentation bzw. Visualisierung ist. Sowohl die Entwickler als auch die Nutzer einer API profitieren von diesem Umstand.

Fazit

REST ermöglicht eine strukturierte Kommunikation innerhalb von verteilten Systemen, welche in den meisten Fällen über HTTP realisiert wird. Nach (Tilkov et al. 2015, S. 2-4) sind als Vorteile des Architekturstils die lose Kopplung und Interoperabilität von Rechenknoten anzusehen. Die Wiederverwendung von Diensten sowie ihre hohe Performanz und Skalierbarkeit sind ebenfalls als positiv aufzufassen. Ein Nachteil von REST ist die fehlende Standardisierung, sodass sich viele Dienste als RESTful bezeichnen, welche die zentralen Constraints nicht erfüllen. Auch die Starrheit bezüglich der Anfrage von Ressourcen bzw. Repräsentationen ist als negativ einzustufen. Die Vor- und Nachteile sind abschließend in Abbildung 9 veranschaulicht.



Abbildung 9: Vor- und Nachteile von REST

GraphQL

GraphQL ist eine Abfragesprache für strukturierte Daten, welche 2012 von Facebook entwickelt und 2015 anschließend veröffentlicht wurde (vgl. GraphQL 2017). Ziel von GraphQL ist es, die Datenübermittlung zwischen Server und Client einfacher und effizienter zu gestalten. Neben der Referenz-Implementierung in JavaScript existieren Bibliotheken für andere Programmiersprachen, wie beispielsweise C#, Java oder Python.

Probleme von REST

Dienste, die streng nach dem Architekturstil REST aufgebaut sind, können zum Flaschenhals für eine Client-Server-Interaktion werden. Dies beruht nach (Facebook 2015) auf drei Tatsachen.

Starrheit der Daten: Ein RESTful Dienst legt statisch fest, welche Ressourcen bzw. Repräsentationen angefragt werden können. Er spezifiziert dazu jeweils die URLs, auf die sich der Client beziehen muss, sowie die Operationen, die ausgeführt werden können. Folgendes Szenario verdeutlicht dabei ein mögliches Problem (siehe Abbildung 10). Der Client möchte beispielsweise wissen, mit welchem Wetter eine bestimmte Person morgen bei der Arbeit rechnen muss. Der Server bietet ihm dafür drei Endpunkte, mithilfe der er Personen, Arbeitgeber und Standorte abfragen kann. Um seine Ausgangsfrage zu beantworten, muss der Client nun drei verschiedene Anfragen absenden und jeweils auf die Ergebnisse warten. Dieser erhöhte Aufwand resultiert daher, dass der Server keinen Endpunkt anbietet, anhand dessen die Problemstellung innerhalb lediglich einer Anfrage bzw. Antwort gelöst werden kann. Eine REST-API zu konstruieren, die auf jede nur mögliche verschachtelte Anfrage innerhalb einer Nachricht antwortet, ist für mittlere bis große Dienste nicht machbar.

Frage des Clients: Mit welchem Wetter muss Person 123 morgen bei der Arbeit rechnen?

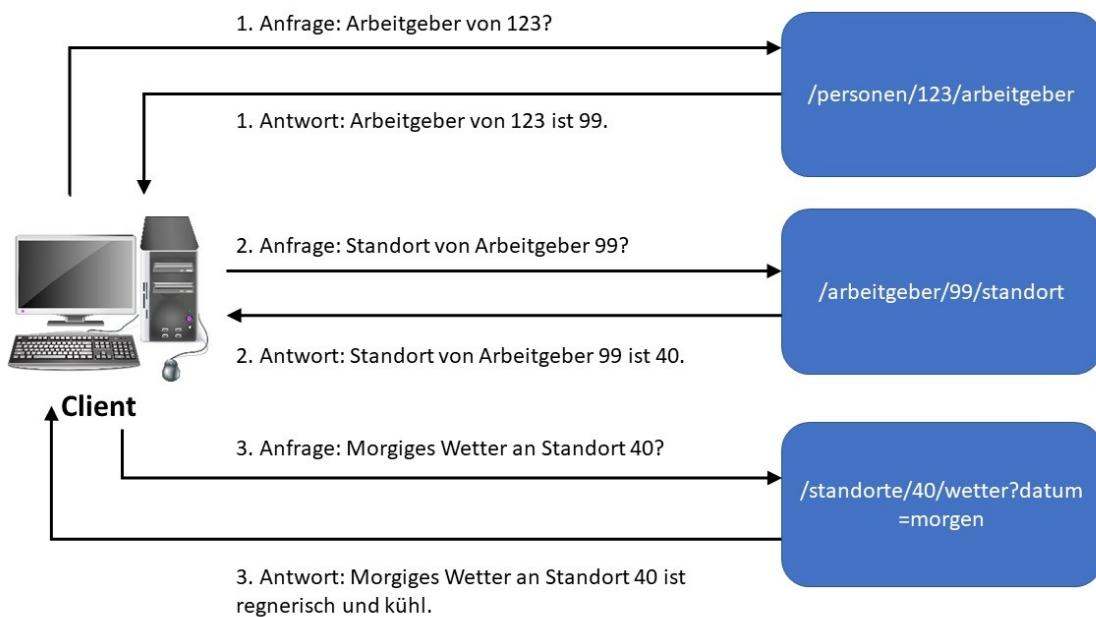


Abbildung 10: Mehrmaliges Anfragen als Nachteil von REST

Fehlende Versionierung: Die Server-Komponenten eines RESTful Dienstes entwickeln sich kontinuierlich weiter. Insbesondere die Ressourcen bzw. Repräsentationen unterliegen dabei häufig Veränderungen in Form von neuen Datenfeldern. In der Regel greifen jedoch heterogene Clients auf den Dienst zu, welche sich im Hinblick auf ihre Bedürfnisse voneinander unterscheiden. Manche von diesen Benutzern benötigen beispielsweise die hinzugefügten Datenfelder als Antwort auf ihre Anfrage, während die Datenfelder für ältere Clients keine Bedeutung haben könnten und nur zusätzlicher Ballast sind. Dadurch bedingt, dass der Server lediglich einen Endpunkt für jede Ressource anbietet, erhalten alle Clients dieselbe Version der Repräsentation. Auf individuelle Ansprüche wird somit keine Rücksicht genommen.

Schwache Typisierung: RESTful Services empfangen Informationen häufig in der Form von JSON- oder XML-Dokumenten. Diese müssen lediglich eine bestimmte Syntax erfüllen, um vom Server akzeptiert zu werden. Einzelne Datenfelder innerhalb der Dokumente sind jedoch nur schwach typisierbar. Für den Server bedeutet es daher einen erheblichen Mehraufwand, jede übermittelte Repräsentation auf ihre semantische Korrektheit zu überprüfen.

Abhilfe

Ein GraphQL-Server wird von jedem Client und unabhängig von den angefragten Daten über denselben Endpunkt erreicht (siehe Abb. 11). Weiterhin unterscheidet der Server nicht zwischen den Aufrufoperationen. Geschieht die Client-Server-Kommunikation über HTTP,

führt der Client beispielsweise nur POST-Anfragen an eine bestimmte URI durch. Die Semantik der Anfrage ist dabei ausschließlich innerhalb des Bodies definiert.

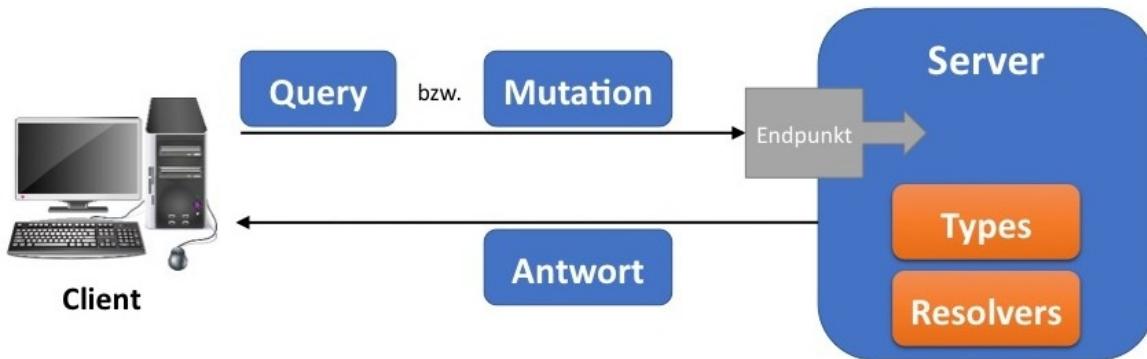


Abbildung 11: Schaubild GraphQL

Anfragen, die dem Client lediglich Daten übermitteln sollen, werden innerhalb einer sogenannten Query formuliert (analog zu GET-Anfrage). Ist eine Manipulation der serverseitigen Daten gewollt, muss eine Mutation übermittelt werden (analog zur POST-, PUT- und DELETE-Anfrage). Zur Verarbeitung von Queries und Mutations besitzt der Server Types und Resolvers. (vgl. GraphQL 2017)

Query

Eine Query veranlasst eine Datenübermittlung vom Server zum Client. Der Client bestimmt dabei dynamisch, welche Informationen er erhalten möchte. Er kann somit nicht nur das Objekt eingrenzen, das übermittelt werden soll, sondern zusätzlich die Datenfelder selektieren.

Besitzt ein Objekt Auto beispielsweise die Attribute Hersteller, Modell, Leistung, Hubraum, Sitzanzahl und Preis, kann der Client explizit mitteilen, welche Datenfelder er tatsächlich benötigt (siehe Code-Beispiel 5). Eine Query hat dementsprechend immer dieselbe Form, wie die zugehörige Antwort. Weiterhin ist es möglich, eine Query beliebig tief zu verschachteln. Ist das Datenfeld eines Objekts wiederum ein Objekt, können auch hier Datenfelder selektiert werden.

```

# Query an den Server
query AllgemeineInformationenZuAuto455{
  auto(id: "455") {
    hersteller{
      name
      standort
    }
    modell
    leistung
    hubraum
  }
}

# Antwort vom Server:
{
  "data": {
    "auto": {
      "hersteller": {
        "name": "vw",
        "standort": "wolfsburg"
      },
      "modell": "golf",
      "leistung": "150",
      "hubraum": "2"
    }
  }
}

```

Code-Beispiel 5: Statische GraphQL-Query mit passender Antwort

Variablen

Der bisherige Ansatz hat den Nachteil, dass Parameter fest mit der Query verknüpft sind. Sollte im vorherigen Beispiel ein Auto mit der Identifikationsnummer 499 abgefragt werden, ist der komplette Query-String zu verändern bzw. eine komplett neue Query zu spezifizieren. Um Queries generisch nutzen zu können, dienen Variablen (siehe Code-Beispiel 6). Innerhalb der Anfrage werden alle dynamischen Parameter durch Platzhalter ersetzt. Abhängig vom eingesetzten Protokoll erfolgt die Auflösung der Variablen in einem separaten Wörterbuch. Eine Anfrage muss somit lediglich einmalig erzeugt werden, während das Wörterbuch dynamisch anzupassen ist.

```
# Query an den Server
query HerstellerinformationenZuEinemBestimmtenAuto($id: ID){
  auto(id: $id) {
    hersteller
    modell
  }
}

# Wörterbuch an den Server
{
  "id": "499"
}

# Antwort vom Server
{
  "data": {
    "auto": {
      "hersteller": "fiat",
      "modell": "500"
    }
  }
}
```

Code-Beispiel 6: Variable GraphQL-Query in Verbindung mit Wörterbuch und passender Antwort

Direktiven

Um Queries noch generischer formulieren zu können, dienen Direktiven (siehe Code-Beispiel 7). Jedes angefragte Datenfeld kann um ein `@include(if: Boolean)` bzw. `@skip(if: Boolean)` ergänzt werden. Soll ein Datenfeld bei einer bestimmten Bedingung beispielsweise nicht angefragt werden, ist ein `@skip(if: $Bedingungsvariable)` anzuhängen. Dieselbe Query kann somit in Abhängigkeit des Wörterbuchs für verschiedene Zwecke genutzt werden.

```

# Query an den Server
query AllgemeineInformationenZuEinemBestimmtenAuto($id: ID, $preisangabe: Boolean){
  auto(id: $id) {
    hersteller
    modell
    preis @include(if: $preisangabe)
  }
}

# Wörterbuch an den Server
{
  "id": "1"
  "preisangabe": true
}

# Antwort vom Server
{
  "data": {
    "auto": {
      "hersteller": "mercedes",
      "modell": "a-klasse",
      "preis": 20000
    }
  }
}

```

Code-Beispiel 7: GraphQL-Query mit Variablen und Direktive

Mutation

Das Pendant zu der Query bildet die Mutation. Immer wenn Daten auf Seiten des Servers generiert, aktualisiert oder gelöscht werden sollen, ist diese Art der Anfrage zu übermitteln. Der Aufbau einer Mutation ähnelt dabei dem einer Query. Es kann ebenfalls eine beliebig tiefe Verschachtelung der Anfrage erfolgen. Auch Variablen und Direktiven sind nutzbar. Lediglich das Schlüsselwort *query* ist in *mutation* zu ändern.

Types

Types liegen im Server vor und bilden die Kommunikationsgrundlage zwischen Server und Client. Hierunter sind die logischen Datenmodelle zu verstehen, auf denen Queries und Mutations arbeiten. Jeder sogenannte GraphQL Object Type beinhaltet eine Menge von Attributen (siehe Code-Beispiel 8). Dies können zum einen Skalare sein, die keine weitere Auflösung erlauben. Von GraphQL bereitgestellte Skalare sind Int, Float, String, Boolean und ID. Auch GraphQL Object Types können als Attribute vorliegen, wenn sie an anderer

Stelle definiert werden. Weiterhin erlaubt GraphQL Listentypen sowie die Deklaration eigener Aufzählungen. Spezielle Typen sind hingegen der Query bzw. Mutation Type. Mithilfe dieser kann spezifiziert werden, welche Objekte seitens des Clients abgerufen bzw. manipuliert werden können. Sie bilden damit den Einstiegspunkt für jede passende Anfrage.

```
# Serverseitiges Datenschema

# Einstiegspunkte für Queries
type Query {
    # Client kann Auto-Objekt anhand von ID abrufen
    auto(id: ID!): Auto
}

type Mutation {
    ...
}

type Auto {
    # Hersteller darf nicht null sein
    Hersteller: Hersteller!
    Modell: String
    Hubraum: Int
    Antrieb: AntriebEnum
    # Liste von Vorbesitzern
    Vorbesitzer: [String]
}

type Hersteller {
    Name: String
    Standort: String
}

enum AntriebEnum {
    Allrad
    Hinterrad
    Vorderrad
}
```

Code-Beispiel 8: Serverseitige Definition von Object Types

Resolvers

Types definieren den Aufbau von einzelnen Datentypen. Durch Kenntnis dieser Struktur kann der Server entscheiden, ob eingehende Anfragen valide sind oder das Schema verletzen. Um Antworten zu generieren, sind Types jedoch nicht ausreichend. Es muss zwangsläufig ein Bezug zwischen der Struktur der Daten und ihrer tatsächlichen Lage hergestellt werden. Hierzu dienen sogenannte Resolvers (siehe Code-Beispiel 9). Ein

Resolver ist eine Funktion (in diversen Sprachen implementierbar), die der Server aufruft, wenn ein spezifisches Datenfeld zurückgeliefert werden muss. Der Inhalt der Funktion wird durch die Geschäftslogik des Serverdienstes definiert und ist demnach eigenständig zu implementieren. Typischerweise wird bei eingehender Anfrage der Root Resolver aufgerufen. Dieser liefert das Ausgangsobjekt (im aktuellen Beispiel ein Auto) zurück. Abhängig von den angefragten Datenfeldern werden auf diesem Objekt weitere Resolver aufgerufen, die zusätzliche Daten beschaffen. Die ermittelten Informationen werden nach Abschluss der Routine geeignet zusammengefügt und an den Client übermittelt.

```
# Resolver-Beispiel-Implementierung in JavaScript

# Root Resolver
Query: {
  auto(obj, args, context) {
    return context.meineDatenbank.ladeAutoAnhandID(args.id).then(
      data => new Auto(data)
    )
  }
}

# Aufgerufen, wenn Auto-Objekt bereits vorhanden
Auto: {
  hersteller(obj, args, context) {
    return context.meineDatenbank.ladeHerstellerAnhandID(obj.herstellerID).then(
      data => new Hersteller(data)
    )
  }

  modell(obj, args, context) {
    return obj.modell
  }

  ...
}

...
}
```

Code-Beispiel 9: Serverseitige Definition von Resolvers

Best Practices

GraphQL ist lediglich eine Abfragesprache und gibt keine Implementierungs- bzw. Nutzungsrichtlinien vor. Dennoch existiert nach (GraphQL 2017) eine Reihe von Best Practices, die im Folgenden skizziert werden.

- **HTTP:** GraphQL-Anfragen und -Antworten können theoretisch über jedes

Netzwerkprotokoll übermittelt werden. Aufgrund seiner Verbreitung ist jedoch das Hypertext Transfer Protocol zu bevorzugen. Der Dienst wartet dabei typischerweise unter dem Pfad `/graphql` auf Anfragen seitens des Clients. Queries und Mutations können weiterhin sowohl im Body eines POST-Requests als auch in der Query eines GET-Requests übergeben werden. Unabhängig davon, ob eine Query oder eine Mutation versendet wurde, antwortet der Server idealerweise mit dem angefragten bzw. dem manipulierten Datenobjekt. Auch Fehlermeldungen werden im Body der Antwort übergeben.

- **JSON:** Vom Server übermittelte Daten befinden sich zwecks Lesbarkeit stets in der JavaScript Object Notation.
- **Keine Versionierung:** Der Client spezifiziert bei jeder Anfrage, auf welche Datenfelder er sich bezieht. Dies macht eine Versionierung des GraphQL-Dienstes überflüssig.
- **Caching:** Im Gegensatz zu REST ist Caching bei GraphQL-Diensten nicht über die URI einer Anfrage möglich. Serverseitiges Caching kann dennoch betrieben werden, wenn jedes Objekt durch eine global eindeutige ID identifiziert wird. Häufige Datenbankzugriffe sind somit vermeidbar, was in einer Performancesteigerung des Dienstes resultiert.

Zusammenfassung

Die Grundprinzipien von GraphQL werden abschließend nach (Facebook 2015) zusammengefasst.

- **Hierarchisch:** Anfragen sowie Datenmodelle sind hierarchisch aufgebaut. Ein Kernelement wird dabei immer weiter in seine Grundbestandteile zerlegt, bis ausschließlich Skalare vorliegen. Dies unterstützt sowohl Entwickler als auch Anwender bei ihrer Arbeit.
- **Produktbezogen:** Im Mittelpunkt von GraphQL stehen die Daten bzw. deren Benutzer.
- **Clientspezifische Anfragen:** Der Client bestimmt dynamisch, welche Daten er benötigt bzw. welche Daten für ihn gegenstandslos sind. Der Server richtet sich vollkommen nach dem Client.
- **Rückwärtskompatibilität:** Jede Clientversion kommuniziert mit demselben Endpunkt und beschafft sich ihre benötigten Daten.
- **Beliebige Datenbeschaffung:** Der Server bestimmt für jedes einzelne Datenfeld seine eigene Beschaffungsroutine, ohne dabei von bestimmten Technologien abhängig zu sein.
- **Applikationsschicht:** GraphQL ist in der Applikationsschicht angesiedelt und muss sich daher nicht um den Transport von Daten kümmern, sondern lediglich um deren Verarbeitung.

- **Stark typisiert:** Queries und Mutations können vom Server effizient hinsichtlich Semantik und Syntax validiert werden.
- **Selbstbeschreibend:** GraphQL-Anfragen und -Datenmodelle werden in einem eigenen Format beschrieben. Hierdurch erhöht sich die Unabhängig zu anderen Systemen und Programmiersprachen.

Fazit

Ein direkter Vergleich der vorgestellten Kommunikationsmethoden ist aufgrund ihrer Heterogenität nur schwer möglich. Es wurden eine Basistechnik, ein Netzwerkprotokoll, ein Architekturstil sowie eine Abfragesprache vorgestellt, welche zwar im Allgemeinen dem Informationstausch dienen, diesen jedoch auf verschiedenen Ebenen unterstützen. Welche Methode Anwendung finden sollte, hängt immer von der jeweiligen Problemstellung ab und ist nicht generell vorherzusagen. Die nachfolgende Gegenüberstellung soll die jeweiligen Anwendungsgebiete trotzdem grob verdeutlichen und somit Klarheit bezüglich der Verwendung schaffen (siehe Tab. 5).

Kommunikationsmethode / Eignung:	Geeignet für:	Ungeeignet für:
RPC	Auslagerung von Methodenlogik	Datenlastige Nachrichtenaustausche
WebSocket	Echtzeitanwendungen	Statische bzw. unkritische Daten
REST	Öffentliche Dienste mit standardisierter Schnittstelle, Austausch von einheitlichen Datenobjekten	Methodenaufrufe, kontinuierlich angepasste Datentypen
GraphQL	Austausch von kontinuierlich angepassten Datentypen / -objekten	Methodenaufrufe

Tabelle 5: Gegenüberstellung Kommunikationsmethoden

Literatur- und Quellenverzeichnis

- Alexander Schill und Thomas Springer. Verteilte Systeme: Grundlagen und Basistechnologien. Springer Vieweg, Berlin; Heidelberg, 2. Auflage, 2012.
- Cesare Pautasso, Erik Wilde und Rose Alarcon. REST: Advanced Research Topics and Practical Applications. Springer, New York; Heidelberg; Dordrecht; London, 2014.
- Erich Reich. <https://blog.qmo.io/ultimate-guide-to-api-design/>, 2017. Aufgerufen: 22.06.2017
- Facebook. <https://facebook.github.io/react/blog/2015/05/01/graphql-introduction.html>, 2015. Aufgerufen: 28.06.2017
- Google. <http://www.grpc.io/>, 2017. Aufgerufen 29.06.2017
- GraphQL. <http://graphql.org>, 2017. Aufgerufen: 28.06.2017
- Guy Levin. <https://dzone.com/articles/top-5-rest-api-security-guidelines>, DZone, 2016. Aufgerufen: 20.06.2017
- Martin Fowler. <https://martinfowler.com/articles/richardsonMaturityModel.html>, 2010. Aufgerufen: 20.06.2017
- OpenAPI Specification. <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md>, 2017. Aufgerufen: 22.06.2017
- Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine, 2000.
- Stefan Tilkov, Martin Eigenbrodt, Silvia Schreier und Oliver Wolf. REST und HTTP. Entwicklung und Integration nach dem Architekturstil des Web. dpunkt.verlag, Heidelberg, 3., aktualisierte und erweiterte Auflage, 2015.
- Swagger. <https://swagger.io>, 2017. Aufgerufen: 22.06.2017
- Tim Berners-Lee, Roy Thomas Fielding und Larry Masinter. RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. 2005.
- Vinay Sahni. <http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>, 2015. Aufgerufen: 22.06.2017
- Fette, I. & Melnikov, A. <https://tools.ietf.org/html/rfc6455>

Backend Entwicklung

Autoren: Niklas Harting

Node.js

Autor: Niklas Harting

Grundlagen

Was ist Node.js?

