



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MASTER THESIS

Entscheidungsunterstützung für Architekten zur Auswahl von Frontend-Technologien für SAP HANA

vorgelegt von
Jennyfa Jurisch

Jennyfa Jurisch

4jurisch@informatik.uni-hamburg.de

Studiengang Informatik

Matr.-Nr. 6712367

Fachsemester 7

Erstgutachter: Prof. Dr.-Ing. Matthias Riebisch

Zweitgutachter: Dr. Philip Joschko

Betreuer: Sebastian Gerdes

Abgabe: 04.10.2017

Abstract

Im Zuge der Softwareevolution kommt es vermehrt zu Szenarien, in denen eine Technologie ausgetauscht oder um zusätzliche Technologien erweitert werden muss. Dabei kommt es häufig zu Fehlern aufgrund von Abhängigkeiten dieser Technologien. Das Wissen hinter diesen Abhängigkeiten steckt meist implizit in der Architektur der Software und wird nicht dokumentiert.

Daraus resultiert, dass die wenigsten Architekten die gesamte Technologielandschaft ihres Systems kennen und wissen, warum sich für oder gegen eine Technologie entschieden wurde. Diese Arbeit des Technologiewissens kann unterstützt werden, indem Abhängigkeiten und Beziehungen in den Entscheidungsprozess mit einbezogen werden.

Der Fokus der Arbeit liegt daher auf der Analyse der Zusammenhänge zwischen Technologien bestehender Anwendungen in Bezug auf Entwurfsentscheidungen während der Softwareevolution und dem Aufbau eines exemplarischen Technologiewissens. Anhand einer umfassenden Literaturrecherche werden Abhängigkeiten analysiert und klassifiziert. Wodurch beantwortet werden soll, welche Abhängigkeiten aus der Literatur sich in Quellen wie Build-Dateien, Dokumentationen und Foren wiederfinden lassen und wie diese in den Quellen abgebildet werden.

Die Arbeit entwickelt ein Vorgehen zur Identifizierung und Klassifizierung von Technologieabhängigkeiten. Anschließend wird die Klassifizierung mit Hilfe von Experteninterviews verifiziert. Darauf folgt eine exemplarische Analyse verschiedener Frontend-Technologien. Ein Prototyp, der mit diesem Technologiewissen gefüllt ist, zeigt, wie dieses Wissen in der Praxis genutzt werden kann, um Entscheidungen durch Empfehlungen für Technologien zu unterstützen. Die anschließende Bewertung des Prototyps durch Experten zeigt die Relevanz und den Nutzen einer solchen Entscheidungsunterstützung.

Die Ergebnisse der Arbeit und die Resonanzen der Experteninterviews zeigen, dass sich eine derartige Entscheidungsunterstützung als sehr sinnvoll darstellt. Vorwiegend, da Architekten durch ein solches Werkzeug viel Zeit und Kosten sparen können, aber auch weil es eine immense Wissensbasis darstellt. Sollten zukünftige Arbeiten die Thematik weiterausarbeiten, sollte der Fokus auf der automatischen Generierung von Wissen aus den verschiedenen Quellen liegen, aber auch die Verbesserung der Usability betrachtet werden.

Inhaltsverzeichnis

Abstract	I
Inhaltsverzeichnis	III
1 Einleitung	1
1.1 Probleme und Herausforderungen der Softwareevolution	1
1.2 Chancen der Softwareevolution und Zielsetzung der Arbeit	3
2 Stand der Technik	5
2.1 Motive und Ziele der Softwarearchitektur	5
2.1.1 Eigenschaften der Softwarearchitektur	5
2.1.2 Chancen durch Entscheidungsunterstützung	7
2.2 SAP HANA: Ausprägungen und Eigenschaften	10
2.2.1 Ausprägungen von SAP HANA	10
2.2.2 Architektureigenschaften von SAP HANA	11
2.3 Frontend-Technologien	15
3 Abhängigkeiten zwischen Technologien	17
3.1.1 Quellen der Abhängigkeiten	20
3.1.2 Relevante Abhängigkeiten	36
3.1.3 Klassifizierung von Abhängigkeiten	38
3.1.4 Bewertung der Klassifizierung	40
4 Technologieanalyse	43
4.1 Kriterien für Technologiewissen	43
4.2 Selektion expliziter Frontend-Technologien	46
4.3 Eigenschaften der Technologien	46
4.3.1 SAPUI5	47
4.3.2 AngularJs	52
4.3.3 Play	53
5 Prototypische Implementierung	55
5.1 Nutzerszenarien	55
5.2 Architektur der Implementierung	55
5.3 Technische Implementierung	56
6 Feedback und Bewertung	62
7 Fazit und Ausblick	66
7.1 Fazit einer Entscheidungsunterstützung	66
7.2 Zukünftige Relevanz von Entscheidungsunterstützungen für Technologien	67
A Anhang	70
A.1 IT-Markt	70