Node.js ist eine Open-Source JavaScript Laufzeitumgebung auf Basis von Google Chrome's V8 JavaScript Engine, die in C/C++ entwickelt wurde. Node.js wird hauptsächlich für die serverseitige Ausführung von JavaScript verwendet. Mit Node.js lassen sich sehr performante und skalierbare Anwendungen entwickeln. Vor dem Ausführen von JavaScript wird dieser von Node.js in nativen Maschinencode kompiliert.^{3 4}

Node.js ist aber noch mehr, es enthält auch eine Reihe von Standardbibliotheken (Module) und Funktionen, die das Entwickeln von Serveranwendungen stark vereinfachen. Eines der bekanntesten Standardmodule ist das HTTP Modul, welches einem Entwickler die Möglichkeit gibt, mit sehr wenig Code einen kompletten Webserver zu erstellen.⁴

Module

Im Node.js Umfeld bezeichnen Module Softwarebibliotheken, die eine Anwendung um bereits vorgefertigte Funktionalitäten erweitern können.

Module können einzelne .js-Dateien oder ein komplettes Verzeichnis mit mehreren .js-Dateien sein. Die Möglichkeit auch ein ganzes Verzeichnis als Modul anzubieten ermöglichte es, komplexeren Modulen in logische Moduleinheiten und in verschiedene Dateien zu untergliedern.⁶

Der Node Package Manager

Was ist der Node Package Manager?

Ein wichtiges Tool bei der Entwicklung von Anwendungen mit Node.js ist der "Node Package Manager (npm)". Mit diesem Paketmanager lassen sich Module bequem von der Kommandozeile verwalten und installieren. Den Node Package Manager kann man sich ähnlich wie die Paketmanager unter den verschiedenen Linux Distributionen vorstellen. Im Wesentlichen lassen sich neue Module und deren Abhängigkeiten installieren, aktualisieren und löschen.^{6 8}

Module verwalten

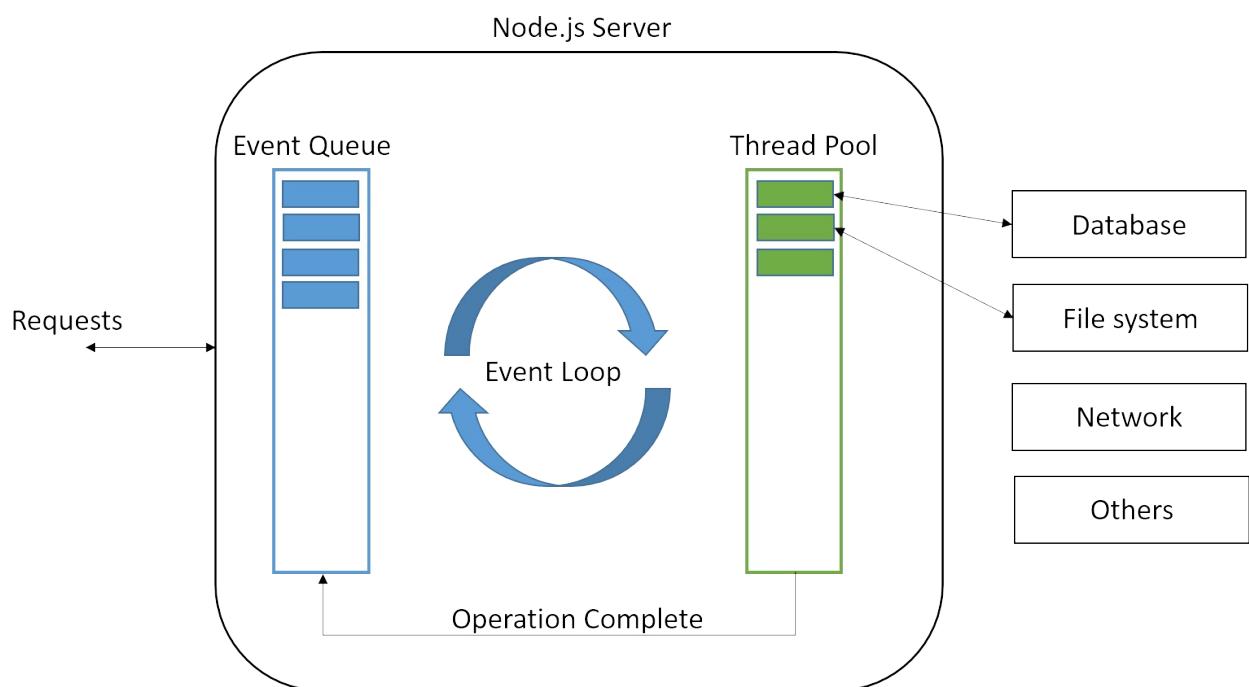
Grundsätzlich lassen sich neue Module global oder lokal installieren, updaten und löschen. Lokal bedeutet in diesem Zusammenhang, dass das Modul nur in der aktuellen Anwendung zur Verfügung steht, während eine globale Installation von Modulen diese systemweit zur Verfügung stellt. Im Nachfolgenden wird ein Überblick über die wichtigsten Funktionen des Node Package Manager gegeben [2](#):

```
$ npm install <Modul>
$ npm install -g <Modul>
$ npm update
$ npm update -g
$ npm uninstall <Modul>
$ npm uninstall -g <Modul>
```

Der Parameter "-g" veranlasst npm, die Operation auf den globalen Kontext anzuwenden. Ist kein Parameter angegeben, geht npm vom lokalen Kontext aus. Wichtig ist auch, dass bei der Verwendung eines dieser Kommandos im lokalen Kontext das aktuelle Arbeitsverzeichnis dem Root-Verzeichnis der betroffenen Anwendung entsprechen muss. [8](#)

Node.js Event-Loop

Node.js Anwendungen werden in nur einem Thread ausgeführt (Code des Programmierers). Alle anderen Operationen laufen parallel und sollte also eine Operation diesen "Anwendungs"- Thread blockieren, wird diese in einen separaten Thread aus einem Pool ausgelagert. Sobald ein Event auftritt (also z.B. Daten wurden aus einer Datei in einen RAM-Puffer geschrieben), wird dieses in die Event Queue eingereiht. Der Event-Loop iteriert nun über alle Einträge in der Event-Queue und ruft die passende Callback-Funktion zu diesem Event auf. [5 6](#)



Node.js Event-Loop 5

Bedingt durch das Modell der eventgesteuerten Programmierung, ist es im Allgemeinen nicht notwendig, sich innerhalb einer Anwendung Gedanken über die Verteilung von Aufgaben in verschiedene Threads zum Zwecke der Laufzeitoptimierung zu machen. Probleme, die nebenläufige Anwendungen für den Entwickler mit sich bringen, verschwinden so gänzlich. Ebenfalls erwächst aus einer Single-Thread Lösung der Vorteil, dass die aufwendige und Ressourcen intensive Verwaltung von verschiedenen Threads und Prozessen durch das Betriebssystem reduziert werden kann. Dadurch werden eventgesteuerte Anwendungen sehr performant und können hoch skaliert werden. Diesen Vorteil hat man aber nur, wenn das Betriebssystem diese Funktion unterstützt, da der Thread Pool vom Betriebssystem verwaltet wird.

Patterns

Standard Callback Pattern

Das Standard Callback Pattern beschreibt die Verwendung von Callback Funktionen als Funktionsparameter, um einen Asynchronen Programmfluss zu ermöglichen. Dies wird auch als Continuation-Passing Style bezeichnet. [2](#)

Event Emitter Pattern

Eher ungeeignet ist das Standard Callback Pattern, wenn bei einem Funktionsaufruf mehrere Events auftreten können. In einem solchen Fall bietet sich das Event Emitter Pattern an. Das Objekt, in dem Events auftreten können, implementiert hier eine Methode „on“ oder „addListener“, mit der Callback-Funktionen an bestimmte Events registriert werden können. Es sind auch mehrere Registrierungen zum gleichen Event möglich. Sollte nun ein Event auftreten, so werden die registrierten Funktionen zu dem zugehörigen Event aufgerufen. [2](#)

Promise Pattern

Eine weitere Alternative zum Callback Pattern ist das Verwenden von Promises. Ein Promise ist das Ergebnis einer Asynchronen Operation und kann einen von drei verschiedenen Zuständen haben. Diese sind pending (der initiale Zustand), fulfilled (die ausgeführte Operation war erfolgreich) oder rejected (die ausgeführte Operation schlug fehl). [7](#)

Beim Promise Pattern hat die Funktion keinen Callback Parameter, sondern es wird ein Promise Objekt zurückgegeben.

Express.js

Autor: Niklas Harting

Was ist Express.js?

Express.js ist ein Webframework für Node.js. Es vereinfacht die Entwicklung von ganzen Webapplikationen und basiert im Ursprung auf dem Node.js Framework Connect. Aus diesem Grund sind alle Connect-Middlewares auch mit Express.js kompatibel. [1](#) [2](#) Express.js eignet sich ebenfalls für die Entwicklung von REST-Schnittstellen, indem durch verschiedene Routen zu den HTTP-Methoden und den angeforderten Ressourcen unterschiedliche Middlewares angesprochen werden können. [1](#) [2](#)

Was ist eine Middleware?

"Middlewarefunktionen sind Funktionen, die Zugriff auf das Anforderungsobjekt (req), das Antwortobjekt (res) und die nächste Middlewarefunktion im Anforderung/Antwort-Zyklus der Anwendung haben. Die nächste Middlewarefunktion wird im Allgemeinen durch die Variable next bezeichnet.“ [1](#)

Es wird zwischen verschiedenen Middleware-Typen unterschieden:

Middleware auf Anwendungsebene

Middlewares auf Anwendungsebene werden direkt an eine Instanz von Express gebunden. Der nachfolgende Ausschnitt gibt ein Beispiel [1](#):

```
let app = express();
app.get('/', function (req, res, next) {
  res.send('Hello World!');
});
```

Middleware auf Routerebene

Seit Express.js Version 4.x wurde eine Router-Klasse eingeführt. Im Wesentlichen handelt es sich dabei um eine abgespeckte Variante der Express-Klasse, es können auf die gleiche Weise Middlewares an ein Router-Objekt gebunden werden. Das Router-Objekt wird wiederum wie eine Middleware an das Express-Objekt gebunden. Dies birgt den wesentlichen Vorteil, dass so eine zusätzliche Abstraktionsschicht möglich ist. [1](#)

```
let express = require('express');
let router = express.Router();
let app = express();
// routes
router.get('/', function (req, res, next) {
  res.send('Hello World!');
});
// bind the routes to the app
app.use('/route', router);
```

Middleware für die Fehlerbehandlung

Fehlerbehandlungen werden wie andere Middlewares an ein Express oder Router Objekt gebunden. Die Callback-Funktion enthält allerdings einen zusätzlichen Parameter "err". [1](#)

```
app.use(function(err, req, res, next) {  
  console.error ( err ) ;  
  res.status(500).send('Error!');  
});
```

Integrierte Middleware

Seit Version 4.x besteht keine Abhängigkeit zu "Connect" mehr. Alle Middlewarefunktionen, die bislang in Express enthalten waren, wurden nun in separate Module ausgelagert. [1](#)

Middleware anderer Anbieter

Durch zusätzliche Module von Drittanbietern kann die Funktionalität von Express.js erweitert werden. Dies wird hier aber nicht näher erläutert. [1](#)

Beispiel-Applikation

Der folgende Code Ausschnitt zeigt eine komplettes Beispiel für eine simple Express.js Anwendung, welche mit Node.js ausgeführt werden könnte:

```
let express = require('express');
let app = express();
app.post('/', function(req, res) {
    res.send('POST request');
});
app.get('/', function(req, res) {
    res.send('GET request');
});
app.put('/', function(req, res) {
    res.send('PUT request');
});
app.delete('/', function(req, res) {
    res.send('DELETE request');
});
app.listen(3000, function() {
    console.log('Server started at port 3000');
});
```

Authentication

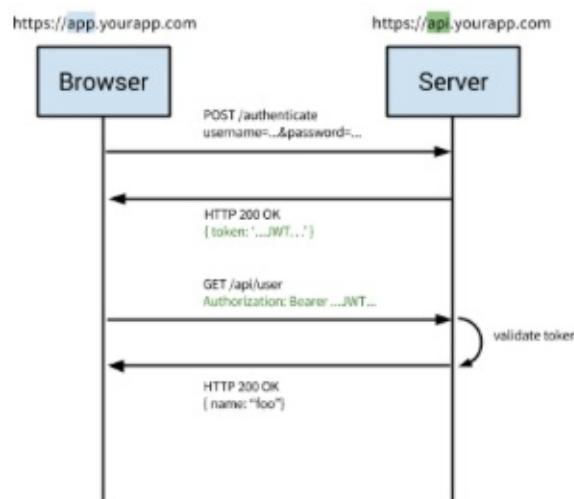
Autor: Niklas Harting

JWT vs. Session Cookie

JSON Web Token (JWT)

Die Token-Based Authentication ist **Stateless**. Das bedeutet, dass auf dem Server keine Information gespeichert werden, sondern nur das JSON Web Token beim Client gespeichert wird. [9](#)

Beim einer Authentifizierung wird auf dem Server ein neues JWT erstellt und zurück an den Client geschickt. Dieser speichert das JWT im Local Storage. Bei jeder weiteren Anfrage sendet der Client nun das JWT mit, welches dann auf dem Server verifiziert wird. [9](#)



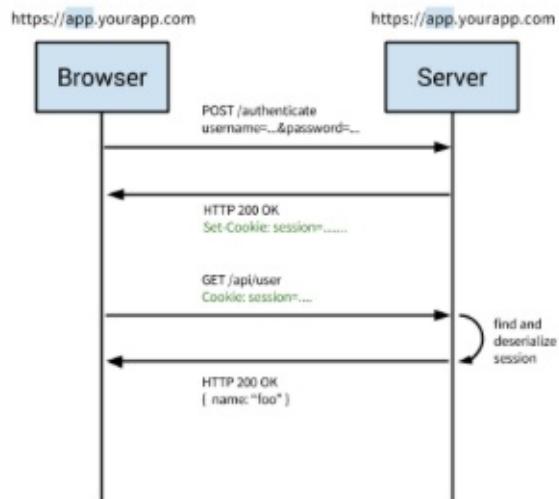
Token-Based Authentication [9](#)

Bei einem Logout des User, wird das JWT dann einfach beim Client gelöscht und es ist keine weitere Interaktion mit dem Server notwendig. [9](#)

Session Cookie

Die Cookie-Based Authentication ist im Gegensatz zur Token-Based Authentication **Stateful**. Es müssen also sowohl auf dem Server sowie beim Client Daten gespeichert werden. Der Server speichert in einer Datenbank alle aktiven Sessions und auf der Client Seite wird ein Cookie mit der Session ID gespeichert. [9](#)

Beim einer Authentifizierung wird dann also auf dem Server eine neue Session ID erstellt, welche in der Datenbank gespeichert wird und in einem Cookie zurück an der Client geschickt wird. Bei jeder weiteren Anfrage zum Server wird nun der Cookie mitgeschickt und es wird überprüft, ob die Session ID aus dem Cookie mit einer Session ID aus der Datenbank übereinstimmt. [9](#)



Cookie-Based Authentication 9

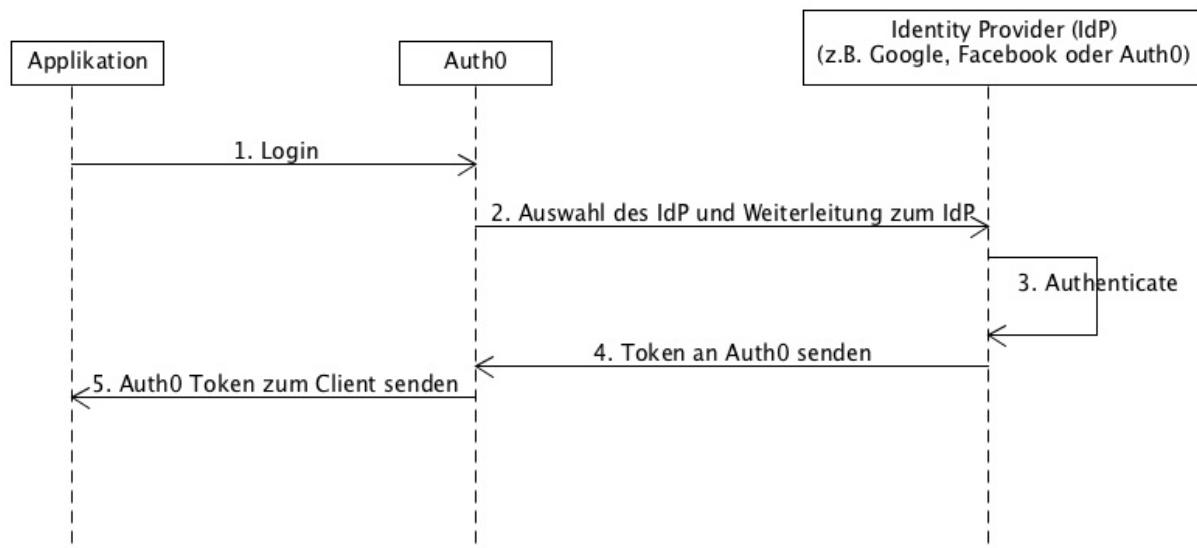
Wenn sich der User nun ausloggt, muss auf dem Client der Cookie und auf dem Server die Session ID aus der Datenbank gelöscht werden. [9](#)

External Identity Providers

Was sind External Identity Providers?

External Identity Providers wie z.B. Auth0 bieten Softwareentwicklern die Möglichkeit, den sehr sensiblen Bereich der User Authentifizierung an einen externen Anbieter auszulagern. Das hat Vorteile für den Softwareentwickler, denn dieser muss sich keine Gedanken über eine sichere Verwaltung der Passwörter oder eine komplexe Authentifizierungsfunktion machen. Aber nicht nur der Softwareentwickler hat Vorteile, denn auch der Anwender hat dadurch die Möglichkeit, sich mit einem bereits bestehenden Konto von z.B. Google oder Facebook anzumelden.

Funktionsweise Auth0



Auth0 Authentication Flow

Quellen

- [1] <https://expressjs.com/de/> besucht am 14.06.2017.
- [2] P. Teixeira, Professional Node.js. John Wiley & Sons, Inc., 2013.
- [3] <https://nodejs.org/> besucht am 13.06.2016.
- [4] "Was ist node.js," <http://nodecode.de/was-ist-nodejs> besucht am 13.06.2016.
- [5] A. R. Olakara, "Understanding the node.js event loop,"
<http://abdelraoof.com/blog/2015/10/28/understanding-nodejs-event-loop> besucht am 27.06.2017.
- [6] S. Springer, Node.js: Das umfassende Handbuch. Serverseitige Webapplikationen mit JavaScript entwickeln. Rheinwerk Computing, 2016.
- [7] <https://www.promisejs.org> besucht am 27.06.2017.
- [8] <https://www.npmjs.com/> besucht am 27.06.2017.
- [9] A. Kukic, "Cookies vs Tokens: The Definitive Guide," <https://auth0.com/blog/cookies-vs-tokens-definitive-guide/> besucht am 30.06.2017.

Verwendete Programme zum Erstellen der Abbildungen:

- UMLet: <http://www.umlet.com/>

JavaScript Testing

Autor: Malte Berg

1 Einführung

Aufgrund der zunehmenden Komplexität von Software steigt die Notwendigkeit von Tests. Werden während der Entwicklung oder Wartung Veränderungen an dem Source Code vorgenommen, kann dies andere Funktionen unbrauchbar machen und Fehlfunktionen hervorrufen. Mithilfe von ausdruckstarken Unit Tests können solche Fehler schnell diagnostiziert und behoben werden, was unter anderem sehr viel Zeit und Geld spart. [5 18](#)

Das automatisierte Testen hilft ebenfalls dabei, dass der Kunde ein qualitativ hochwertiges Produkt erhält, welches die Zufriedenheit steigert und zukünftige Wartungsaufwendungen einschränkt. [18](#) Tests können ebenfalls als Entwicklungsmethodik verwendet werden. Das Test Driven Development (TDD) befasst sich mit dem Schreiben von Tests und der darauffolgenden Implementierung. Häufig werden Tests erst nach dem Implementierungsprozess geschrieben, wodurch unter Umständen unzureichend viele Tests entwickelt werden. [1](#)

2 Testvarianten

Grundsätzlich sollte in jeder Projektphase ausgibig getestet werden. Dennoch sind einige Testvarianten in bestimmten Phasen sinnvoller. Unit Tests sind eine Unterstützung für Entwickler und werden einfach und performant entworfen, um während des Entwicklungsprozesses Feedback zu geben. Integrationstests können Anwendung finden, wenn bestimmte Interaktionen zwischen Modulen geprüft werden müssen, wie beispielsweise die Kommunikation zwischen Applikation und Datenbank. Um dem Endkunden eine funktionierende Software auszuliefern, hilft der funktionale Test dabei, das Zusammenspiel der ganzen Applikation über die User Oberfläche zu testen. [18](#)

2.1 Unit Test

Beim Unit Test werden isolierte, meist einfache Funktionen auf korrektes Verhalten überprüft. Diese Tests bieten dem Entwickler eine sehr gute Unterstützung, besonders wenn es Echtzeitfeedback liefert. So kann überwacht werden, wenn Änderungen implementiert wurden, ob diese an anderen Stellen des Programmcodes Fehlverhalten auslösen. Diese Tests sollten so einfach wie möglich, performant und ausdrucksstark sein. Unit Tests helfen ebenfalls bei der Verbesserung der Code Qualität. So können untestbare Komponenten häufig auf eine schlechte Qualität hinweisen. [18](#) [20](#)

2.2 Integrations Test

Hierbei werden die Interaktionen zwischen mehreren Modulen getestet. So können Datenbankzugriffe, Netzwerk Interaktionen, Logger und vieles mehr, was mit einer zu testenden Funktion verknüpft ist, überprüft werden. Diese Tests können auch mit Unit Tests ausgeführt werden, jedoch ist der Unterschied, dass das Modul mit dem interagiert wird, beispielsweise durch ein Stub ersetzt wird. Dies hat bei Unit Tests den Vorteil, dass die Performance nicht unter dem Aufruf anderer Module leidet. [18](#)

2.3 Funktionaler Test

Bei funktionalen Tests handelt es sich um das Simulieren von echten Nutzerhandlungen und das Überprüfen der Rückgabe. So kann die Benutzung der Oberfläche, wie beispielsweise das Klicken von Buttons oder das Ausfüllen von Eingabefeldern in simulierten Browsern deklariert und ausgeführt werden. Die Ergebnisse werden dann auf bestimmte Annahmen

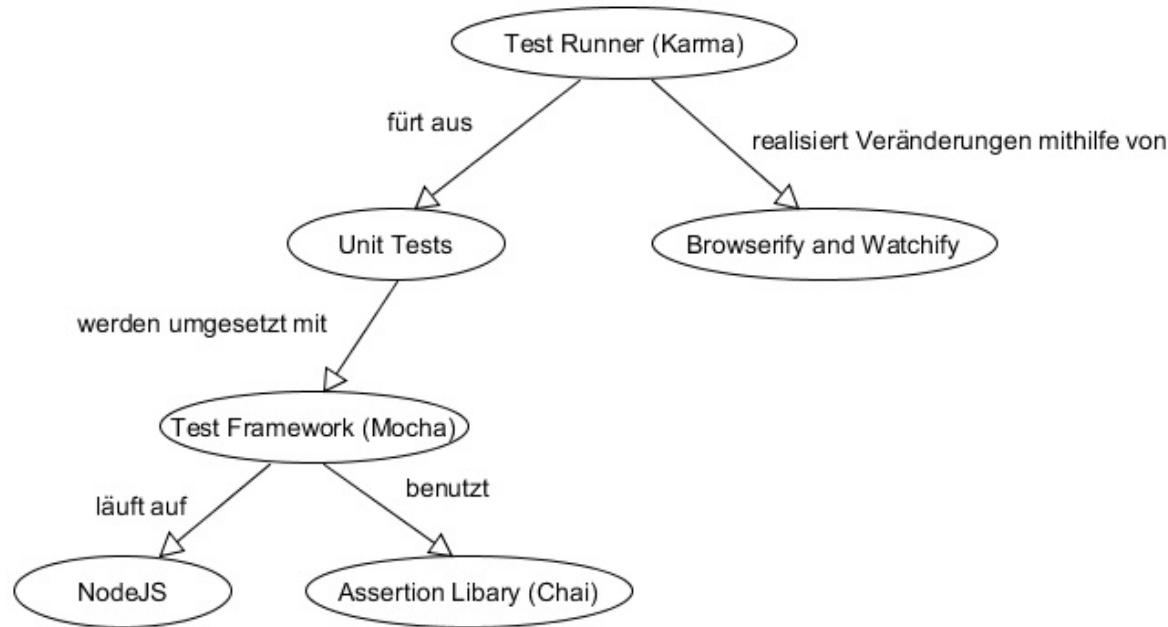
überprüft. Diese Tests werden auch End-To-End Tests genannt, bei denen alle Bereiche der Applikation angesprochen und überprüft werden. Funktionale Tests sind sehr komplex und benötigen viel mehr Zeit als die anderen Testarten. Daher sollten diese Tests nicht zu präzise auf alle Funktionalitäten eingehen, sondern eher auf die wichtigsten und häufigsten Endnutzerhandlungen. [18](#) [20](#)

2.4 Smoke Test

Hierbei wird überprüft, ob die Applikation auch in der Produktivumgebung fehlerfrei funktioniert. Dazu werden erneut die funktionalen Tests in der Produktivumgebung ausgeführt. [18](#)

3 Aufbau einer Testumgebung

Das folgende Diagramm soll bei dem Verständnis helfen, wie eine Testumgebung aufgebaut ist und die einzelnen Frameworks und Libraries miteinander interagieren.



Auf die einzelnen Punkte des Diagramms wird im späteren Verlauf eingegangen. Die in Klammern aufgeführten Frameworks und Libraries geben ein Beispiel zu dem Oberpunkt an. Es gibt wesentlich mehr Möglichkeiten diese Struktur mit anderen Technologien aufzubauen.

3.1 Test Runner

Test Runner erfüllen das Ziel die entwickelten Tests automatisiert auszuführen und eine gut formatierte und ausdrucksstarke Ausgabe der Testergebnisse anzuzeigen. [10](#)

3.1.1 Karma

Karma ist ein Test Runner und wird verwendet um die entwickelten JavaScript Tests automatisiert auszuführen zu lassen. Es wurde zuallererst unter dem Namen Testacular veröffentlicht. Google entwarf es bei der Entwicklung von AngularJS und stellte es kostenlos, quelloffen und öffentlich zur Verfügung. Ziel war es, eine effiziente und einfache Testumgebung zu entwerfen. Karma basiert auf Node.js und Socket.io. Es zeichnet sich besonders durch seine Einfachheit und gute Performance aus. Durch das automatisierte

Ausführen von Tests können Fehler während der Entwicklungsphase vermieden werden. Ebenfalls kann mittels Karmas Unterstützung sehr gut Test Driven Development (TDD) umgesetzt werden. [1](#)

Karma besitzt eine Konfigurationsdatei, in der verschiedene Herangehensweisen deklariert werden können. So kann durch das `autowatch` Attribut mitgeteilt werden, dass die Tests immer dann ausgeführt werden sollen, sobald sich eine Datei verändert. Es können auch Einstellungen bezüglich der Browserwahl, wie beispielsweise, dass die Tests in Chrome und Firefox laufen sollen, vorgenommen werden. Dazu werden die Source Dateien in ein IFrame geladen und die Tests werden ausgeführt. Die Ergebnisse werden dann zurück an den Server versandt. [4](#)

Es ist ebenfalls möglich mit Karma End-to-end Tests durchzuführen, jedoch ist Karmas Design auf low level unit testing ausgelegt. Für diesen Anwendungszweck sollte vorzugsweise auf high level unit testing zurückgegriffen werden, auf welche im späteren Verlauf eingegangen wird. Karma lässt sich zudem sehr einfach mit einem Continuous Integration System, wie beispielsweise Jenkins und mit IDE's wie Webstrom oder dem Browser Google Chrome verknüpfen. [2](#) [3](#)

3.2 Browserify und Watchify

Browserify hilft dabei die externen Javascript Abhängigkeiten und die selbst entwickelten Module zu bündeln und im Webbrowser bereit zu stellen. Dazu wird für jede Browser Ansicht eine JavaScript Datei erstellt, welche die zu bündelnden Abhängigkeiten inkludiert. In dieser Datei können dann Funktionen aufgerufen werden, welche auf dem Browser ausgeführt werden sollen. Wird eine Abhängigkeit bzw. eine JavaScript Datei verändert oder neu hinzugefügt, muss ein einziger Befehl aufgerufen und die aktuelle Seite im Browser neu geladen werden, um die aktualisierten Module nutzen zu können. Um diesen Schritt noch weiter zu vereinfachen gibt es Watchify. Hiermit werden Veränderungen automatisch erkannt und das JavaScript Bündel automatisch neu erstellt. Ein einfaches Neuladen der Seite reicht hierbei aus. [24](#)

3.3 Test Frameworks

3.3.1 Mocha

Mocha wurde 2011 von Tj Holowaychuk, dem Gründer von ExpressJS und weiteren erfolgreichen NodeJS Projekten, entwickelt und ist eine der meistgenutzten Test Frameworks. Die erste Version wurde 2012 veröffentlicht. Es ist für das Behaviour Driven Development (BDD) geeignet und hat sowohl einen eigenen Test Runner als auch eine

Browser Unterstützung integriert. Das Testen von asynchronen Funktionen wird durch die eigene done() Funktion unterstützt. Mocha bietet die Möglichkeit an vor dem Test Initialisierungsfunktionen und nach dem Test Aufräumfunktionen sowohl synchron als auch asynchron auszuführen. [6](#) [7](#) [12](#) [19](#)

```
var assert = require('chai').assert;
describe("Numbers",function(){
  it('should add two numbers',function(){
    assert.equal(5, 3 + 2, '5 equal 3 + 2');
  })
});
```

Mocha besitzt keine integrierte Assertion Library. In diesem Beispiel wurde Chai gewählt, welches im späteren Verlauf näher erläutert wird. Der Test erhält eine Überschrift, welche in der Test Ausgabe als erstes ersichtlich ist und die Unit Tests somit in der Ausgabe gruppiert. Danach wird die Funktion getestet, welche ebenfalls eine Beschreibung erhält. Über die Assertion Library Chai werden dann zwei Objekte auf Gleichheit geprüft und der Test abgeschlossen. Es können ebenfalls mit beforeEach und afterEach Vorbereitungen bzw. Aufräumarbeiten vor und nach den Tests ausgeführt werden. [6](#)

3.3.2 Tape

Tape's Zielsetzung liegt darin, Tests so einfach und minimal wie möglich umzusetzen. Daher wird zum Ausführen von Tests nur das absolute Minimum an Tools benötigt. Ebenso gibt es keine Konfigurationsdateien. Es soll dem Entwickler die Möglichkeit bieten einfache, verständliche und modulare Tests zu schreiben. Tape besitzt standartmäßig einen Test Runner und eine Assertion Library. [10](#)

```
var test = require('tape');
test('My first tape test', function(assert){
  assert.equal(1, 2, 'Numbers 1 and 2 are the same');
  assert.end();
});
```

Über node kann diese Test Datei einfach ausgeführt werden. Das Beispiel zeigt, dass wieder eine Beschreibung des Tests eingegeben werden kann, welcher in dem Testergebnis verzeichnet wird. Über den Funktionsparameter assert können Objekte getestet werden. In dem Beispiel werden zwei unterschiedliche Objekte auf Gleichheit geprüft. Da dies Fehlschlagen wird, wird in dem Fehlerbericht ebenfalls die Beschreibung des Tests ausgegeben. Durch assert.end() kann ausgesagt werden, dass der Test vorbei ist und keine Überprüfungen mehr stattfinden. Die standartmäßige Assertion Library von Tape bietet einen wesentlich eingeschränkteren Umfang als beispielsweise Chai. Es hilft jedoch dabei, sich

schnell einarbeiten zu können und knappe und einfache Tests zu schreiben. Außerdem sollten die Funktionalitäten ausreichen, um alle möglichen Tests abilden zu können. Tape besitzt zudem die Möglichkeiten mittels Browserify und testling, Tests im Browser auszuführen und Initialisierungs- sowie Aufräummethoden zu verwenden, jedoch müssen diese im Funktionsrumpf des Tests manuell aufgerufen werden. [10](#) [11](#)

3.3.3 Jest

Facebooks JavaScript testing Framework Jest kann mit dem vorher beschriebenen Framework Mocha verglichen werden. Es bietet jedoch viele Möglichkeiten, wie beispielsweise Mocking und Assertion ohne diese vorher installieren zu müssen. Eine einfache Installation von Jest reicht aus und es benötigt nur eine äußerst geringe Konfiguration. Ebenso ermöglicht Jest sehr vereinfacht Snapshot Tests mit dessen Hilfe überprüft werden kann ob sich die UI unerwartet verändert hat. [26](#) [27](#)

3.3.4 Vergleich von Javascript testing Frameworks

X	Mocha	Jest	Tape
Flexible Assertion Library Wahl	Ja	Nein	Nein
Installation	Komplizierter, da keine All-In-One Lösung	Einfach	Einfach
Performance	Schnell	Automocking verlangsamt	Schnell
CI-Unterstützung	Ja	Ja mit Einschränkungen	Ja
asynchrone Tests	Ja	Ja	Ja
snapshot testing	sehr schwer	sehr einfach	mit Snapshotter einfach
Mächtigkeit des Frameworks	mächtig	mächtig	nur das nötigste
Dokumentation	ausführlich	sehr gering	sehr gering

Bei der Wahl eines JavaScript testing Frameworks gibt es einige Punkte zu beachten. Es kommt meist auf den Anwendungsfall an und die Wahl sollte darauf abgestimmt sein. So bleibt dem Entwickler bei Mocha die freie Wahl, Reporting Tools, Test Runner und Assertion Libraries auszuwählen. Bei Jest und Tape sind diese bereits vorhanden. Natürlich können auch hier Assertion Libraries hinzugefügt werden, jedoch eingeschränkter als bei Mocha. Die gewährte Flexibilität erschwert die Installation und Konfiguration. So haben Jest

und Tape All-In-One Lösungen, welche hervorragend für einen schnellen Einsatz sind. Bei Mocha müssen erst die Libraries ausgewählt und hinzugestellt werden. Von der Performance sind alle Test Frameworks schnell, jedoch bietet Jest die Möglichkeit Abhängigkeiten automatisch zu mocken. Dies ist in der neuen Jest Version per default deaktiviert, kann jedoch im aktivierte Zustand zu Verlangsamungen führen. [21](#) [8](#)

Alle testing Frameworks besitzen eine Continuous Integration Unterstützung, jedoch kann es bei Jest beim snapshot testing auf einem CI System zu Fehlern kommen. Das automatische Erzeugen von Snapshots kann möglicherweise fehlschlagen. Es wird hierbei empfohlen die Snapshots beim Test statisch in dem Projektordner zu speichern. [15](#) [22](#) [23](#)

Eine weitere Art des Testens, ist das Vergleichen von Snapshots, welche beispielsweise im JSON Format vorliegen können und somit das Herausfinden von unerwarteten Änderungen auf dem User Interface. Hierfür wird eine Datei, welche die aktuelle Ansicht darstellt, erstellt und beim Durchlaufen des Tests die veränderte Ansicht mit der gespeicherten verglichen. Jedes Framework besitzt die Möglichkeit, solche Tests durchzuführen. Bei Jest ist dies vorab schon vorhanden und sehr einfach umzusetzen. Bei den anderen beiden Frameworks müssen externe Libraries hinzugefügt werden, welche dies ermöglichen. [15](#) [16](#)

Da Tape's Grundprinzip lautet, sich nur auf das Wichtigste, nämlich das Testen, zu konzentrieren, ist der Umfang des Testframeworks wesentlich geringer als bei Mocha und Jest. Tape möchte mit möglichst wenig Einarbeitungszeit und nur den nötigsten Funktionen das Testen ermöglichen. Die Dokumentation ist bei Mocha im Gegensatz zu den anderen beiden Frameworks am ausführlichsten. [6](#)

3.4 Assertion Libaries

Assertion Libaries versuchen die Tests durch eigene Funktionen übersichtlicher, einfacher, aussagekräftiger und lesbarer zu machen. Teilweise können die Fehlermeldungen mit Beschreibungen verknüpft werden, welche vorher definiert wurden um direkt darauf hinzuweisen, an welcher Stelle der Fehler aufgetreten ist. Außerdem werden die Testfälle übersichtlich formatiert ausgegeben. Es werden meist auch zusätzliche Untersuchungsmöglichkeiten zur Verfügung gestellt. [10](#)

3.4.1 Chai

Chai bietet viele nützliche Erweiterungen. So kann mithilfe der Library getestet werden, ob Objekte ungleich null sind, größer oder kleiner sind und vieles mehr. Es besitzt drei verschiedene Assertion Arten. Eines davon ist die Assert Schreibweise. Dieses ähnelt der von nodeJS bereitgestellten Assert Schnittstelle und fügt weitere Testmöglichkeiten und Browserkompatibilität hinzu. [6](#) [9](#)

```
var assert = require('chai').assert;
var foo = 'bar';
assert.typeOf(foo, 'string', 'foo is a string');
assert.equal(foo, 'bar', 'foo equal `bar`');
assert.lengthOf(foo, 3, 'foo`s value has a length of 3');
```

Anhand des Beispiels können Tests eingesehen werden, welche den Typen, die Gleichheit und die Länge von Objekten überprüfen kann. Optional kann als letzter Parameter eine Beschreibung mitgegeben werden, welche im Fehlerfall mit angegeben wird. Die anderen beiden Assertion Arten befassen sich mit dem Behavior Driven Design (BDD). Hierbei werden verkette Konstrukte aufgestellt, um eine Annahme zu überprüfen.⁹

Eines dieser BDD Assertion Art ist Chai's Expect.

```
var expect = require('chai').expect;
var foo = 'bar';
expect(foo).to.be.a('string');
expect(foo, 'description: expect: [foo]').to.equal('bar');
expect(foo).to.have.lengthOf(3);
```

Dieses Beispiel zeigt das Testen mithilfe von Chai's Expect. Es werden Verschachtelungen augebaut wie to.be.a (Typprüfung), to.equal (Vergleich mit anderem Objekt) und to.have.lengthOf (Größenprüfung). Der Anfang einer Verkettung beginnt immer mit der expect-Funktion. Das zu testende Objekt wird der expect-Funktion übergeben und kann optional noch eine Beschreibung als zweiten Parameter erhalten, welche im Fehlerfall zu mehr Verständnis verhelfen kann.⁹

Should besitzt eine starke Ähnlichkeit zu Expect und baut ebenfalls verkettete Konstrukte auf. Jede Assertion benötigt hierbei jedoch am Anfang der Verkettung ein should. Hierbei sollte immer auf Browser Kompatibilität geachtet werden, da einige Fehler verursacht werden können.⁹

```
var should = require('chai').should();
var foo = 'bar';
foo.should.be.a('string');
foo.should.equal('bar');
foo.should.have.lengthOf(3);
```

Das Beispiel sieht dem Expect Beispiel sehr ähnlich, jedoch wird auf die Expect Funktion verzichtet und stattdessen auf das Should Objekt gesetzt.

3.5 Mocking Frameworks

Im Allgemeinen werden Mocking Frameworks verwendet, um die Interaktion mit Abhängigkeiten zu überprüfen. Dabei kann getestet werden, wie sich das externe Modul verhält und was bestimmte vorgegebene Rückgabewerte im weiteren Verlauf auslösen. Ebenso kann verhindert werden, dass die externen Module überhaupt ausgeführt werden und stattdessen ein gemocktes Modul ausführen. Dies ist notwendig, da es Anwendungsfälle gibt, bei denen nicht direkt getestet werden kann oder sollte, wie zum Beispiel Datenbankzugriffe, bei denen eine Testdatenbank mit Testdaten erforderlich ist. [13](#) [14](#)

3.5.1 Sinon.js

Sinon bietet Funktionalitäten an, um diese Anforderungen vereinfacht durchzuführen. Zum Analysieren der Funktionsaufrufe bietet Sinon Spies an. Damit kann überprüft werden, wie oft eine Funktion aufgerufen wurde und welche Parameter übergeben wurden. [13](#) [14](#)

```
it('should call save once', function() {
  var test = sinon.spy(ModuleName, 'testFunction');
  functionWhichIncludesTestFunction();
  sinon.assert.calledOnce(test);
});
```

Anhand des Beispiels kann ein solcher Aufruf dargestellt werden. Der Spy wird auf eine bestimmte Funktion in einem Modul initialisiert. Danach kann eine Funktion aufgerufen werden, welche die Testfunktion beinhaltet. Nun können Aussagen getroffen werden, ob die untersuchte Funktion tatsächlich aufgerufen wird oder nicht.