A.2	INIT Solution GmbH.....	70
A.3	GitHub Projekt	71
A.4	Dokumentationen	71
A.5	Andere Build-Tools	72
A.6	Stack Exchange	73
A.7	Metamodelle	75
A.8	SAPUI5.....	76
A.9	AngularJS.....	79
A.10	Play	80
A.11	Architektur des Prototyps	82
A.12	Lokale Installation.....	83
A.13	Fragebogen zur Bewertung.....	83
	Literaturverzeichnis	92
	Abbildungsverzeichnis	99
	Tabellenverzeichnis	99
	Glossar	100
	Eidesstattliche Versicherung	103

1 Einleitung

Der weltweite Markt für Software und IT-Services wird sich laut einer Untersuchung von 2005 bis 2019 mehr als verdoppeln [Statista, 2017]¹ und mit ihm steigt auch der Anteil an Software in Produkten und Dienstleistungen [Broy et al., 2006]. Damit ist Informatik eine der am schnellsten wachsenden und sich wandelnden Wissenschaften dieser Zeit [Vogel et al., 2009]. Die Tatsache, dass 80 % der Kosten für Softwareprojekte nach der initialen Entwicklung entstehen [Bass, 2012], lässt erahnen, dass Softwarearchitekten in der Softwareevolution vor vielen Herausforderungen stehen.

1.1 Probleme und Herausforderungen der Softwareevolution

Die Softwareevolution beginnt nach der initialen Entwicklung der Software. Die Architektur des Systems ist also bereits entworfen und implementiert worden oder es wurde lediglich ein System implementiert, ohne ein Konzept für die Architektur entworfen zu haben [Vogel et al., 2009]. Doch auch in letzterem Fall hat das System eine Architektur, nur ist diese niemanden bewusst [Bass et al., 2012]. Die Architektur des Systems resultiert dabei aus einer Vielzahl an Entwurfsentscheidungen, die häufig sehr früh getroffen werden müssen [Bass et al., 2003; Soliman u. Riebisch, 2014]. Beispielsweise muss früh entschieden werden, welche Technologien genutzt werden [Klein u. Gorton, 2015].

Viele dieser Entscheidungen bestimmen die Entwicklung des Systems [Bass et al., 2003] oder beeinflussen weitere Entscheidungen [Jansen u. Bosch, 2004], da auf Grundlage der ersten Entscheidungen manche Optionen automatisch ausgeschlossen werden müssen. Häufig fehlt es bei diesem Prozess an einer Dokumentation, dadurch verfügen nur an Entscheidungen beteiligte Personen über das entsprechende Wissen. Wenn diese das Team oder sogar das Unternehmen verlassen, durch Firmenwechsel oder Ruhestand, geht dieses Wissen verloren [Jansen u. Bosch, 2004].

Dies führt zu Problemen und Herausforderungen. Fundamentale Risiken, die sich bei großen Softwareprojekten ergeben, sind die Komplexität des Projektes, die hohen Kosten für Änderungen und die Erosion der Softwarearchitektur [Jansen u. Bosch, 2005]. Um diese Risiken zu vermeiden, müssen die vorhandene Systemarchitektur zunächst rekonstruiert und auch die Auswirkungen von Änderungen erkannt werden

¹Bis 2015 bereits um 82 % gestiegen, siehe Anhang A.1.

[Jansen u. Bosch, 2004]. Wenn hierbei Fehler gemacht werden, kann dieser kostenintensive Prozess noch teurer werden, da die Änderungen frühere Entwurfsentscheidungen verletzen und die Software im schlimmsten Fall nicht mehr funktioniert. Hierbei könnte der Architekt jedoch unterstützt werden, wenn es mehr Werkzeuge geben würde, welche die Entwurfsentscheidungen während der Softwareevolution unterstützen.

Fünf Werkzeuge dieser Art wurden von Jansen u. Bosch [2004] evaluiert. Nur eines davon unterstützt den Architekten bei Entwurfsentscheidungen und nur zwei Werkzeuge können mit Architekturveränderungen umgehen. Insgesamt kommen Jansen u. Bosch zu einem ernüchternden Urteil:

„Evolution of software systems, and their associated software architecture, however, is not equally well supported by the existing state-of-the-art approaches.“ [Jansen u. Bosch, 2004]

Neben dem Verlust des Wissens über Entwurfsentscheidungen besteht eine weitere Herausforderung darin, dass sich viele Entwurfsentscheidungen in der frühen Entwicklung mit Technologien beschäftigen [Klein u. Gorton, 2015; Soliman et al., 2015]. Wurde sich für eine Technologie entschieden, ist es kostspielig und vor allem aufwendig, diese Entwurfsentscheidung zu revidieren [Klein u. Gorton, 2015]. Zudem wirkt sich die Entscheidung aufgrund von Abhängigkeiten zu anderen Technologien oder aufgrund von Inkompatibilitäten auf weitere Entscheidungen aus [Chen u. Xing, 2016].