Spies sind lediglich Beobachter und nehmen keinen Einfluss auf Funktionen. Wird dies benötigt, kann auf Stubs gesetzt werden. Ein Beispiel dazu wäre der Test einer Funktion, welche Daten aus einer Datenbank ausliest. Ist die Datenbank leer oder keine Testdatenbank vorhanden, können solche Tests schlecht ausgeführt werden. Das Einrichten würde nun viel Zeit kosten. Stattdessen kann der Code durch ein Stub so manipuliert werden, dass das Testen auch ohne spezielle Einrichtungen stattfinden kann. [13](#)

```
var stub = sinon.stub(modelService, 'getModel')
  .returns(new Model('Example Model'));
```

Bei diesem Beispiel wird deutlich, dass die Methode des modelService, 'getModel' immer ein neu erzeugtes Model zurückliefert. Diese Methode könnte ohne den Stub versuchen, das Model aus der Datenbank zu laden. Durch diese Deklarierung wird die ursprüngliche Methode überschrieben und Tests könnten mit diesem Model-Objekt weiterarbeiten. [13](#)

Ebenfalls kann mit Stubs geprüft werden, ob im Fehlerfall wirklich bestimmte, im Test vorher festgelegte, Fehlermeldungen auftreten. Es können auch asynchrone Funktionen getestet werden, indem ein erwartetes Callback mitgegeben wird und der Test synchron ausgeführt wird. [14](#)

Sinon bietet dazukommend noch die Möglichkeit Mocks zu erzeugen. Mocks werden dann interessant, wenn mehrere, bestimmte Verhaltensweisen geprüft werden sollen. Darum werden die Annahmen auch vorher definiert und erst dann wird die Funktion ausgeführt. Die verschiedenen Annahmen können so in einem Statement deklariert werden. Mit Stubs können solche Tests ebenfalls ausgeführt werden, jedoch müssten dann auch mehrere Annahmen unabhängig voneinander geschrieben werden, was zu großen Codeblöcken führen kann. Bei Mocks sollte darauf geachtet werden, nicht zu viele Annahmen auf einmal zu überprüfen, da sonst kleine Änderungen im Code, welche keine dramatischen Auswirkungen haben, zu Fehlerfällen führen. [14](#)

3.6 E2E-testing Libraries

Neben Unit Tests ist es ebenfalls sinnvoll GUI-Tests zu entwickeln, um eine hohe Qualität zu erzielen. Diese Tests werden meist mit realen Endgeräten und Browsern automatisiert ausgeführt. Bei Single Page Application kann das Testen zu Problemen führen. Hierbei können Inhalte ohne das Laden einer neuen Seite dargestellt werden, was beim Testen schwer zu realisieren ist, da keine eindeutige Aktion kommt, dass eine Anfrage abgeschlossen wurde. Protractor bietet hier eine Lösung. [17](#)

3.6.1 Protractor

Die von Google für AngularJS entwickelte E2E-testing Library Protractor bietet die Möglichkeit das Frontend zu testen. Es nutzt die WebDriver-API von Selenium und zusätzlich das BDD-Testframework Jasmine. Als BBD-Framework können auch andere Frameworks, wie beispielsweise Mocha oder Cucumber verwendet werden. Protractor kann eigenständig identifizieren, wenn eine Seite komplett fertig geladen ist und realisiert automatisch Veränderungen. Es bietet Möglichkeiten, vereinfacht Elemente der Oberfläche ausfindig zu machen. [17](#)

Nach der Installation und Konfiguration von Protractor wird ein Selenium Server gestartet.

```
exports.config = {
  seleniumAddress: 'http://localhost:4444/test',
  specs: ['*.e2e.js'],
  multiCapabilities: [
    { browserName: 'firefox' },
    { browserName: 'chrome' }
  ]
}
```

Eine solche Konfigurationsdatei kann im Beispiel eingesehen werden. Es wird die Adresse des Testservers eingestellt und die Browser auf denen getestet werden soll. Mit dem "specs"-Attribut werden alle Dateien (hier mit der Endung ".e2e.js") geladen, welche die auszuführenden Tests enthalten. [17](#)

```
describe('Web Example', function() {
  it('should have a page title and submit button', function() {
    browser.get('https://testaddress/');
    expect(browser.getTitle()).toEqual('Test Page');
    expect(element(by.buttonText('Submit')).isDisplayed()).toBeTruthy();
  });
});
```

In diesem Beispiel lässt sich solch eine Testdatei erkennen. Es werden Beschreibungen angegeben, damit die Formatierung und Ausdrucksstärke in dem Testbericht vorhanden ist. Danach wird über die deklarierten Browser eine URL aufgerufen und das Ergebnis überprüft. Browser ist hierbei eine globale Variable, welche eine Webdriver Instanz darstellt. In diesem Test wird erst überprüft, ob der Titel der Seite tatsächlich "Test Page" lautet und ob auf der Oberfläche ein Button angezeigt wird, welcher den Text "Submit" beinhaltet. Mithilfe der globalen Variable "by" können HTML Elemente durch verschiedene Funktionen erleichtert gefunden werden. Durch "element" wird der Treffer zu einem Objekt, welches analysiert werden kann. [17](#)

3.7 RESTful API Test

Mit den in den vorherigen Kapiteln vorgestellten Technologien lassen sich ebenfalls RESTful API Tests durchführen. Es ist sinnvoll für jede API Methode einen Test zu schreiben. Es können ebenfalls schnelle native Tests mit beispielsweise Postman durchgeführt werden, in denen ein simulierter Request abgeschickt wird und das Ergebnis als JSON dargestellt wird, jedoch bieten diese nicht den Umfang, welcher mit Unit Tests möglich ist. So kann beispielsweise bei Negativ und Positiv Tests, auf bestimmtes Verhalten geprüft werden. Ebenfalls ist es sehr aufwendig, bei Veränderungen der Applikation die Requests neu in Postman umzusetzen. [25](#)

```
describe('/GET model', function() {
  it('should return all the models', (done) => {
    chai.request(server)
      .get('/model')
      .end((err, res) => {
        res.should.have.status(200);
        res.body.should.be.a('array');
        done();
      });
  });
});
```

In diesem mit Mocha und Chai erstelltem Testbeispiel wird zuallerst ein Request aufgebaut, indem die Route und der anzusprechende Server mitgeteilt wird. Wenn eine Response ankommt, wird diese analysiert. In diesem Fall wird der Status der Antwort und der Datentyp des Bodys überprüft. Hierbei wird ein Array erwartet, da diese Route eine Liste aller Models zurückliefern sollte. [25](#)

Quellen

- 1 "JavaScript Tests mit Karma schnell an den Start bringen", <https://blog.mayflower.de/4333-Karma-Testrunner-Einfuehrung.html> besucht am 13.06.2017
- 2 <https://karma-runner.github.io/1.0/index.html> besucht am 13.06.2017
- 3 <https://karma-runner.github.io/1.0/intro/faq.html> besucht am 13.06.2017
- 4 <https://karma-runner.github.io/1.0/intro/how-it-works.html> besucht am 13.06.2017
- 5 <https://solidgeargroup.com/mocha-javascript-unit-testing> besucht am 15.06.2017
- 6 <https://raygun.com/blog/javascript-unit-testing-frameworks/> besucht am 15.06.2017
- 7 <http://www.techtalkdc.com/which-javascript-test-library-should-you-use-qunit-vs-jasmine-vs-mocha/> besucht am 16.06.2017
- 8 "TESTING REACT APPLICATIONS WITH KARMA, JEST OR MOCHA"
<http://instea.sk/2016/08/testing-react-applications-with-karma-jest-or-mocha/> besucht am 02.07.2017
- 9 <http://chaijs.com/guide/styles/> besucht am 17.06.2017
- 10 "Unit Testing JavaScript Applications With Tape" <https://medium.com/@andy.neale/unit-testing-javascript-applications-with-tape-1d4f5d5fc825> besucht am 22.06.2017
- 11 "Testing JavaScript Modules with Tape" <https://ponyfoo.com/articles/testing-javascript-modules-with-tape> besucht am 22.06.2017
- 12 "Javascript unit testing tools" <https://mo.github.io/2017/06/05/javascript-unit-testing.html> besucht am 22.06.2017
- 13 "Unit Tests in JavaScript with Sinon" <https://solidgeargroup.com/unit-tests-javascript-sinon> besucht am 22.06.2017
- 14 "Best Practices for Spies, Stubs and Mocks in Sinon.js"
<https://semaphoreci.com/community/tutorials/best-practices-for-spies-stubs-and-mocks-in-sinon-js> besucht am 25.06.2017
- 15 "Snapshot Testing" <http://facebook.github.io/jest/docs/en/snapshot-testing.html> besucht am 24.06.2017
- 16 <https://github.com/cdlewis/snapshotter> besucht am 24.06.2017

17 <https://m.heise.de/developer/artikel/End-to-End-Tests-mit-Protractor-2461535.html?artikelseite=all> besucht am 28.06.2017

18 "JavaScript Testing: Unit vs Functional vs Integration Tests"
<https://www.sitepoint.com/javascript-testing-unit-functional-integration/> besucht am
28.06.2017

19 Teixeira, Pedro (2013), Testing Node.js Web UIs, Packt Publishing

20 "What are Unit Testing, Integration Testing and Functional Testing?"
<https://codeutopia.net/blog/2015/04/11/what-are-unit-testing-integration-testing-and-functional-testing/> besucht am 28.06.2017

21 "An Overview of JavaScript Testing in 2017" <https://medium.com/powtoon-engineering/a-complete-guide-to-testing-javascript-in-2017-a217b4cd5a2a> besucht am 29.06.2017

22 <https://medium.com/javascript-scene/why-i-use-tape-instead-of-mocha-so-should-you-6aa105d8eaf4> besucht am 29.06.2017

23 <https://mochajs.org/> besucht am 29.06.2017

24 <https://spapas.github.io/2015/05/27/using-browserify-watchify/> besucht am 30.06.2017

25 <https://scotch.io/tutorials/test-a-node-restful-api-with-mocha-and-chai> besucht am
02.07.2017

26 <http://jonathancreamer.com/testing-typescript-classes-with-jest-and-jest-mocks/> besucht
am 02.07.2017

27 <https://daveceddia.com/snapshot-testing-react-with-jest/> besucht am 02.07.2017

Design

Autor: Lutz Winkelmann

Einführung

Im Folgenden sollen Designentscheidungen, deren Auswirkungen und die Notwendigkeit dieser beschrieben werden. Der erste Eindruck ist entscheidend und wird durch das Aussehen und die Handhabe einer Anwendung geprägt. Das Konzipieren eines Designs bringt gewisse Herausforderungen mit sich. Einige Fragen die sich jeder Designer stellen sollte wären:

- Auf welchen Endgeräten wird die Anwendung genutzt?
- Mit welchen Browsern muss die Anwendung kompatibel sein? (Webanwendungen)
- Welche Displaygrößen müssen unterstützt werden?
- Sind alle Displays hochauflösend?
- Stellen die Displays eine ausreichende Farbpalette für ein ausgefallenes Design zur Verfügung?
- Müssen technische Voraussetzungen wie der Verbrauch von Datenvolumen, beispielsweise beim Laden von Bildern, betrachtet werden?
- Welche Zielgruppe wird adressiert?
- Wird die Anwendung womöglich von Nutzern mit Einschränkungen genutzt?
- Wenn ja, wie kann man dennoch eine gute Nutzbarkeit gewährleisten?

Die nachfolgenden Kapitel beschäftigen sich mit diesen Fragestellungen und sollen einen groben Überblick über Designentscheidungen und deren Folgen nahe bringen.

Grundlagen

Bei der Erstellung eines Designkonzepts werden oft grundlegende Dinge vergessen, da versucht wird ein für sich persönlich ansprechendes Design zu erstellen. Allerdings ist die eigentliche Zielgruppe meist eine andere. Außerdem werden häufig Gadgets eingebaut, welche zwar einen gewissen Nutzen erfüllen, aber für die Anforderungen der Anwendung nicht notwendig sind.

10 Gesetze der Einfachheit

Bereits 2006 machte sich John Maeda Gedanken über Designentscheidungen. Damit eine Anwendung verständlich ist, muss sie einfach gehalten sein. Das Wort "einfach" wird hierbei durch 10 Gesetze definiert, welche auch heute noch auf die Entwicklung von Designkonzepten zutreffen. Die letzten Gesetze werden hier nicht betrachtet, da sie keinen direkten Einfluss auf das Design einer Anwendung haben, beziehungsweise die vorigen Gesetze zusammenfassen.

1. Reduzieren

Durch das gezielte Weglassen von Komponenten wird die Anwendung übersichtlicher. Es sollten nur Komponenten verwendet werden, welche aufgrund der Anforderungen unumgänglich sind. Natürlich sollten Komponenten, welche die Nutzung der Software vereinfachen, dennoch beibehalten werden. Ein Beispiel wäre die Nutzung eines Datepickers mit Eingabefeld, anstatt eines einfachen Eingabefelds, um Daten abzufragen. (vgl. [1] S.1-11)

2. Organisieren

Durch eine gezielte Anordnung von zusammengehörigen Komponenten erscheint das Gesamtbild übersichtlicher. Gruppierte Elemente sollten auch als solche zu erkennen sein. (vgl. [1] S.11-14)

In einem Eingabeformular werden hierfür alle eine Adresse betreffenden, Eingabefelder beieinander dargestellt.

3. Zeit

Eine Anwendung sollte für den Nutzer zielführend gestaltet sein. Ein einfaches Beispiel ist die Startseite, wie sie Anfang der 2000er Jahre genutzt wurde. Diese beinhaltete oft nur den Namen der Seite und einen Link zur eigentlichen Hauptseite. Unnötige Klicks verbunden mit erneuter Ladezeit fühlen sich für den Nutzer komplex an.

Sind Wartezeiten unumgänglich aufgrund von Hintergrundberechnungen einer Anwendung oder ähnlichem, so sollte die Wartezeit durch eine Fortschrittsleiste oder ähnliches angezeigt oder anders überbrückt werden. (vgl. [1] S.23-33)

4. Lernen

Die Anwendung sollte nach Möglichkeit einfach zu handhaben sein. Dies kann dadurch erzielt werden, dass Komponenten genutzt werden, welche die Nutzer bereits von anderen ähnlichen Anwendungen kennen. Das Rad sollte nicht neu erfunden, sondern wiederverwendet werden, damit der Nutzer keine Zeit verschwendet, in welcher er sich die neuen Technologien aneignen muss. Dieses Argument steht außerdem im Zusammenhang mit dem 3. Gesetz der Einfachheit. Wenn eine neue Technologie verwendet wird, die der Nutzer höchstwahrscheinlich nicht kennt, so sollten Hilfestellungen zur Nutzung gegeben werden. Das Lesen der Anleitung ist meist weniger zeitintensiv als das Lernen durch ausprobieren. (vgl. [1] S.33-45)

5. Unterschiede

Es ist nicht möglich nur einfache Anwendungen zu haben. Es könnte nur der Maßstab für die Einfachheit niedriger gesetzt werden wodurch das, was zuerst simpel erschien, nun komplex ist. Somit ist es nicht falsch auch komplexe Konstrukte zu verwenden, solange die gesamte Anwendung einfach wirkt.

Außerdem sollte versucht werden Unterschiede zu anderen Anwendungen hervorzuheben, um Nutzer von der eigenen Software zu überzeugen. Durch Kontraste lernt der Nutzer manche Dinge zu schätzen, welche er sonst als selbstverständlich ansehen würde. (vgl. [1] S.45-53)

6. Kontext

Nicht alle Komponenten einer Darstellung erhalten gleich viel Aufmerksamkeit vom Nutzer. Dies liegt zum Teil an der Platzierung und die Gestaltung dieser. Zentrierte und gestalterisch auffällige Komponenten erhalten eher Beachtung. Dieser Effekt kann zwar erwünscht sein, jedoch können Komponenten hierdurch auch übersehen werden, was einen negativen Effekt hat. (vgl. [1] S.53-63)

7. Emotionen

Zwar sollte eine Anwendung einfach gestaltet sein, jedoch sollte sie die Gefühle des Nutzers nicht vernachlässigen. Dieser Punkt ist extrem abhängig von der Zielgruppe, welche womöglich ein sehr modernes, einfaches und flaches Design bevorzugt oder aber viele Verzierungen erwartet. (vgl. [1] S.63-69)

8. Vertrauen

Dieser Punkt ist sehr eng mit dem 4. Gesetz verwoben. Sobald ein Nutzer weiß, wie man mit einer Komponente umgeht, vertraut dieser darauf, dass diese ihn immer zu dem gewünschten Ergebnis führt. Der Anwender kann die Software, ohne viel zu überlegen, nutzen. Wird die Komponente jetzt verändert oder ausgetauscht, muss der Nutzer sich den Ablauf neu aneignen. Dies sollte, wenn möglich, umgangen werden. (vgl. [1] S.73-80)

Zielgruppe erkennen

Bereits bevor mit dem Design-Konzept begonnen wird, sollte die Zielgruppe einer Anwendung feststehen. Von der Zielgruppe hängen viele Gestaltungsmerkmale ab. Ältere Personen lernen neue Technologien in der Regel langsamer kennen und weigern sich eher eine neue Technologie zu nutzen. Aus diesem Grund sollte hier auf den Skeumorphismus (siehe [Konzepte und Design-Ansätze](#)) zurückgegriffen werden.

Auch auf Gesetze in bestimmten Ländern oder auf religiöse Beschränkungen der Zielgruppe muss geachtet werden. Des Weiteren ist die Sprache und die Bedeutung von Symbolen in anderen Ländern zu berücksichtigen.

Abgesehen von der Zielgruppe müssen allerdings auch alle weiteren Personen betrachtet werden, welche die Anwendung nutzen könnten. Hier muss die Frage gestellt werden, ob eine Anpassung des Designs aufgrund dieser einfach umzusetzen, beziehungsweise ob sie überhaupt nötig ist, da nur ein sehr kleiner Anteil der Nutzer betroffen wäre.

Barrierefreiheit

Der Zugang zu Informations- und Kommunikationstechnologien gilt als Menschenrecht. Dies ist einer der Gründe, aus welchem eine Anwendung barrierefrei, also auch nutzbar für Personen mit körperlichen oder geistigen Behinderungen, sein sollte. Häufig unterstützen die Ansätze für eine barrierefreie Anwendung auch die Nutzbarkeit durch ältere oder

technologisch eingeschränkte Anwender (niedrige Bandbreite o.Ä.). Somit kann die Zielgruppe schnell erweitert werden. Im Folgenden ist eine Auflistung von Möglichkeiten, eine (Web-)Anwendung barrierefrei zu machen.

Behinderung	Maßnahme
Schlechtes Sehvermögen, niedrige Bandbreite	Bereitstellung eines Alternativtextes (alt Attribut in einem HTML-img Tag) für Bilder, welcher auch von Screenreadern erkannt werden kann.
eingeschränkte Farbwahrnehmung	Anbieten alternativer Styles. Nutzung von kontrastreichen Farben.
eingeschränkte Feinmotorik der Hände	Steuerung einer Anwendung komplett durch die Tastatur ermöglichen (Tastatur kann durch Sprachsteuerung immittiert werden).
eingeschränktes Hörvermögen	Abschriften von Audiodateien zur Verfügung stellen.

(vgl. [2])

Konzepte und Designansätze

Bisher wurde die Designkonzeption nur sehr abstrakt geschildert. Im Folgenden sollen jetzt einige Konzepte und Ansätze erläutert werden, welche diese aufgreifen und ergänzen. Bei der Konzeptionierung eines Designs hat der Entwickler meist schon eine grobe Vorstellung des Aussehens einer Anwendung im Kopf. Nachdem die Zielgruppen analysiert wurden kann diese Vorstellung jetzt überarbeitet werden.

Skeumorphismus

Beim Skeumorphsimus wird versucht, das Aussehen von Dingen zu immitieren. Der Skeumorphismus taucht in der Softwareentwicklung bereits bei den Anfängen der Betriebssysteme mit grafischer Oberfläche auf. Icons wie der Papierkorb oder Ordner sollten hier ein schnelles Verständnis und einen leichten Einstieg in die Computerwelt bieten. Wenn eine Anwendung zur Verwaltung von Notizen geschrieben werden soll, so könnte man diese Notizen beispielsweise wie reale Notizzettel aussehen lassen. Hierbei wird das dritte, vierte, siebte und achte Gesetz der Einfachheit erfüllt. Der Anwender kennt, was er sieht und braucht somit keine Zeit für das Lernen der Handhabung. Es kann auf eine Nutzbarkeit wie bei normalen Notizzetteln vertraut werden und durch das reale Aussehen werden bei den Nutzern positive Emotionen ausgelöst. Aus den genannten Gründen ist der Skeumorphismus besonders bei Zielgruppen angebracht, welche sich nicht schnell an neue Technologien anpassen. In der Regel sind dies ältere Personen, welche technisch nicht sonderlich bewandt sind.

Ein weiterer sinnvoller Anwendungsfall ist gegeben, wenn die Anwendung der Nutzung des realen Gegenstands ähnelt. Ein DJ-Pult kann digital ähnlich dargestellt werden wie ein reales. Außerdem können die Schalter, Regler und Scheiben intuitiv genutzt werden, wie in der folgenden Abbildung dargestellt.

Oft bekommen Anwendungen einen Hintergrund aus der realen Welt, ohne dass dieser einen Zusammenhang zu der Anwendung darstellt. Beispielsweise könnte eine Kalenderanwendung einen Baumrinden- oder Leder-Hintergrund bekommen, um ein edleres Aussehen zu vermitteln. (vgl [3])



Mischpult im Skeumorphismus Design ([4])

Flat Design

Das Flat Design wurde mit dem Kachel-Design des ersten Windows Phones bekannt. Hier wird nicht versucht die Realität zu immitieren, sondern Elemente so einfach wie möglich ohne große Schnörkel darzustellen. Dies bringt die Möglichkeit mit sich, Elemente viel freier zu gestalten und völlig neue Ansätze für bereits vorhandene Dinge zu definieren.

Beispielsweise könnte ein Taschenrechner einen größeren Fokus auf Funktionalität bekommen, anstatt mit dem bekannten Layout nur die einfachsten Rechenfunktionen zur Verfügung zu stellen.

Die dargestellten Elemente sind meist in einer Farbe gestaltete, flache Objekte ohne 3D-Effekte wie einen Schlagschatten. Dies sorgt für eine schnellere Umsetzung der Design-Konzeption, da keine Grafiken pixelgenau platziert werden müssen. Die gesparte Zeit kann dann in die Logik der einzelnen Komponenten investiert werden.

Durch das Flat Design kann das erste, zweite, fünfte und sechste Gesetz der Einfachheit erfüllt werden. Die Darstellung der Elemente wird auf ein Minimum reduziert, wodurch die Anwendungen meist geordneter wirken. Außerdem wird die Anwendung durch einheitlich große Elemente, eine gleiche Farbgestaltung dieser oder eine Gruppierung dieser, organisiert. Zuerst mag das Flat Design aufgrund seiner geringeren Ausprägungen der gestalterischen Merkmale komplexer erscheinen, jedoch wird die Nutzung nach kurzer Einarbeitungszeit intuitiver, da es in vielen verschiedenen Anwendungen ähnlich eingesetzt

wird. Es ist sehr einfach den gewünschten Kontext mit den Elementen zu schaffen, da hierbei großer Wert auf die Farbgestaltung sowie die Positionierung gelegt werden kann. (vgl. [3])

Material Design

Im Jahr 2014 hat Google das Material Design vorgestellt. Auf den ersten Blick ähnelt es stark dem Flat Design aufgrund seiner einfachen Darstellung. Allerdings wird beim Material Design die Dreidimensionalität durch eine z-Koordinate ergänzt. Jede Komponente hat die gleiche Dicke von 1dp (Pixel in Abhängigkeit zur Bildschirmgröße). Es ist möglich, Komponenten übereinander darzustellen, wodurch diese einen Schatten auf die unterliegende Ebene werfen. Des Weiteren können die Komponenten durch dp Angaben skaliert und verschoben werden, was für eine bessere Darstellung auf verschiedenen Bildschirmgrößen sorgt. Der Inhalt von Komponenten kann ebenfalls frei auf diesen verschoben und skaliert werden. Die Form der Komponenten kann sich während der Laufzeit ändern was für neue Möglichkeiten des Designs sorgt. Google entwickelt das Material Design stetig weiter und stellt bereits viele Komponenten zur Verfügung, welche sich einfach in eigene Projekte einbauen und verändern lassen. Besonders hervorzuheben sind die vielseitigen Animationen, welche im Material Design eingesetzt werden können und somit, abgesehen von den im Flat Design besprochenen, besonders das siebte Gesetz der Einfachheit ansprechen. (vgl. [5])

Skeumorphismus vs Flat Design vs Material Design

Welches Design Konzept angewendet wird, ist weitestgehend geschmackssache. Allerdings sollte der Designer bei der Wahl des Konzepts die Zielgruppe beachten. Für Personen, welche sich nicht schnell an neue Technologien gewöhnen, beziehungsweise diese verweigern, ist der Skeumorphismus eine gute Wahl, da in diesem bekannte Dinge der realen Welt eingesetzt werden. Auch wenn die Anwendung lediglich einen realen Gegenstand imitieren soll, ist es sinnvoll, den Skeumorphismus anzuwenden, da der Nutzer bereits mit dem Umgang oder dem Aussehen des Gegenstands vertraut ist.

Steht die Funktionalität oder die Einfachheit einer Anwendung hingegen im Vordergrund, so ist das Flat Design oder das Material Design aufgrund der einfacheren Designerstellung vorteilhaft. Auch abstrakte Dinge können mit diesen Konzepten einfacher dargestellt werden. (vgl. [3])

Das Material Design wirkt zwar durch 3D-Ebenen und Animationen freundlicher als das Flat Design, allerdings stellen diese Eigenschaften auch einen gewissen Overhead dar. Bei Anwendungen, welche häufig genutzt werden, beispielsweise weil der Beruf die Nutzung erfordert, kann dies auf Dauer eher negativ als positiv wirken. Also sollten vor allem die Animationen mit Bedacht eingesetzt werden, da sie gegen das dritte Gesetz der Einfachheit, die Zeit, sprechen.

Dynamische Bildschirmgrößen

Häufig werden Anwendungen für verschiedene Plattformen oder Geräte zur Verfügung gestellt. So ist eine Webanwendung beispielsweise auf einem Desktop-PC, einem Tablet, einem Smartphone oder ähnlichem ansteuerbar. Hierbei ergibt sich das Problem, dass Komponenten der Anwendung entweder mit der Displaygröße skalieren oder austauschbar sein müssen. Für diese Problemstellung gibt es mehrere Ansätze, wovon einige im Folgenden beschrieben werden sollen. Diese arbeiten mit Media Queries, welche die verschiedenen Viewportgrößen des Displays an die Anwendung weitergeben. (vgl. [6])

Adaptive Design

Beim Adaptive Design werden mehrere Layouts erstellt, welche für verschiedene Bildschirmgrößen verwendet werden sollen. Meist wird ein Layout für ungefähr 19 Zoll und größer, 13 Zoll und größer und kleiner als 13 Zoll erstellt. Die Layouts werden in CSS mit dem Zusatz der Bildschirmgrößen formuliert und entsprechend der Viewportgröße des Endgeräts angezeigt. Das Adaptive Design ist also an die Endgeräte ausgerichtet, von welchen die Anwendung genutzt wird. (vgl. [6])

Responsive Design

Beim Responsive Design werden die Komponenten so gestaltet, dass sie sich dynamisch der vorhandenen Bildschirmgröße anpassen. Auch hier gibt es einige Layoutvorgaben, welche bei bestimmten Viewportgrößen angewendet werden. Diese beziehen sich allerdings meist auf eine Umstrukturierung des Layouts. Hierdurch können Elemente versteckt, anders angeordnet oder sichtbar gemacht werden. (vgl. [6])

Eine Navigationsleiste am linken Bildschirmrand wird bei großen Bildschirmen beispielsweise meist mit einem Icon und einem beschreibenden Namen pro Element versehen. Auf mittelgroßen Bildschirmen wird dann nur noch das entsprechende Icon angezeigt. Auf einem Smartphone verbirgt sich die Navigationsleiste dann oft hinter einem sogenannten Burger-Menü, welches ein Button ist, der durch Anklicken die Navigationsleiste erscheinen lässt.

Adaptive Design vs Responsive Design

Das Adaptive Design lässt sich schneller umsetzen, da es nur für eine bestimmte Anzahl an Bildschirmgrößen getestet werden muss. Außerdem ähneln die Resultate meist stärker den Mockups als beim Responsive Design, da hauptsächlich die Größe der Elemente angepasst wird. Beim Responsive Design können verschiedene Viewportgrößen zu unterschiedlichen Layouts führen, was den Nutzer verwirren kann, allerdings ist beim Responsive Design eine gute Darstellung für alle Viewportgrößen bis zu einem gewissen Maximum gegeben. Wenn jemand bei einer adaptiv gestalteten Anwendung auf einem 17 Zoll Monitor eine Anwendung startet, welche für 13 und 19 Zoll optimiert wurde, so gibt es viele Leerräume, welche aufgrund der statischen Darstellung entstehen. Somit kann man sagen, dass der Entwicklungsaufwand eines adaptiven Designs zwar deutlich geringer ist, jedoch das Resultat eines Responsive Designs meist besser aussieht und zukunftssicherer ist, was neue Bildschirmgrößen anbelangt. (vgl. [6])

Pagination vs Infinite Scrolling

Muss in einer Anwendung eine Liste mit vielen Einträgen dargestellt werden, so gibt es hierfür zwei Möglichkeiten mit Vor- und Nachteilen. Die bekannteste Variante ist wohl eine Tabelle, welche eine gewisse Anzahl an Elementen auf einer Seite darstellt. Durch einen Klick auf die nächste Seitenzahl werden neue Elemente geladen und anstatt der bisher angezeigten, dargestellt. Dieses Prinzip ist bei den meisten Onlineshops anzutreffen, da schnell durch die Elemente navigiert werden kann. Wird eine Liste beispielsweise nach dem Preis sortiert und man ist auf der Suche nach etwas aus dem mittleren Preissegment, so muss man nur auf die mittlere Seitenzahl gehen und kann von dort aus weiter navigieren.

Auch beim Infinite Scrolling gibt es häufig die Möglichkeit Suchergebnisse zu sortieren. Hier werden die einzelnen Elemente allerdings nicht auf mehreren Seiten angezeigt, sondern durch das Runterscrollen automatisch nachgeladen. Dieser Prozess wird "Lazy Loading" genannt. Ein Entwickler muss sich hierbei allerdings die Frage stellen, wie viele Elemente pro Ladezyklus geladen werden sollen und wann dieser Zyklus ausgelöst wird. Auf mobilen Endgeräten sollten beispielsweise maximal 15 bis 30 Elemente geladen werden, da das Display kleiner ist und das Datenaufkommen möglichst gering gehalten werden sollte. Teilweise wird das Infinite Scrolling auch mit einem "Elemente Nachladen" Button versehen, wodurch der Ladezyklus manuell ausgeführt werden muss. Dieser Ansatz ähnelt dann bereits wieder der Paginierung, allerdings werden die neuen Elemente zusätzlich, zu den bereits geladenen, angezeigt und nicht stattdessen.

Auch muss für das Infinite Scrolling eine Routine definiert werden, die bei der anwendungsinternen Navigation dafür sorgt, dass nach der Detailansicht eines Elements wieder die richtige Stelle der Liste präsentiert wird, damit der Nutzer die Liste nicht ein

weiteres Mal durchscrollen muss.

Somit kann man sagen, dass das Infinite Scrolling flüssiger abläuft, die Pagination allerdings Vorteile beim Wiederfinden von Elementen hat. (vgl. [7])

CSS-Technologien und Erweiterungen

Mit Cascading Stylesheets kann unter anderem das Aussehen von HTML-Elementen bestimmt werden. Auch die Anordnung von Elementen innerhalb anderer Elemente wird hierdurch gesteuert. CSS ist eine sehr leichtgewichtige Technologie, die durch andere Ansätze erweitert wurde. Dies soll im Folgenden näher beleuchtet werden.

CSS Grid

CSS Grid bietet die Möglichkeit (HTML-)Elemente innerhalb eines anderen (HTML-)Elements in einem zweidimensionalen System anzugeben. Das umschließende Element wird durch das Setzen eines Attributes zum Grid. Hier wird auch die Spalten- und Zeilenanzahl definiert. Die einzelnen Elemente innerhalb des Grids können durch Angabe der Zeile und Spalte innerhalb des CSS positioniert werden. Die einzelnen Zellen, Zeilen und Spalten können ebenfalls durch CSS bearbeitet werden, wodurch viele Möglichkeiten der Darstellung entstehen. Ein Element muss nicht zwingend nur in einer Zelle dargestellt werden, sondern kann durch Angabe von Endzeile und Spalte auch über einen größeren Bereich gehen. Es ist ebenfalls möglich eine Zelle aufzuteilen, indem das beinhaltete Element zu einem Sub-Grid wird.

Die Möglichkeiten der Darstellungsanpassung mit dem CSS Grid sind enorm und können vor allem im Zusammenhang mit Media Queries gut für responsive Anwendungen verwendet werden, da das Grid hierbei komplett umgestaltet werden kann. (vgl. [8])

Flexbox

Die Flexbox stellt im Gegensatz zum CSS Grid einen eindimensionalen Container dar. Es werden hier ebenfalls Elemente innerhalb eines anderen Elements dargestellt. Die Anordnung der inneren Elemente kann vertikal oder horizontal sein, wobei die Reihenfolge der Elemente gespiegelt werden kann. Der große Vorteil der Flexbox ist, dass die Größen der Komponenten sehr viel dynamischer als die des CSS Grids dargestellt werden können. Beispielsweise kann der, durch den Container zur Verfügung gestellte, Platz gleichmäßig unter allen inneren Elementen aufgeteilt oder aber einem einzelnen Element gutgeschrieben werden. Die Anordnung der einzelnen Elemente kann sehr dynamisch gestaltet werden und auch hier ergibt sich die Möglichkeit, durch Media Queries die Reihenfolge der Elemente zu ändern oder andere Anpassungen vorzunehmen. Allerdings ist dies oft nicht nötig, da sich der Inhalt der Flexbox flexibel an die vorgegebene Containergröße anpasst. (vgl. [9])

Erweiterungen

Es gibt viele Erweiterungen von CSS, wobei der angefertigte Code durch einen Compiler in CSS umgewandelt wird. Durch dieses Compiling können Entwickler einiges an Zeit und Code sparen, da viele Grundprinzipien von einfachen Programmiersprachen, wie das Anlegen von Variablen, benutzt werden können. Somit muss der Entwickler die Hauptfarbe einer Anwendung nicht in jeder einzelnen CSS-Klasse aktualisieren, sondern kann die Änderung an einer zentralen Stelle durchführen.

Es gibt sehr viele dieser Tools, welche sich hauptsächlich in der Syntax unterscheiden. Außerdem wird bei der Nutzung mancher Frameworks eine bestimmte CSS Erweiterung vorausgesetzt, wodurch der Entwickler gezwungen ist, die Syntax zu lernen.

SCSS (Sassy CSS) nutzt die gleiche Formatierung wie CSS, wodurch es einfach zu lernen ist. Die bekannten Erweiterungen SASS (Syntactically Awesome Stylesheets) und Less haben hingegen andere Formatierungen. (vgl. [10])

Alle Erweiterungen haben gemeinsam, dass sie die Stylesheets wiederverwendbarer machen. Zum einen können Variablen genutzt werden, welche alle gültigen Attributwerte wie Farben oder Zahlen speichern und verwalten. Außerdem können CSS-Klassen ineinander verschachtelt werden, wodurch die Regeln für die innere Klasse nur auf Elemente angewendet werden, welche in der (HTML-)Struktur innerhalb der äußeren Klasse liegen.

Mixins in den Erweiterungen können mit Funktionen aus Programmiersprachen verglichen werden. Sie können Parametrisiert sein und dementsprechend CSS-Werte einsetzen. Diese Mixins können dann in Klassen inkludiert werden. Auch die Vererbung von Attributen einer Klasse an eine andere ist möglich. (vgl. [11])

Die Nutzung der Erweiterungen erfüllt meist eher die Anforderungen eines Entwicklers, da dieser die gegebenen Unterstützungen von Programmiersprachen kennt. Außerdem ist die Wartbarkeit deutlich besser als bei der Nutzung von reinem CSS.

Quellen

- [1] John Maeda. The Laws of Simplicity. Design, Technology, Business, Life. The MIT Press, 2006.
- [2] W3C. Accessibility. URL:<https://www.w3.org/standards/webdesign/accessibility> (besucht am 27.06.2017).
- [3] Nadja Krakow. Flat Design vs. Skeuomorphismus – eine Frage des Geschmacks?.
1. Mai 2013.
URL:<https://www.mittwald.de/blog/webentwicklung-design/webdesign/flat-design-vs-skeuomorphismus-eine-frage-des-geschmacks> (besucht am 27.06.2017).
- [4] John Biggs. After Skeumorphism. 04. Aug. 2013.
URL:<https://techcrunch.com/2013/08/04/after-skeumorphism/>. (besucht am 27.06.2017).
- [5] Apache. Material design. URL:<https://material.io/guidelines/>. (besucht am 28.06.2017).
- [6] Jonas Hellwig. Adaptive Website vs. Responsive Website. 09. Aug. 2015.
URL:<https://blog.kulturbanause.de/2012/11/adaptive-website-vs-responsive-website/>. (besucht am 28.06.2017).
- [7] Christian Holst. Infinite Scrolling, Pagination Or “Load More” Buttons? Usability Findings In eCommerce. 01. März 2016.
URL:<https://www.smashingmagazine.com/2016/03/pagination-infinite-scrolling-load-more-buttons/>. (besucht am 28.06.2017).
- [8] Chris House. A Complete Guide to Grid. 03. Mai 2017. URL:<https://css-tricks.com/snippets/css/complete-guide-grid/#prop-grid-area>. (besucht am 28.06.2017).
- [9] Chris Coyier. A Complete Guide to Flexbox. 21. März 2017. URL:<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>. (besucht am 28.06.2017).
- [10] Mario Janschitz. Sass vs. Less: So findest du den richtigen Präprozessor für dich. 09. Sep. 2015. URL:<http://t3n.de/news/sass-vs-less-636820/>. (besucht am 28.06.2017).
- [11] SASS. SASS Basics. URL:<http://sass-lang.com/guide>. (besucht am 28.06.2017).

Cloud Security

Projektleiter: Philipp Viertel

Projektteam: Philipp Viertel, Andre Kaleja

Github-Repo: <https://github.com/pviertel/cloudsecurity.git>

Einführung

Autor: Philipp Viertel

Clouds nehmen in der IT-Landschaft einen immer größeren Stellenwert ein. Sie ermöglichen es Speicherplatz, Applikationen, Dienste und Rechenleistung zu verteilen und von überall aus erreichbar zu machen. Dies bedeutet eine komplett neuartige IT-Infrastruktur im Vergleich zur bisherigen.

Diese Verteiltheit und das beständige Verfügbarsein der Dienste, macht sie auch wiederum angreifbar und verwundbar. Die Frage der Sicherheit, also wie man Cloud Lösungen sicherstellt, hängt immer stark davon ab, um was für eine Art Cloud es sich handelt und mit welcher Technologie sie umgesetzt wurde. Gerade das Verständnis, was eine Cloud ist und wie sie aufgebaut ist, ist eine Grundvoraussetzung für eine sicherheitskritische Betrachtung.

Im nachfolgenden Kapitel "Grundlagen" werden einige Definitionen beschrieben. Im Kapitel "Einsatzgebiete" wird dem Leser deutlich, wie weit verbreitet Clouds mittlerweile im alltäglichen Leben geworden sind und wie erfolgreich sie dabei sind. Von Anwendungen für den Endanwender bis hin zu umfangreichen Firmenlösungen für Großkonzerne. Das Kapitel "Cloud Management Platforms" geht auf die technische Seite ein. Welche Lösungen gibt es? Wie sind sie aufgebaut und wie lassen sie sich vergleichen? Ein Fokus bei dieser Betrachtung liegt auf OpenStack. Anschließend im Kapitel "Sicherheit" wird das namensgebende Thema näher betrachtet. Im Schluss wird im Kapitel "Ausblick" ein kleines Fazit gegeben sowie ein Blick in die nähere Zukunft gewagt, in welche sich die Cloud unter Berücksichtigung des Sicherheitsaspekts entwickeln könnte.

Grundlagen

Autor: Philipp Viertel

Um das Verständnis der Cloud Systeme näher zu bringen, müssen einige Definitionen vorab erläutert werden. Hierzu wird auf die Definitionen des NIST (National Institute of Standards and Technology) in [8] zurückgegriffen, welche allgemein anerkannt sind.

Das Cloud Computing Modell stellt einen Netzwerk Zugriff auf einen geteilten Pool von konfigurierbaren Rechenressourcen zur Verfügung. Diese beinhalten z.B. Server, Speicherkapazitäten, Applikationen, Dienste, usw. Dieser Zugriff stellt dabei einige Anforderungen: Er muss ubiquitär sein, zweckmäßig und auf Abruf immer bereit stehen. Ein solches Modell muss schnell verfügbar sein und das mit wenig Interaktion seitens des Anbieters.

Es werden dabei fünf essentielle Eigenschaften definiert:

- On-demand self service: Ein Anwender muss automatisch Rechenkapazitäten oder andere Leistungen von der Cloud beziehen bzw. nutzen können, ohne menschliche Interaktion mit dem Anbieter.
- Broad network access: Der Zugriff auf die Leistungen muss über herkömmliche Netzwerkstandards erfolgen (z.B. über Smartphones, Laptops, Tablets, usw.)
- Resource pooling: Über Methoden der Virtualisierung müssen auch mehrere Anwender gleichzeitig bedient werden. Die Ressourcenzuordnung muss dynamisch geschehen.
- Rapid elasticity: Die Anforderungen an die Skalierbarkeit sind enorm. Kapazitäten müssen schnell und dynamisch an die Benutzeranforderungen angepasst werden.
- Measured service: Ressourcen und ihre Nutzung müssen überwacht, kontrolliert und optimiert werden.

Es gibt drei unterschiedliche Service Modelle:

- Software as a Service (SaaS): Die Anwendung die vom Anbieter bereitgestellt wird, läuft auf der Infrastruktur der Cloud. Sie kann über herkömmliche Netzwerkapplikationen wie Internet Browser or ein anderes Interface genutzt werden. Auf die dahinterliegenden Elemente hat er jedoch keinerlei Einfluss.
- Platform as a Service (PaaS): Der Anwender kann auf die Cloud Infrastruktur selbst erstellte Applikationen ablegen, die mittels Programmiersprachen und/oder Bibliotheken vom Anbieter angeboten und realisiert wurden. Darüber hinaus hat er keinen Einfluss auf die Infrastruktur der Cloud.
- Infrastructure as a Service (IaaS): Der Anwender kann Rechenressourcen nutzen um eigene Software (inklusive Betriebssysteme) auf der Cloud abzulegen und auszuführen.

Darüber hinaus hat er keinen Einfluss auf die Cloud, hat jedoch Zugriff auf Betriebssysteme, Speicher und die abgelegten Anwendungen.

Die Arten von Clouds in ihrem Einsatzgebiet lassen sich in vier Kategorien einteilen:

- Private Cloud: Die Infrastruktur ist für die bestimmte Nutzung einer einzigen Organisation gedacht. Sie kann im Besitz der Firma oder einer dritten Partei sein, die auch die Verwaltung übernimmt.
- Community Cloud: Die Infrastruktur wird für eine bestimmte Nutzung einer Gemeinschaft von Anwendern angeboten. Sie wird von einer oder mehreren Organisatoren der Gemeinschaft verwaltet.
- Public Cloud: Die Infrastruktur ist offen für die Öffentlichkeit. Sie gehört einem Unternehmen, einer Regierungsorganisation, einer akademischen Organisation oder einer Mischung aus diesen.
- Hybrid Cloud: Die Infrastruktur ist eine Mischung aus zwei oder mehreren andren Infrastrukturen (Private, Community, Public). Sie werden zusammengeführt durch entsprechende IT-Lösungen.

Einsatzgebiete

Autor: Andre Kaleja

Im Folgenden werden einige reale Anwendungsbeispiele genannt.

Das Modell SaaS stellt wie der Name schon sagt, Software als Service bereit und kann über den Browser angesteuert werden. Unternehmen müssen selber keine Server und Software Lizenzen mehr beschaffen. Gebühren für die Nutzung fallen nun monatlich oder pro Nutzer an. Microsoft z.B. vertreibt unter dem Namen Microsoft Online Services, Services wie Exchange Online, Share Point Online, Communications Online und LiveMeeting. Noch ein Beispiel wäre die Firma Google mit Google Apps.

Das Utility Computing stellt virtuelle Server und Datenspeicherung auf Wunsch bereit. Es beinhaltet Rechenleistung, Datenhaltung oder Applikationen eines lokalen Rechenzentrums. Über ein Grid können eine Vielzahl von Nutzern auf die Ressourcen zugreifen. Diese Art der Cloud nennt man public cloud. Amazon, Google oder Salesforce.com bieten solche Services an.

Bei Web Services werden verhäuft komplett APIs über das Internet bereitgestellt. Hier kann man z.B. Google Maps erwähnen. Über die Google Maps API kann man auf seiner eigenen Website Karten integrieren.

Das Modell PaaS stellt Applikationen zur Entwicklung bereit. Diese laufen über die Infrastruktur und werden über das Internet an den Endanwender ausgeliefert. Die von Microsoft entwickelte Plattform Azure ist ein gutes Beispiel. Neben Speicher und Rechendiensten liefert Azure eine Ausführungs-Managementumgebung für Anwendungen und Dienste. Entwickler haben hier ein hohes Maß an Flexibilität und können ganze oder auch nur Teile von Anwendungen in den Betrieb von Azure einbringen. Ein weiteres Beispiel ist die Hosting Plattform von Google, Google App Engine. Es können intensiv genutzte Webanwendungen erstellt werden, ohne dabei die Verwaltung einer hochbelastbaren Infrastruktur betreiben zu müssen.

MSP (manages service provider) Ist eine Applikation, die die Verantwortung für die Bereitstellung von Dienstleistungen für seine Kunden übernimmt und verwaltet. Hier geht es weniger um den Gebrauch durch den End Nutzer. Beispiele für solche Services können ein Virus Scanning Service für Emails oder ein Applikations Monitoring sein. Reale Produkte sind z.B. Secure Works, IBM, Verizon und Cloud Anti Spam Services wie Postini von Google.

Eines der bekanntesten Cloud Computing Beispiele ist das Amazon Web Services (AWS). Es ist eine sichere Plattform für Cloud-Services, Rechenleistung, Datenspeicherung und das Bereitstellen von Inhalten und weiteren Funktionen. AWS stellt die folgenden computing Services bereit: Amazon S3, EC2, Virtual Private Cloud, CloudFront, eine virtuelle Datenbank, IAM (Identity and Access Management), CloudWatch, Amazon Direct Connect und viele mehr. Dabei liefert AWS bereits Datensicherheitslösungen auf verschiedenen Systemebenen. Bekannte Dienste wie Dropbox, Netflix, Foursquare oder Reddit greifen auf die Dienste von AWS zu.

Cloud Management Platforms

Autor: Philipp Viertel

Wenn man eine eigene Cloud betreiben bzw. nutzen möchte, kann man auf viele unterschiedliche Anbieter und Lösungen zurückgreifen. Wenn man eine Technologie für die eigenen gewünschten Bedürfnisse aussucht, muss man sich die Frage stellen, für welchen Zweck man sie nutzen möchte. Es gibt eine Reihe von Cloud Management Platforms (CMP), Implementierungen von Cloud-Lösungen, die man für einen bestimmten Anwendungsfall in Betracht ziehen kann. An dieser Stelle sollen einige dieser Lösungen vorgestellt werden. Drei große Projekte, welche näher beschrieben und miteinander verglichen werden, sind OpenStack, CloudStack und OpenNebula. Der Fokus liegt dabei auf OpenStack, da dieses System auch in der praktischen Arbeit verwendet wurde.

OpenStack

OpenStack ist ein open-source Software Projekt zur Realisierung von Cloud Computing Diensten als infrastructure-as-a-service (IaaS). Es ist in Python geschrieben und ursprünglich wurde das Projekt als eine gemeinsame Arbeit von Rackspace Hosting und der NASA ins Leben gerufen. Seit 2016 wird es mittlerweile von der OpenStack Foundation verwaltet. Zur Beschreibung wurde auf die Dokumentation in [3] und [4] zurückgegriffen.

OpenStack ist ein sehr komplexes System. Es enthält eine große Anzahl von unterschiedlichen Komponenten, die das Projekt ausmachen. Diese sind für die Bereitstellung der einzelnen Dienste notwendig und sind selbst sehr umfangreich. Die wichtigsten Komponenten werden an dieser Stelle erläutert, um zu verstehen, was OpenStack im Kern ausmacht und wie es aufgebaut ist. Zu diesen gehören Nova (Compute), Glance (Image Service), Swift (Object Storage), Keystone (identity), Horizon (Dashboard), Cinder (Block Storage) und Neutron (Networking). Die zusätzlichen Komponenten Ceilometer (Telemetry Service), Heat (Orchestration Service), Trove (Database Service) und Sahara (Data Processing Service) werden der Vollständigkeit halber ebenfalls beschrieben.

Die im weiteren Verlauf näher erläuterten Komponenten von OpenStack und deren gegenseitiges Zusammenspiel, lassen sich in Abbildung 1 im Zusammenhang darstellen.

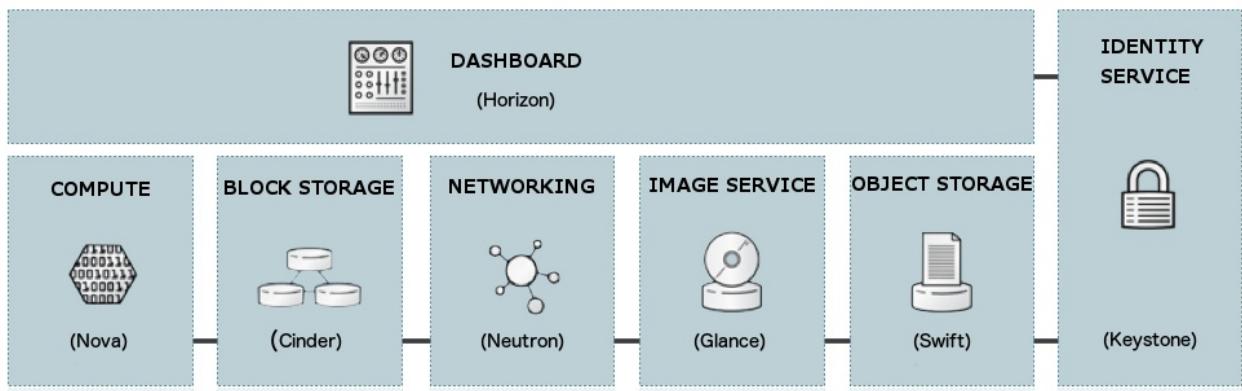


Abbildung 1: Logischer Aufbau von OpenStack. Quelle: <https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html>

Nova (Compute)

Das Kernstück eines jeden Cloud-Dienstes ist der Cloud Computing Fabric Controller (CCFC). Dieser verwaltet die Ressourcen, welche in der Regel aus virtuellen Maschinen bestehen. Diese virtuellen Systeme werden über Nodes, also Compute-Knoten verteilt. Dies geschieht mittels eines Hypervisors (Virtual-Machine-Monitor). Ein Hypervisor ist dabei ein System welches zur Abstrahierung der tatsächlichen Hardware und den zusätzlichen Betriebssystemen eingesetzt wird. Auf dieser Ebene werden also virtuelle Umgebungen definiert, wobei man recht frei von den gegebenen Hardware-Bedingungen operieren kann. Als praktische Implementierung verwendet OpenStack dafür KVM und Xen.

KVM

KVM steht für Kernel-based Virtual Machine und ist ein open-source Typ2-Hypervisor zur Virtualisierung und seit Kernelversion 2.6.20 ein Bestandteil des Linux-Kernels. Bei der eigentlichen Virtualisierung kommt QEMU zum Einsatz; KVM sorgt für die notwendige Infrastruktur und QEMU für die virtuellen Elemente wie Festplatten, Grafikkarten, usw.

Verwendung findet dieser Hypervisor u.a. in Ubuntu und openSUSE.

Xen

Xen ist ein Typ1-Hypervisor um mehrere Systeme auf einem physischen Computer zu virtualisieren. Xen unterteilt die virtualisierten Systeme in sogenannte Domänen, die untereinander keinerlei Beziehung zueinander haben. Die Kommunikation und Steuerung erfolgt über die erste Domäne, welche in der Lage ist, andere Domänen zu verwalten und steuern.

Xen wird unterstützt von Unternehmen wie Microsoft, Intel und AMD.

Glance (Image Service)

Der Glance Service macht es möglich, Abbilder, also Images von virtuellen Maschinen den Nutzern zur Verfügung zu stellen. Die Anbindung erfolgt hier über REST. Mit Glance können ebenfalls Metadaten wie z.B. Betriebssysteme gespeichert werden.

Swift (Object Storage)

Zur Bereitstellung von Speicher kann auf Dienste wie SheepDog oder den hauseigenen Dienst von Openstack namens Swift zurückgegriffen werden. Ähnlich wie bei Nova, findet die Ansteuerung über eine REST API statt. Die Objekte werden hier gruppiert in Containern abgespeichert, welche wiederum an entsprechende Accounts gebunden sind bzw. diesen gehören.

Keystone (identity)

Die Komponente Keystone fungiert als Authentifizierungs- und Rechtesystem in OpenStack für die Nutzung von Clouds. Der Nutzer einer Cloud wird als Mandant von Keystone betrachtet, wenn er einen erlaubten Zugriff auf ein Projekt innerhalb der Cloud besitzt. Es können mehrere Benutzer einem Mandanten zugeordnet werden, welche mit unterschiedlichen Rechten ausgestattet sind. Die Konfigurationsmöglichkeiten weisen dabei eine recht hohe Komplexität auf.

Horizon (Dashboard)

Horizon ist das Webinterface (GUI) mit welchem man die unterschiedlichen Komponenten von OpenStack verwalten kann und mit der die einzelnen Funktionalitäten eingesehen und gesteuert werden können. Realisiert wird dies über eine REST API.

Cinder (Block Storage)

Cinder ist ähnlich wie Swift, nur dass es hier nicht um Objektspeicher, sondern um die Verwaltung von Blockspeicher geht. Blockspeichermedien sind z.B. Festplatten, USB Datenträger oder CD/DVDs.

Neutron (Networking)

Die Komponente Neutron übernimmt den Netzwerkdienst für OpenStack. Alle Netzwerkangelegenheiten, Subnetze und IP Adressen lassen sich hiermit verwalten. Hier wird auch der Zugang nach "Außen" definiert, also die IP, mit der die Cloud im öffentlichen Netzwerk erreichbar ist. Neutron hat auch eine eingebaute Firewall mit der sich das System absichern lässt und die Zugriffe reglementieren lassen.

Ceilometer (Telemetry Service)

Ceilometer sammelt Daten über alle anderen Komponenten hinweg um diese für zukünftige Komponenten bereit zu stellen. Die gemessenen Daten können abgespeichert werden und für die Auswertung genutzt werden, um z.B. Ressourcen nachzuverfolgen. Ereignisse in der Cloud-Infrastruktur werden zu messbaren Gegenständen umgewandelt (metering). Es ist dabei so abstrakt gehalten wie möglich, um die Daten für unterschiedlichste Tätigkeiten zu verwerten.

Metering im Detail beschreibt den Prozess der Informationsgewinnung von Ereignissen die fakturiert werden können. Die Informationen umfassen alle möglichen Attribute (Wer hat was wann getan und wie oft?). Das Ergebnis bezeichnet eine Sammlung von Tickets.

Heat (Orchestration Service)

Heat ist ein Dienst um einen Verbund von Cloud Applikationen zu orchestrieren. Dafür bietet es ein Template Format an, welches über eine REST API angesteuert wird. Ein Heat Template beschreibt dabei die Infrastruktur einer Cloud Applikation. So eine Template Datei ist für Menschen lesbar und auch bearbeitbar. Sie enthalten ebenfalls die Verbindungen zu Ressourcen. Gerade bei der Ausführung mehrerer Applikationen, die sensible Abhängigkeiten zwischeneinander besitzen, helfen solche Templates dabei, die Applikationen in der richtigen Reihenfolge und wohlgeordnet zu starten.

Zusätzlich erlauben die Templates auch bestimmte Ressourcen Typen zu erzeugen (Instanzen, Datenträger, Benutzer, etc.).

Trove (Database Service)

Trove bietet einen relationalen Datenbank Service an. Dieser kann von Administratoren und Nutzern verwendet werden.

Sahara (Data Processing Service)

Sahara zielt darauf ab, datenverarbeitende Frameworks wie z.B. Apache Hadoop auf OpenStack lauffähig zu machen. Die Einbindung solcher Frameworks findet grundsätzlich über umfangreiche Konfigurationen statt, um die notwendigen Ressourcen (Knoten, Hardware) dafür anzupassen.

Sicherheitsaspekt

Domänen

Zur Definition und Beschreibung des Sicherheitsaspekts von OpenStack wurde die entsprechende Dokumentation in [5] verwendet. OpenStack fasst Benutzer, Applikationen und Netzwerke, welche die gleichen Authentifizierungsanforderungen besitzen, in sogenannten Sicherheitsdomänen (oder Security Domains) zusammen. Standardmäßig um CloudStack einzusetzen gibt es minimal vier Sicherheitsdomänen:

- Public
- Guest
- Management
- Data

Alle Zugriffe, Rechte, Komponenten, usw. können in diese Domänen abgebildet werden. Es ist wichtig anzumerken, dass manche Elemente jedoch auch in gemischten Domänen vorkommen können, obwohl sie sich physisch in einem Netzwerk befinden. Die Sicherheitsregeln und Domänen sollten daher immer an der tatsächlichen Topologie des Systems angepasst sein, und der Art der Cloud Instanz (private, öffentliche oder hybride Cloud?).

Public ist nicht vertrauenswürdig, da es sich dabei um Netzwerke handelt, wo der Cloud Betreiber keinerlei Kontrolle hat wie z.B. das Internet. Aktionen in dieser Domäne sollten daher sehr vorsichtig gehandhabt und besonders geschützt werden. Die Guest Domäne ist für den Verkehr zwischen Instanzen gedacht. Es betrifft die Daten die von den Instanzen generiert werden, nicht jedoch Dienste der Cloud die über API Aufrufe ausgeführt werden. Diese Domäne ist ebenfalls nicht vertrauenswürdig, egal ob es sich um eine private oder public Cloud handelt. Im Falle einer privaten Cloud kann es als vertrauenswürdig gelten, aber nur unter geeigneten Sicherheitsrichtlinien. In der Management Domäne interagieren Services, also Dienste miteinander. Hier werden sensible Daten übertragen wie z.B. Parameter der Cloud Konfiguration, Passwörter, Benutzernamen, usw. Der Zugriff auf diese Domäne sollte sehr gut abgesichert und überwacht sein. Diese Domäne wird als vertrauenswürdig betrachtet. Die Domäne Data enthält Daten bezüglich der Datenträger, also dem Storage Service. Diese Domäne ist ebenfalls vertrauenswürdig, und es muss strengstens auf die Integrität und Sicherheit der Daten geachtet werden.

Bridges

Unter einer Bridge versteht man eine Komponente, die sich in mehr als einer Domäne befindet, z.B. in einer vertrauenswürdigen und einer nicht vertrauenswürdigen Domäne. In diesem Fall muss eine Komponenten besonders vorsichtig konfiguriert werden. Oft stellen diese Komponente Schwachstellen dar und sind daher für Angriffe besonders verwundbar. Um diesen Fällen vorzubeugen, gibt es die Option, eine Bridge besonders abzusichern, das heißt, sicherer als beide Domänen in der sich die Komponente befindet.

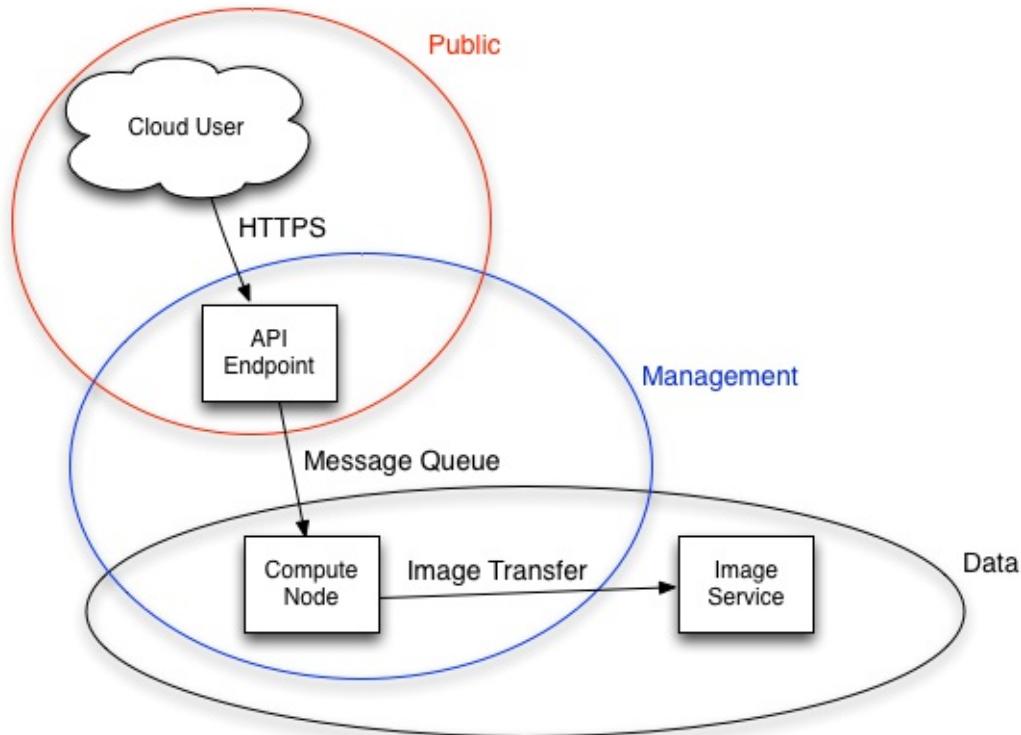


Abbildung 2: Beispielhaftes Szenario. Der API Endpoint in der Domäne Management überschneidet sich mit der Domäne Public. Quelle: <https://docs.openstack.org/security-guide/introduction/security-boundaries-and-threats.html>

In der Abbildung 2 ist ein solches Beispiel dargestellt. Der API Endpoint (aus der Domäne Management) befindet sich ebenfalls in der Domäne Public. Dies macht einen Angriff auf diese Stelle sehr wahrscheinlich, da die Sicherheitsrichtlinien entsprechend niedrig sind in dieser Domäne.

Angriffstypen

Das nachfolgende Diagramm stellt die möglichen Angriffe exemplarisch dar. Wieder sollte die Frage anstehen, welches Szenario für die Cloud am wahrscheinlichsten ist? Handelt es sich um eine private oder public Cloud, wie sensibel sind die Daten, die Ressourcen, etc.? Anhand dieser Analyse kann ermittelt werden, welches Angriffsszenario am wahrscheinlichsten ist, und welche Maßnahmen man anschließend ergreifen muss, um sich dagegen zu wehren.

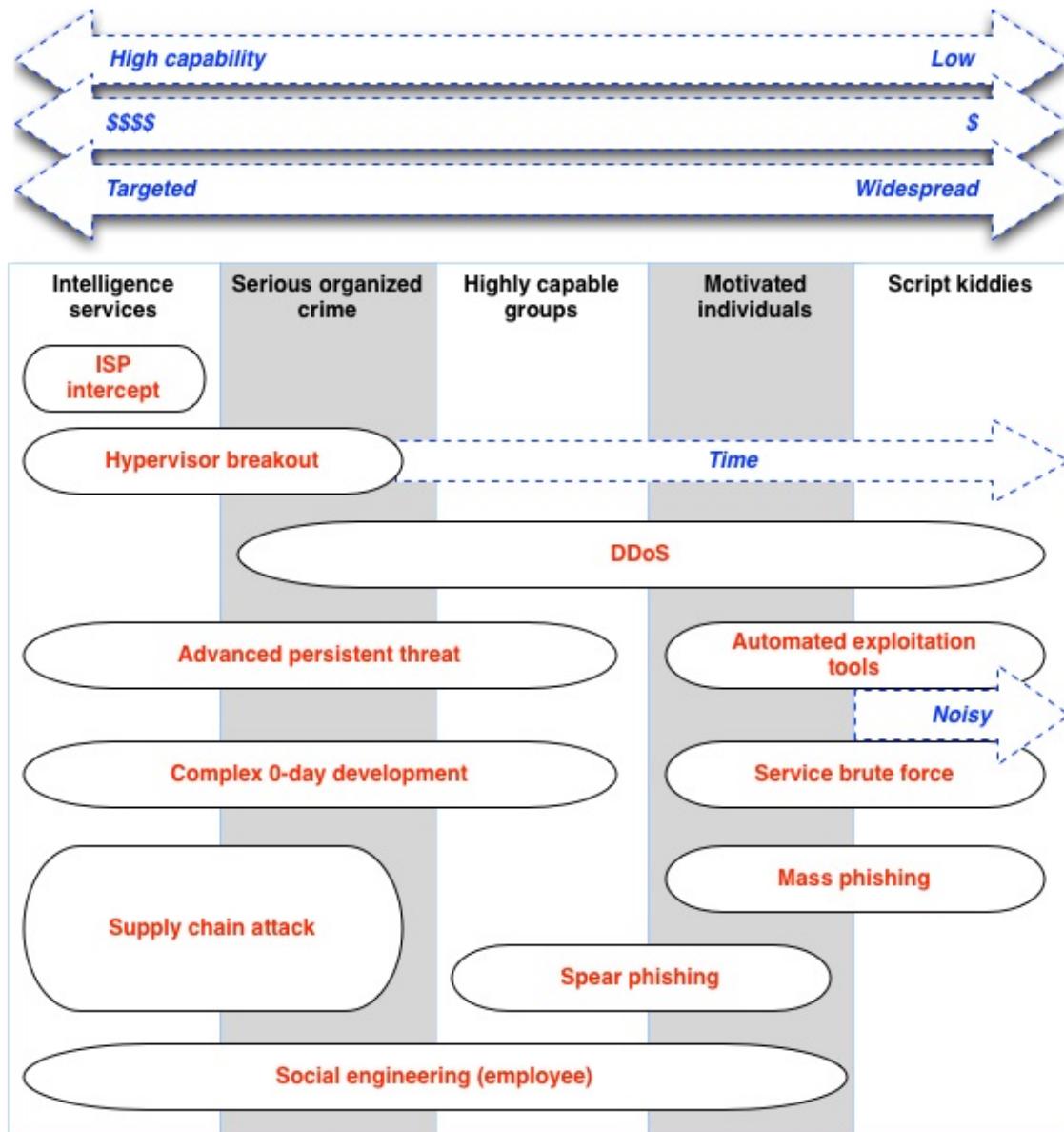


Abbildung 2: Verschiedene Angriffsszenarien. Quelle: <https://docs.openstack.org/security-guide/introduction/security-boundaries-and-threats.html>

- DDoS: Bei einem DDoS (Distributed Denial of Service) Angriff wird der Server mit einer extrem hohen Anzahl von Anfragen (z.B. HTTP Requests oder Pings) überflutet um bestimmte Dienste des Servers damit zu überlasten und die Aufgabe ihrer Funktionalität zu verursachen. Dies geschieht oft von mehreren Maschinen aus oder von einem Bot-Netz. Bei einem DoS Angriff geschieht dies nur von einer Maschine oder z.B. werden Bugs von Systemkomponenten ausgenutzt um Ressourcen des Systems auszulasten bis diese unbrauchbar geworden sind oder das System sogar abstürzt.
- Hypervisor Breakout: Bei diesem Angriff gelingt es dem Angreifer aus der virtualisierten Maschine herauszukommen und auf das Host-System zuzugreifen um so Schaden zu verursachen.
- Pishing: Bei einem Pishing-Angriff wird versucht über gefälschte E-Mails, Webseiten

oder andere Medien einen Nutzer reinzulegen um seine Identität zu stehlen. Spear-Pishing stellt dabei eine ausgedachtere Variante dar, indem die Angreifer die Identität einer vertrauenswürdigen Organisation übernehmen (z.B. Bank, Kunde, Dienstleister).

- Supply Chain Attack: Bei diesem Angriff wird versucht, auf weniger gesicherte Elemente in der Topologie Angriffe durchzuführen, die unterstützende und erweiternde Funktionen auf das Gesamtsystem haben. Dadurch verliert das System oft wichtige Ressourcen die einen funktionierenden Betrieb unmöglich machen.
- Automated exploitation tools: Mit diesen Werkzeugen werden automatisiert nach Schwachstellen in der Software gesucht um diese dann böswillig auszunutzen.
- Service brute force: Bei einem solchen Angriff werden z.B. bei einer Passwort und Nutzernname Eingabe einfach alle möglichen Kombinationsmöglichkeiten durchgetestet.
- Complex 0-day development: Zero-Day Angriffe sind besonders gefährlich, da hier Schwachstellen ausgenutzt werden die bekannt wurden, bevor der Anbieter ein entsprechendes Patch oder Fix veröffentlichen konnte.

Prinzipiell sind alle diese Angriffe auf die Modelle einer Cloud (IaaS, PaaS, ...) anwendbar. Jedoch sind die Auswirkungen unterschiedlich. Als Beispiel, wenn man sich illegal Zugang zu einer IaaS Cloud verschafft hat, kann Schadcode einfacher hochgeladen und ausgeführt werden als auf einer SaaS Cloud. Dieser Fall muss aber natürlich entsprechend berücksichtigt werden.

CloudStack

CloudStack ist ein open-source Cloud Projekt vom Typ IaaS. Die Beschreibung wurde der Dokumentation in [6] entnommen. Es ist in Java geschrieben und wird von der Apache Software Foundation betrieben. Ein Vorteil von CloudStack ist die Unabhängigkeit von einer bestimmten Plattform. Als Webinterface wird eine AJAX-basierte GUI verwendet. Ähnlich wie OpenStack, ist es auch hier möglich KVM als Hypervisor zu nutzen, falls das System auf einer geeigneten Linux Distribution läuft, wie Ubuntu. Andernfalls gibt es auch die Möglichkeit Windows Server 2012 R2 zu verwenden, Xen und noch einige andere.

OpenNebula

OpenNebula ist ebenfalls ein open-source Projekt zum Betrieb einer Cloud als IaaS. Zur Beschreibung der Cloud wird die Dokumentation in [7] herangezogen. Als Vorteil ist zu nennen, dass mit OpenNebula hybride Clouds möglich sind und auch die Nutzung von Computerclustern, also einer verteilten Infrastruktur. Als weitere wichtige Merkmale sind die Standardisierung und die Interoperabilität zu nennen. Das macht OpenNebula sehr flexibel und dynamisch. Im Gegensatz zu OpenStack ist OpenNebula wesentlich schlanker, und laut

eigenen Aussagen einfach und simpel zu installieren, verwenden und warten. Geeignet ist diese Lösung für Linux Systeme und als Hypervisor lässt sich ebenfalls KVM oder Xen nutzen.

Vergleich und Fazit

Um die unterschiedlichen Lösungen in Relation zueinander zu setzen, ist es möglich, eine Klassifizierung der Cloud Dienste anzugeben, wie dies in [2] getan wurde. Dazu kann man die Cloud Plattformen in zwei verschiedene Kategorien einteilen: Datacenter Virtualization und Infrastructure Provision. Das erstere, versteht Clouds als eine Erweiterung der Datenverarbeitung. Eine Cloud-Infrastruktur soll die virtuellen Ressourcen verwalten und diesen Prozess vereinfachen. Die zweite Kategorie meint Clouds on-premise, also vor-Ort, und stellt daher die Anforderung zusätzliche virtuelle Ressourcen auf Abruf bereitstellen zu können.

Die Abbildung 3 stellt die Unterschiede und Spezifikationen die diese Einteilung mitsich bringt übersichtlich dar.

	DATACENTER VIRTUALIZATION	INFRASTRUCTURE PROVISION
Applications	Multi-tiered applications defined in a traditional, "enterprise" way	"Re-architected" applications to fit into the cloud paradigm
Interfaces	Feature-rich API and administration portal	Simple cloud APIs and self-service portal
Management Capabilities	Complete life-cycle management of virtual and physical resources	Simplified life-cycle management of virtual resources with abstraction of underlying infrastructure
Cloud Deployment	Mostly private	Mostly public
Internal Design	Bottom-up design dictated by the management of datacenter complexity	Top-down design dictated by the efficient implementation of cloud interfaces
Enterprise Capabilities	High availability, fault tolerance, replication, scheduling... provided by the cloud management platform	Most of them built into the application, as in "design for failure"
Datacenter Integration	Easy to adapt to fit into any existing infrastructure environment to leverage IT investments	Built on new, homogeneous commodity infrastructure

Abbildung 3: Gegenüberstellung der Anforderungen und Anwendungen von Cloud Lösungen. Quelle: <https://opennebula.org/eucalyptus-cloudstack-openstack-and-opennebula-a-tale-of-two-cloud-models/>

Eine Einteilung von CloudStack, OpenNebula, Eucalyptus und OpenStack lässt sich folgendermaßen darstellen, wie in Abbildung 4 abgebildet.

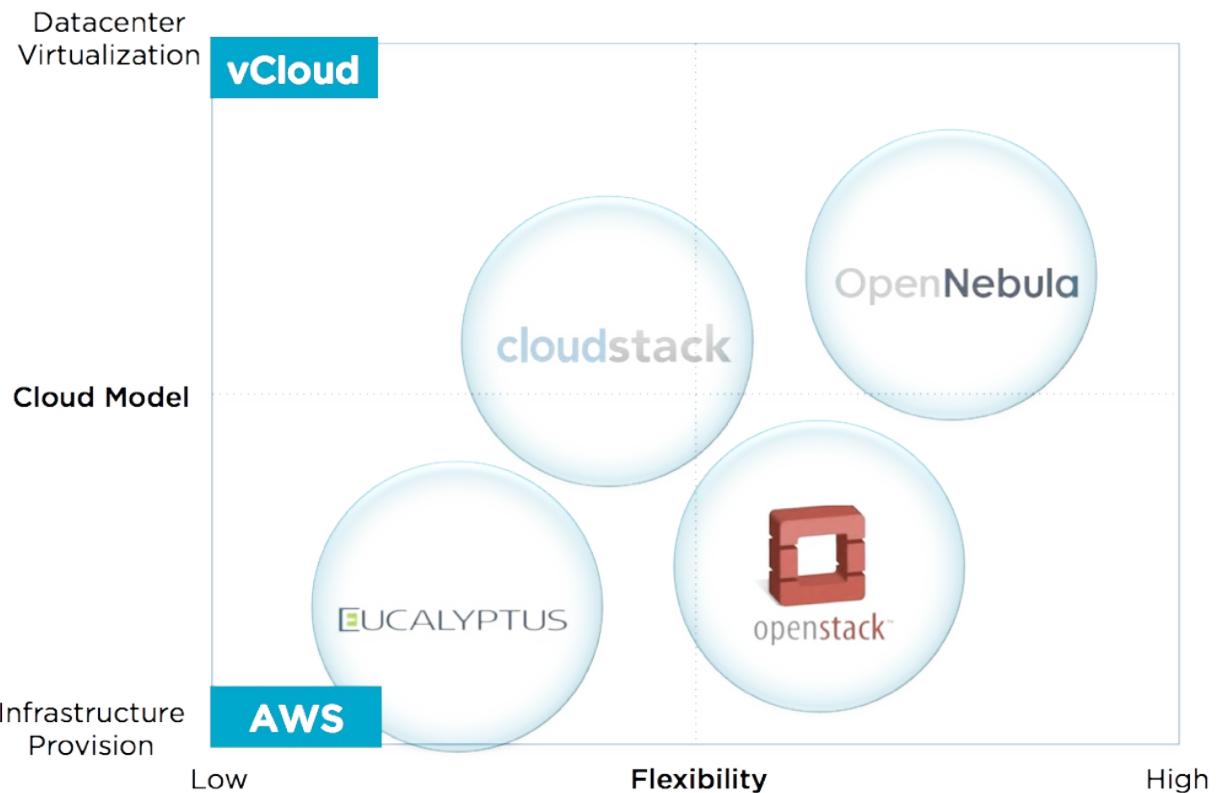


Abbildung 4: Positionierung vier unterschiedlicher Cloud Plattformen. Quelle: <https://opennebula.org/eucalyptus-cloudstack-openstack-and-opennebula-a-tale-of-two-cloud-models/>

Anhand von [1] lässt sich abschließend folgendes Fazit ziehen: OpenStack hat eine hohe Komplexität, jedoch ist es die am weitesten verbreitete Cloud Lösung. Mehr als 150 Firmen wie z.B. AMD, Dell, IBM und Yahoo verwenden OpenStack und leisten auch ihren Beitrag zur Weiterentwicklung des Projekts. Dies macht OpenStack zu einem. Die erste stabile Version von CloudStack erschien 2013 und ist daher noch recht jung. Kritiker bemängeln die umständliche Installation, die einige Kenntnisse vom Anwender abverlangt. Der größte Nutzer von CloudStack, das Unternehmen DataPipe, nennt einige Gründe für ihre Wahl; zu den Wichtigsten gehören die Skalierbarkeit des Datenspeichers und die Nutzung von hochgeschwindigkeits VMs, welche damit ressourcensparender arbeiten. Auch wenn CloudStack mehr an Popularität gewinnt und in der Zukunft womöglich interessanter sein wird, ist die momentan verbreiteste und akzeptierteste Lösung OpenStack.

Anhand der grafischen Positionierung und den Beschreibungen der einzelnen Cloud Anbieter, wird es klar, dass die Wahl der Cloud immer vom speziellen Anwendungsfall abhängig ist. Daher haben die Einsteiger-freundliche OpenNebula, CloudStack und auch die umfangreiche OpenStack Plattform jeweils ihre Existenzberechtigung und man darf gespannt sein, wie sie in Zukunft vom Markt aufgenommen werden und sich weiter entwickeln.

Sicherheit

Autor: André Kaleja

Der größte Vorteil durch Clouds ist das hohe Performanz/Kosten Verhältnis. Grade kleine bis mittelständische IT Betriebe, können von den großen Recheneinheiten profitieren, ohne dabei selber in den Bau und den Kauf dieser Datenzentren investieren zu müssen. Der Wunsch nach einer kritischen cyber-physischen Infrastruktur löst einige Sicherheitsbedenken aus, da riesige Mengen an sensiblen Daten übertragen und in großen Datencentern verarbeitet werden müssen. Außerdem kann es vorkommen, dass diese sensiblen Daten an Vermittler ohne volle Zugriffsrechte, weitergegeben werden müssen. [9 Part V S.357]

Viele Sicherheitsprobleme konzentrieren sich auf die neuen Eigenschaften der Cloudbeschaffung. Allerdings stellt ihre einzigartige und innovative Natur die Industrie und Entwickler vor immer neue Herausforderungen. [10 Kap 5 S. 37] Auch wenn die Virtualisierung keine Technologie ist, die aus der Cloud hervorkam, sondern eine eigenständige Komponente darstellt, wird ihr durch die Cloud immer mehr Aufmerksamkeit zugeteilt. Sie ist inzwischen eine der Hauptmerkmale einer Cloud und von extremer Wichtigkeit. Allerdings ermöglicht die Virtualisierung VM (Virtual Machine) zu VM Attacken, aufgrund der VMM (Virtual Machine Management) Schwachstellen. [10 Kap 5 S. 37]

Laut dem National Institute of Technology (NIST) sind die Bereiche Sicherheit, Interoperabilität und Portabilität die größten Barrieren für den Einsatz von Clouds. Da es bisher nur sehr wenige Standards für Clouds gibt, sind Kunden abhängig von einem Anbieter. Zusätzlich gibt es noch keine Möglichkeit, die in einer Cloud gespeicherten Daten auf eine Cloud eines anderen Anbieters zu übertragen. Diesem Problem könnte man mit sogenannten „Interclouds“ (eine Art Netzwerk von Clouds) entgegenwirken. Interclouds vereinen Interoperabilität, universelles und Utility Computing, sowie Datenspeicherung miteinander. Sie würden die Anbieterabhängigkeiten beseitigen und die freie Datenübertragung zwischen Clouds ermöglichen. Da Clouds via Internet erreicht werden, erben sie alle bekannten Sicherheitsprobleme des Internets. Zusätzlich sind auch neuartige Sicherheitsprobleme durch die Cloud entstanden. [10 Kap. 1.2 S. 5]

Die Tatsache, dass ein Netzwerk und auch seine physischen Bestandteile geteilt werden, ist einer der Hauptgründe, weswegen viele potenzielle Cloud Kunden noch zögern. Sind Clouds und ihre Dienste nicht richtig überprüfbar und nachvollziehbar, sinkt das Vertrauen zwischen Kunde und Anbieter. Das Outsourcen der IT Pflichten an Third Party Cloud Anbieter ist ein Problem, dass noch überwunden werden muss. [10 Kap 5 S. 37] Das Datenoutsourcing bringt viele Vorteile, allerdings wirft es auch einige Datenschutz und

Sicherheitsbedenken auf. Tatsächlich sind diese Daten gar nicht mehr unter der direkten Kontrolle des Inhabers, dadurch sind ihre Vertraulichkeit und Integrität einem großen Risiko ausgesetzt. Solche outgesourceten Daten müssen nicht nur vor externen Nutzern, sondern auch vor böswilligen Insidern geschützt sein. In den meisten Fällen sollten externe Server nur für die Erreichbarkeit und das Speichern der Daten sorgen und keine Berechtigungen besitzen, die Daten zu lesen. Solch ein effektiver, praktischer Datenschutz ist sehr komplex und benötigt dementsprechend ein effektives Design. Dieses sollte den Dateninhabern die Möglichkeit liefern, selbst die Anforderungen an den Datenschutz spezifizieren zu können. [9 Kap. 14.1 S.361]

Aufgrund der noch nicht ausgereiften Sicherheit, zögern die meisten Menschen, ihr ganzes Business auf Clouds abzulegen. Das bremst den Fortschritt der Clouds, da sich Forschung und Industrie mehr darauf konzentrieren müssen Sicherheitsprobleme zu beheben, anstatt das Potential der Clouds zu erforschen. Die meisten Menschen sind einfach noch nicht bereit dazu, ihre sensiblen Daten in der Cloud zu speichern. Daraus geht unweigerlich hervor, wie wichtig die Sicherheit ist. [10 Kap. 1.2 S. 5]

Durch große Risiken wie Datenverluste, Datenenthüllungen und die dadurch entstehenden finanziellen Verluste, steht viel auf dem Spiel. Clouds bieten einige ansprechende Features und Vorteile, aber solange die Sicherheitsrisiken nicht besser verstanden sind, werden sich die wichtigsten Firmen und Kunden zurückhalten. Clouds sind daher risikoreich für alle Beteiligten: Kunden, Firmen und Investoren. [10 Kap. 1.2 S. 5]

Um schneller auf diesem Gebiet voranzukommen, sollten sich Studien und Untersuchungen nicht mehr nur noch auf vereinzelte Probleme beziehen. Es muss die komplette Breite der Cloud Sicherheitsprobleme betrachtet und analysiert werden, um das Vertrauen in Clouds stärken zu können. [10 Kap 5 S. 37]

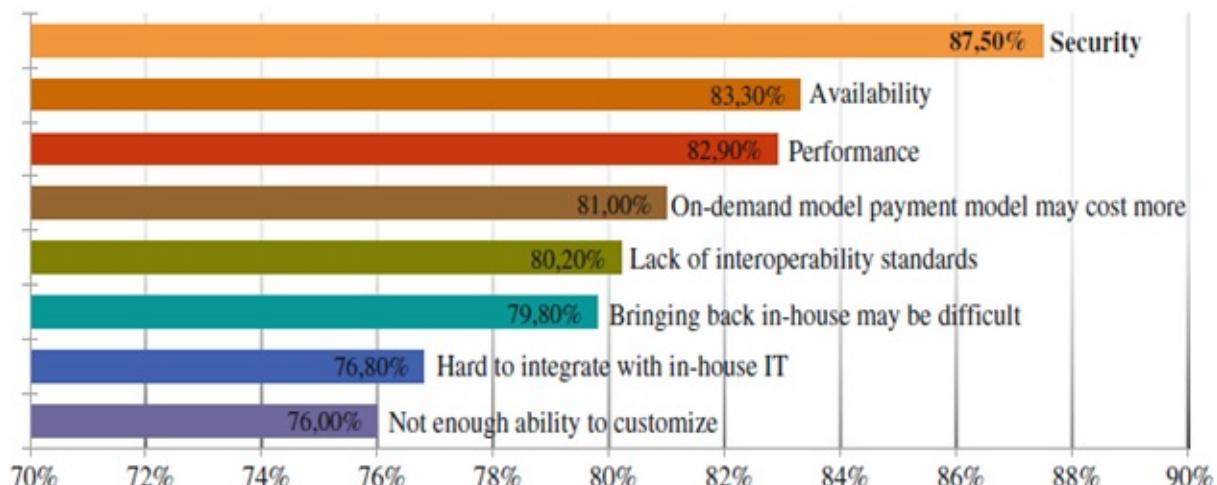


Abbildung 1: Herausforderungen und Probleme von Clouds und ihre Relevanz entsprechend der International Data Corporation (IDC) und zugehörige Ergebnisse der Cloud User Survey [10 Fig. 1 Kap 1.2 S. 6]

Interessante verwandte Arbeiten

In der Industrie aber auch in der Literatur genießt das Thema Cloud Sicherheit immer mehr Beliebtheit. Einige verschiedenste internationale Konferenzen haben sich ausschließlich darauf konzentriert. Darunter fallen Organisationen wie die Association for Computer Machinery (ACM) Workshop on Cloud Computing Security, die International Conference on Cloud Security Management und die einzige europäische Konferenz, Secure Cloud, die bereits drei Auflagen veröffentlicht hat. Das steigende Interesse beschränkt sich längst nicht mehr auf wissenschaftliche Veröffentlichungen oder Konferenzen, inzwischen befassen sich auch internationale Magazine und Zeitschriften mit diesem Thema. Die Untersuchung in (12) z.B. befasst sich mit Sicherheits- und Datenschutzangelegenheiten von Cloud Anbietern. Die Sicherheit wurde in Hinblick auf Erreichbarkeit, Vertraulichkeit, Datenintegrität, Kontrolle und Prüfungseigenschaften diskutiert. Beim Datenschutz lag der Fokus auf veralteten Datenschutz Vorgängen, die Informationen nicht mehr vor Regierungen oder third-parties beschützen können. Außerdem wurden die multi-location Probleme von Clouds besprochen. Es ist eine Grundvoraussetzung, zu wissen in welchem Land sich die Daten befinden. [10 Kap 2 S. 7]

In der Untersuchung (13) standen erneut die Aspekte Vertraulichkeit, Datenschutz, Integrität und Erreichbarkeit unter Betrachtung. Als Lösungsansatz dieser Untersuchung wurde eine vertrauenswürdige third-party Lösung vorgeschlagen, um Sicherheitsbedrohungen auf Vertraulichkeit, Integrität, Authentizität und Erreichbarkeit auszumerzen. Der Ansatz kombiniert Public Key Infrastructure (PKI), das Lightweight Directory Access Protocol (LDAP) und Single Sign-On (SSO). Die Prämissen der Untersuchung war es darzulegen, dass die Vorteile der Cloud den Defiziten überlegen sind. [10 Kap 2 S. 7]

Eine andere Studie zielte auf die Sicherheitsprobleme der Cloud Modelle ab. (20). Jedes Modell wurde einzeln betrachtet, um dabei die signifikantesten Schwachstellen, Bedrohungen und Risiken zu finden. Es ist auffällig, dass das Software as a Service (SaaS) Modell die Mehrheit an Problemen hervorbrachte. [10 Kap 2 S. 7]

Die Sicherheit und Datenschutz Themen wurden erneut in (14) diskutiert. Vertraulichkeit, Integrität, Verantwortung und Datenschutz wurden als die signifikantesten Attribute identifiziert. Zu jeder Eigenschaft wurden Sicherheitsprobleme, mit zugehörigen Verteidigungslösungen, beschrieben. [10 Kap 2 S. 7]

In (15) sind einige Sicherheitsherausforderungen als Schlüsselthemen spezifiziert. Diese Themen beziehen sich auf Ressourcenvergabe, Systemüberwachung und Protokollierung, Computer forensic, Virtualisierungen, Multi Vermietungen, Authentifizierung und Autorisierung, Erreichbarkeit und Cloud Standards. Die Untersuchung konzentrierte sich danach auf die Service Level Agreements (SLAs). [10 Kap 2 S. 7]

Wichtige Konzepte der Cloud Sicherheit

Da Clouds auf Virtualisierungstechniken basieren, ist es wichtig diese zu identifizieren und beschreiben und zu wissen, welche Elemente das Rückgrat der Virtualisierung bilden. Nutzer können auf speziell für die Cloud entwickelte Applikationen zugreifen. Daher ist es umso wichtiger, dass bei der Software besonders auf die Sicherheit geachtet wird. [Kap 3.4 ab S. 11] Die wichtigsten Cloud Sicherheitskonzepte sind: Virtualisierung, multi-tenancy, Cloud Software, Daten Outsourcing, Datenspeichersicherheit und Standardisierung und Vertrauen. Diese werden näher beschrieben in [10 Kap 3.4 ab S. 11]

Generelle und Coud Spezifische Probleme

Unter das Wort Sicherheitsproblem fallen viele Probleme:

- Schwachstellen (Defekte oder Schwächen eines Systems)
- Bedrohungen (Ausnutzen von Schwachstellen)
- Lücken (Defekte oder Schwächen eines Systems)
- Angriffe (Ausnutzen einer Bedrohung)
- Risiken (Wahrscheinlichkeit eine Schwachstelle auszunutzen)

Die meisten Personen verstehen den Unterschied zwischen generellen und cloud spezifischen Problemen nicht. [10 Kap 3.5 S. 13-14] Clouds basieren stark auf den Möglichkeiten, die durch verschiedene Core Technologien, wie Web Applikationen, die Cloud Dienst Modelle, Virtualisierungen und kryptografischen Mechanismen, bereitgestellt werden. Ein Cloud spezifisches Sicherheitsproblem kommt im Wesentlichen aus einer Core Technologie. Sie haben ihre Wurzeln in einem der fünf essentiellen Charakteristiken, welche von NIST vorgeschlagen wurden.

Die fünf Charakteristiken sind:

- On-demand self-service
- Broad network access
- ressource pooling
- rapid elasticity
- measured service

Um einen besseren Überblick über die Sicherheitsprobleme zu bekommen, sollte man versuchen sie zu kategorisieren. [10 Kap 3.5 S. 13-14]

Cloud-Modelle und ihre Sicherheit

Die immer steigende Bandbreite treibt die Entwicklung des Web 2.0 und damit einhergehende neue Klassen an Diensten voran. Cloud Systeme bestehen aus einer three-Model Architecture. Diese drei Stufen sind Infrastructure as a Service (IaaS), Platform as a Service (PaaS) und Software as a Service (SaaS). [10 Kap 3.1 S. 8]

IaaS beschäftigt sich mit der Investition der Geschäftsleute in die IT Infrastruktur. Anstatt große Mengen in Hardware und technisches Personal zu stecken, stellt IaaS virtuelle Server zur gebrauchten Zeit bereit. Amazon Web Services (AWS) ist ein passendes reales Beispiel für solch einen Anbieter, dabei zahlt der Kunde nur das was er auch braucht. Das IaaS Model stellt eine grundlegende Sicherheit, durch Perimeter Firewalls und Auto Balancing, bereit. Der Anbieter sollte aber die Sicherheit des Virtual Machine Managers (VMM) gewährleisten. [10 Kap 3.1 S. 8]

PaaS liefert Kunden die Möglichkeit ihre eigenen Cloud Applikationen, durch bereitgestellte Plattformen und Frameworks, zu entwickeln. Im Gegensatz zu SaaS liefert PaaS mehr Flexibilität in Sachen Sicherheit, da kundenfertige Features bereitgestellt sind. Allerdings können durch unsicher integrierte Integrated Development Environments (IDEs) und Application Programming Interfaces (APIs) mehr Sicherheitslücken entstehen. [10 Kap 3.1 S. 8]

SaaS erlaubt Applikationen auf Clouds zu hosten, um Anforderungsressourcen in Form von Diensten via Internet zu liefern. In diesem Modell tauchen die meisten Sicherheitsprobleme auf, die in den meisten Fällen die Datenspeicherung betreffen. Cloud Anbieter müssen sicherstellen, dass Nutzer keine Möglichkeiten haben, die Daten anderer Nutzer zu verstehen oder Zugriff zu bekommen. [10 Kap 3.1 S. 9]

Diese drei Modelle bilden den Grundstein von Clouds. Allerdings fließen sie immer mehr zu einer Art Anything as a Service (XaaS) zusammen, weil sie virtuell unendlich sein können und daher alles und jeden in Form von Diensten unterstützen können. [10 Kap 3.1 S. 9]

Sicherheit der Datenzentren

Clouds sind Computersysteme in speziell designten Räumen, um die massive Anzahl an Servern und Netzwerken unterzubringen. Diese Räume werden in Hinsicht auf geologische und umwelttechnische Aspekte wie Ort, Temperatur, Luftfeuchtigkeit und Erdbeben gebaut. Auch politische, regierungstechnische und energiesparende Aspekte spielen eine große Rolle. [10 Kap 3.2 S. 9]

Sicherheitsprobleme identifiziert von Organisationen

Cloud Sicherheit interessiert nicht nur Entwickler sondern auch Unternehmen. Die folgenden Arbeiten waren bahnbrechend für die Cloud Sicherheit (16).[10 Kap 4.1 S. 15]

TheGartner veröffentlichte das Dokument mit dem Namen „Assessing the Security Risks of Cloud Computing“ (17). Das Dokument beschreibt sieben Sicherheitsrisiken mit denen sich Kunden auseinandersetzen sollten, bevor sie sich einem Anbieter gegenüber verpflichten. Die sieben Risiken sind Nutzerzugriff, Einhaltung gesetzlicher Vorschriften, Datenstandort, Datentrennung, Wiederherstellung, Nachforschungsunterstützung und Langzeifähigkeit. [10 Kap 4.1 S. 15]

Die European Network and Information Security Agency (ENISA) ist verantwortlich für Cyber Sicherheitsprobleme in der EU und veröffentlichte das Dokument „Cloud Computing: Benefits, Risks and Recommendations“. (18) Acht Cloud spezifische Risiken werden hier als Top Risiken definiert, auch aus Sicht der Kunden. Sie sind Regierungsausfall, Ausschaltsperrre, Isolationsfehler, Übereinstimmungsrisiken, Management Schnittstellen Gefährdung, Datenschutz, unsichere unvollständige Datenlöschung und böswillige Insider. [10 Kap 4.1 S. 15]

Die Cloud Security Alliance (CSA) veröffentlichte das Dokument „Top Threats to Cloud Computing“. (19) Es beschreibt Top Bedrohungen, echte Beispiele und Abhilfemaßnahmen. Die folgende Tabelle zeigt die Bedrohungen und die betroffenen Service Modelle. [10 Kap 4.1 S. 15]

Threat #	Name	IaaS	PaaS	SaaS
1	Abuse and Nefarious use of cloud computing	✓	✓	✗
2	Insecure interfaces and APIs	✓	✓	✓
3	Malicious insiders	✓	✓	✓
4	Shared technology issues	✓	✗	✗
5	Data loss or leakage	✓	✓	✓
6	Account of service Hijacking	✓	✓	✓
7	Unknown risk profile	✓	✓	✓

Tabelle 1: Top Bedrohungen des Cloud Computing beschrieben von der CSA [10 Table 1 Kap 4.1 S. 15]

Model	Ownership	Management	Location	Cost	Security
Public	TP	TP	Off-site	Low	Low
Private and community	O or TP	O or TP	On-site	High	High
Hybrid	O and TP	O and TP	On-site and Off-site	Medium	Medium
VPC	B	B	B	B	High

Tabelle 2: Zusammenfassung der Hauptcharakteristiken von Cloud Entwicklungsmodellen, Zuständigkeit (Organisation O, Third-Party (TP) or both (B)), Management (O, TP or B), Standort (Off-site, On-site or B), Kosten (Low, Medium, High) und Sicherheit (Low, Medium, High), VPC steht für Virtual Private Cloud [10 Table 2 Kap 4.1 S. 16]

Einleitung in die Überwachung

Das Netzwerkmanagement entwickelt sich stetig weiter und ist inzwischen riesig und komplex. Es gibt sehr viel Tools, Software und Hardware Produkte, die Administratoren helfen, Netzwerke zu überwachen und zu steuern. Sicherheit, Performanz, Verlässlichkeit und viele Klassen an Diensten müssen dauerhaft überwacht werden.[11 Kap 22.1 S. 565] Es spielt längst nicht mehr nur noch die Überwachung eine Rolle. Zusätzlich optimiert man den Datenfluss und den Datenzugriff in einer sich ständig ändernden Umgebung. Dienste und Tools sind zahlreich vorhanden und variieren je nach Umgebung in der sie analysieren. Fehlermeldungen und ungewöhnliche Ereignisse können dem Admin durch E-Mails, SMS, Warnungen oder Alarmen mitgeteilt werden. Die Netzwerküberwachung macht nur Sinn, wenn man auch die richtigen Dinge überwacht. Dazu zählen z.B. die SLA Metriken und Sicherheitsbedingungen. Normalerweise werden Bandbreite, Verbrauch, Applikationsperformanz, Serverperformanz und Datenverkehr untersucht. [11 Kap 22.1 S. 565]

Netzwerküberwachungssysteme können inzwischen Switches, Router, Server, Desktops, backbone devices, Netzwerkpunkte, Telefone und mehr überwachen. Sie zeichnen automatisch Geräteaktivitäten auf. Die Funktionen unterscheiden die Geräte anhand von IP Adressen, Diensten, Arten (Switch, Router, etc.) und dem physischen Standort. Es ist ein offensichtlicher Vorteil, in Echtzeit zu wissen was genau passiert. [11 Kap 22.1 S. 566]

Überwachung und Kontrolle in der Cloud

Das Messen, Überwachen und Analysieren ändert sich, sobald die Umgebung und Dienste outgesourced werden. Cloud Computing ist anders als die bisherigen angebotenen Dienste. Das liegt daran, dass Eigenschaften wie Betrieb, Elastizität, gemessene Dienste und universeller Zugriff komplexer sind. Ohne geeignete Metriken ist das Hosten von IT Diensten auf öffentlichen, privaten oder hybriden Clouds problematisch. Vereinheitlichte Sichtbarkeit, Kontrolle und Verständnis der kompletten Cloud Infrastruktur sind unerlässlich für die Überwachung der Cloud. [11 Kap 22.2 S. 567] Es gibt zwei Seiten der Cloudüberwachung. Zum einen die Überwachung der Kern Infrastruktur, zum anderen das Überwachen der kompletten Dienste und ihrer Metriken. Die Überwachung der Kern Infrastruktur kann durch Cloud Ebenen abgegrenzt werden. Hier können allerdings Konflikte zwischen den

Interessen von Kunde und Anbieter entstehen. Die Überwachung der Dienste kann mithilfe von Metriken stattfinden. Ein Problem an dieser Überwachung ist, dass Cloud Anbieter ungenaue und irreführende Reports dieser Metriken angeben. [11 Kap 22.2 S. 567]

Überwachung und Kontrolle der Cloud Ebenen

Laut der CSA kann Cloud Computing in die folgenden sieben Ebenen eingeteilt werden: Facility(F), Network(N), Hardware(H), OS(O), Middleware(M), Application(A) und User(U). Die folgende Abbildung veranschaulicht diese Ebenen: [11 Kap 22.2.1 S. 567]

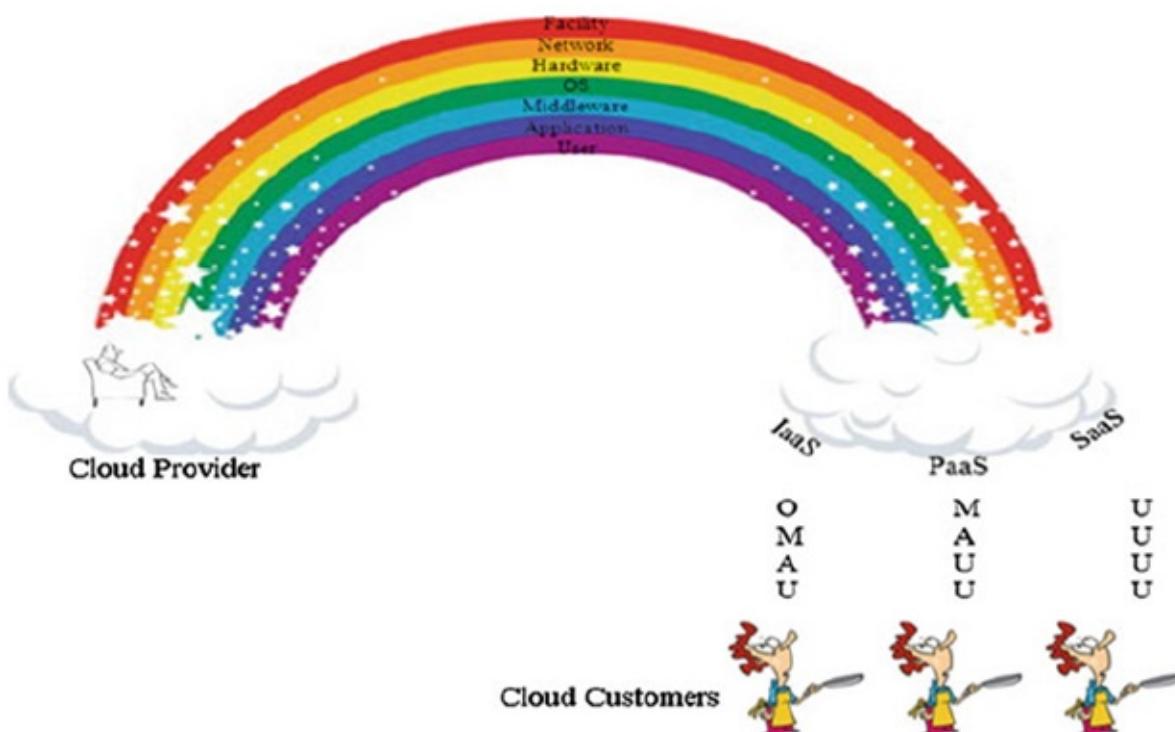


Abbildung 2: Cloud Ebenen, Implementierungsmodelle, Anbieter- vs. Kundenkontrolle [11 Fig 22.1 Kap 22.2.1 S. 568]

Die Kunden können abhängig von dem Implementierungsmodell, die Cloud bis zu einem gewissen Grad überwachen und kontrollieren.

In SaaS gibt es eine Nutzerebene, die von Cloud Kunden überwacht und kontrolliert werden soll. Die anderen Ebenen werden komplett vom Anbieter gehandhabt. [11 Kap 22.2.1 S. 568]

In PaaS gibt es eine Nutzer-, Applikations- und Middleware Ebene für den Kunden. In Diesem Modell, kann darüber verhandelt werden, wer die Middleware Ebene kontrolliert und überwacht. Die anderen Ebenen werden wieder vom Anbieter überwacht. [11 Kap 22.2.1 S. 569]

In IaaS gibt es eine Nutzer-, Applikations-, Middleware und OS Ebene für den Kunden. In Diesem Modell kann darüber verhandelt werden, wer für die OS Ebene zuständig ist. Die anderen Ebenen werden wieder vom Anbieter überwacht. [11 Kap 22.2.1 S. 569]

Ausblick auf zukünftige Möglichkeiten der Sicherheit in Clouds

Im Folgenden werden kurz drei Möglichkeiten der zukünftigen Sicherheit in Clouds beschrieben.

Datenverschlüsselung

Eine mögliche Lösung wäre es, die Daten die auf dem Server gespeichert werden sollen, zu verschlüsseln. Selbst wenn der Server bereits kompromittiert ist, schützt die Verschlüsselung die Daten vor Enthüllungen. Es gibt bereits Methoden, die externen Servern ermöglichen, Befehle und Queries mit verschlüsselten Daten, auszuführen. [9 Kap. 14.2 S.362]

Fragmentation zum Schutz vertraulicher Daten

Komplette Datensets zu verschlüsseln bringt aber auch Nachteile mit sich. Es ist nicht immer möglich, Queries und Zustände effizient auf diesen verschlüsselten Datensets auszuführen. Neueste Ansätze liefern eine Möglichkeit, die Datenvertraulichkeit mithilfe von Fragmentationen zu schützen und somit die Verschlüsselungen zu limitieren. [9 Kap. 14.3 S.371]

Datenintegrität schützen

Es ist auch sehr wichtig, die Integrität und Authentizität der Daten zu schützen. Nutzer und Organisationen sind immer abhängiger von Daten, die für den täglichen Betrieb gebraucht werden. Ein externer Server muss also garantieren können, dass die gespeicherten Daten in keiner Art und Weise verändert werden. Außerdem muss der Server richtige Antworten auf Queries liefern. [Kap. 14.4 S.379-380] Die Integrität kann auf den folgenden verschiedenen Ebenen gewährleistet werden: Tabellen, Attribute, Tupel und Zellen. Tabellen und Attribute können nur verifiziert werden, wenn der Kunde die komplette Tabelle oder Zeile erhält. Verifizierungen auf der Zellenebene leiden stark unter einem Verifizierungs-Overhead. Um die Integrität der Daten zu wahren, wird empfohlen mit digitalen Signaturen zu arbeiten. Dabei signiert der Inhaber jedes Tupel mit seinem privaten Schlüssel und die Signatur wird mit dem Tupel verkettet. Diese Verkettungen werden dann verschlüsselt. Bei Erhalt der

verketteten Tupel, kann der Empfänger überprüfen ob die Signatur mit den Tupels, unautorisierte Veränderungen beinhaltet. Ein Nachteil dieser Methode ist, dass die Verifizierungskosten für den Klienten linear mit der Menge an Tupeln steigen. [9 Kap. 14.4.1 S.380]

Ausblick

Autoren: Philipp Viertel, André Kaleja

Clouds werden immer vielseitiger eingesetzt. Regierungen, Industrie und private Nutzer entdecken mehr und mehr die Cloud für sich. Daten müssen in Zukunft immer flexibler und universeller verfügbar und greifbar sein. Ein wichtiges Beispiel für die Zukunft wird dabei die Automobil-Industrie mit der Entwicklung des autonomen Fahrens sein. Standorte der Autos, Kommunikation zwischen den einzelnen Fahrzeugen und weitere Informationen müssen alle in Clouds gespeichert sein und in Echtzeit abrufbar sein. Gerade hier, wird es sehr klar wie sensibel diese Daten sind und was für ein Schaden entstehen kann, wenn diese Daten kompromittiert werden würden. Neben dem autonomen Fahren betrifft dies auch Parkleitsysteme oder andere Einrichtungen des öffentlichen Verkehrs. In diesem Zusammenhang kann man davon sprechen das Autos mehr und mehr zu fahrenden Computern, bzw. Entitäten in einem Cloud Netzwerk werden. Das verlangt eine große Transformation und einen Umstieg auf IT Lösungen der großen Automobil-Hersteller. Alle entwickelten Lösungen müssen daher auch ein starkes Sicherheitsprofil aufweisen.

Nicht nur die Autoindustrie ist stark an der Cloud Technologie interessiert, sondern auch Software Entwickler die an unterschiedlichen Standorten arbeiten. Über die Clouds ist ein verteiltes Arbeiten jederzeit und von überall aus möglich.

Auch für Regierungen werden Clouds immer interessanter, vor allem in der öffentlichen Verwaltung. Hier wird ebenfalls deutlich, dass der Sicherheitsaspekt dabei größtmöglicher Beachtung bedarf. Es muss auch bedacht werden, dass die Datenschutzrichtlinien in Deutschland sehr stark ausgeprägt sind und dies mit Cloud Lösungen rechtlich kollidieren kann.

Auch für den privaten Anwender haben Clouds immer mehr Relevanz. Mehr und mehr Nutzer speichern Daten in Clouds oder nutzen diese ganz unbewusst wie z.B. bei NetFlix. Netflix hat sein Angebot mittlerweile auch in Deutschland stark erweitert.

Diese Beispiele geben nur einen kleinen Eindruck davon, wie sehr sich die Clouds in Zukunft in allen möglichen Bereichen etablieren werden. All diese Entwicklungen machen auch deutlich, wie unerlässlich es ist die Sicherheit von Clouds auf einen angemessenen Standard zu heben.

Quellen

Autoren: Philipp Viertel, André Kaleja

1. <http://www.datacenterknowledge.com/archives/2015/02/23/openstack-vs-cloudstack-the-platforms-and-the-cloud-apis/> (Abgerufen am 30.06.2017).
2. <https://opennebula.org/eucalyptus-cloudstack-openstack-and-opennebula-a-tale-of-two-cloud-models/> (Abgerufen am 30.06.2017).
3. <https://docs.openstack.org/admin-guide/> (Abgerufen am 30.06.2017).
4. <https://docs.openstack.org/ocata/install-guide-ubuntu/InstallGuide.pdf> (Abgerufen am 30.06.2017).
5. <https://docs.openstack.org/security-guide/> (Abgerufen am 30.06.2017).
6. <http://docs.cloudstack.apache.org/en/latest/> (Abgerufen am 30.06.2017).
7. <http://docs.opennebula.org/5.2/index.html> (Abgerufen am 30.06.2017).
8. <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf> (Abgerufen am 02.07.2017)
9. Handbook on Securing Cyber-Physical Critical Infrastructure, Foundations and Challenges Sajal Das, Krishna Kant, Nan Zhang ISBN: 978-0-12-415815-3
10. Security, Privacy and Trust in Cloud Systems Surya Nepal, Mukaddim Pathan ISBN 978-3-642-38586-51010
11. Bio-inspiring Cyber Security and Cloud Services: Trends and Innovations Abould Ella Hassanien, Tai-Hoon Kim, Janusz Kacprzyk, Ali Ismail Awad ISBN 978-3-662-43616-5
12. Zhou M, Zhang R, XieW, QianW, Zhou A (2010) Security and privacy in cloud computing: a survey. In: 6th international conference on semantics knowledge and grid (Ningbo, China, 2010), pp 105–112
13. ZisisD, LekkasD(2012) Addressing cloud computing security issues. FutureGener ComputSys 28(2012):583–592
14. Xiao Z, XiaoY(2012) Security and privacy in cloud computing. IEEECommunSurvTutorials 2012:1–17
15. Rong C, Nguyen ST, Jaatun MG (2012) A survey on security challenges in cloud computing Comput Elect Engi Beyond Lightning
16. KhorshedMT, Ali ABMS, Wasimi SA(2012) A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. Future Gen Comput Sys 28(2012):833–851
17. Gartner (2008) Assessing the security risks of cloud computing. <http://cloud.ctrls.in/files/assessing-the-security-risks.pdf>. White Paper
18. ENISA (2009) Cloud computing: benefits, risks and recommendations for information security. <http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment> . White Paper

19. CSA (2010) Top threats to cloud computing.
<https://cloudsecurityalliance.org/research/topthreats/> . White paper
20. Subashini S, Kavitha V (2011) A survey on security issues in service deliverymodels of cloud computing. J Netw Comput Appl 34(2011):1–11

Software-Architektur

Teilbereich(e): Serverless, Microservices

Projektleiter: Christian Holzberger

Projektteam: Christian Holzberger

Github-Repo: <https://github.com/cHolzberger/2017-Sw-Eng-Projekt.git>