Schon bei der initialen Entwicklung ist es hierbei schwer für den Architekten, einen Überblick über die gesamte Technologielandschaft zu haben und so die beste Auswahl für das System zu treffen [Chen u. Xing, 2016]. Aufgrund ihrer Erfahrung mit der Verwendung bestimmter Technologien bevorzugen Architekten diese häufig in ihren Projekten. Wenn sie jedoch andere Technologien nutzen müssen, holen sie sich häufig Rat bei anderen Experten [Chen u. Xing, 2016], wie etwa in Onlineforen, oder sie bauen selbst Wissen auf durch das Lesen von Dokumentationen der Technologien [Chen u. Xing, 2016]. Dieser Prozess braucht allerdings Zeit und führt damit zu Kosten für das Projekt. Zudem sind Beiträge in Foren und auch Artikel von Experten häufig nicht objektiv, da sie meinungsbasiert sind [Chen u. Xing, 2016].

Bei der Softwareevolution erschwert sich dieser Prozess der Technologieauswahl zu-

sätzlich, da häufig nicht genau bekannt ist, welche Technologien verwendet bzw. warum sie ausgewählt wurden. Auch hierbei fehlen dem Architekten Informationen zu den Entscheidungen beim Architekturentwurf. Hinzu kommt, dass Technologien oft ausgetauscht werden müssen, unter anderem weil:

- sich die Anforderungen an das System geändert haben und diese durch die Technologie nicht erfüllt werden können;
- die Technologie vom Entwickler nicht mehr aktualisiert wird,
 - wodurch Sicherheitsrisiken entstehen können, wenn Sicherheitsupdates fehlen,
 - und Fehler nicht behoben werden;
- es keinen Support mehr gibt, um Probleme schnell zu beseitigen;
- sich die Kosten für die Technologie erhöht haben;
- die weiteren Abhängigkeiten einer Technologie aus den genannten Gründen nicht eingehalten werden können und die Technologie so auch nicht weiter genutzt werden kann.

Hierzu muss der Architekt zunächst Alternativen zu diesen Technologien recherchieren und verstehen. Um den Architekten bei dieser Aufgabe zu unterstützen, sollten Abhängigkeiten von Technologien zeitsparend aufzufinden sein und Alternativen zu diesen Technologien aufgezeigt werden.

1.2 Chancen der Softwareevolution und Zielsetzung der Arbeit

Ein Werkzeug, welches den Architekten bei den genannten Herausforderungen während der Softwareevolution unterstützt, indem es aufzeigt, welche Technologien voneinander abhängig sind und wie diese durch andere Technologien ersetzt werden könnten, würde eine Menge Zeit und somit auch Kosten sparen.

Aus diesen Gründen liegt der Fokus dieser Arbeit auf der Analyse der Zusammenhänge zwischen Technologien bestehender Anwendungen in Bezug auf Entwurfsentscheidungen während der Softwareevolution. Dazu sind exemplarisch Informationen zu Technologien, wie Vor- und Randbedingungen und Beziehungen zu anderen Technologien, sowie Alternativen zu ermitteln. Das erfasste Technologiewissen soll für die Entscheidungsunterstützung, etwa beim Technologieaustausch, genutzt werden. Dadurch soll ein Prototyp eines Werkzeuges entwickelt werden, welches Architekten bei den Herausforderungen der Technologieauswahl unterstützt. Die vorliegende Arbeit grenzt sich von anderen Arbeiten, wie etwa von Kruchten et al. [2004] oder Zimmermann et al. [2009], ab, da speziell Abhängigkeiten von Technologien untersucht werden und nicht Entwurfsentscheidungen im Allgemeinen.

Für die Analyse der Zusammenhänge zwischen Technologien werden wissenschaftliche Arbeiten auf die verschiedenen Arten von Abhängigkeiten untersucht. Diese werden mit Technologieabhängigkeiten aus anderen Quellen verglichen. Dadurch soll beantwortet werden, welche Abhängigkeiten aus der Literatur sich auch in der Praxis wiederfinden und wie diese Abhängigkeiten in den verschiedenen Quellen abgebildet werden. Durch diese Analyse soll eine Klassifizierung von Technologieabhängigkeiten entstehen, welche dem Architekten dabei hilft, Entwurfsentscheidungen zu treffen.

Anhand dieser Klassifizierung soll dann exemplarisch für Frontend-Technologien, welche mit SAP HANA verwendet werden können, ein Technologiewissen erstellt werden, welches die Vor- und Randbedingungen, Beziehungen sowie Funktionalitäten (nachfolgend Features genannt) beinhaltet. Mithilfe dieses exemplarischen Technologiewissens wird dann der Prototyp entwickelt, welcher den Architekten bei Entwurfsentscheidungen unterstützen soll. Um den Nutzen des exemplarischen Technologiewissens sicherzustellen, wird der Prototyp von Entwicklern der Firma INIT Solution GmbH² bewertet.

Die Arbeit ist folgendermaßen aufgebaut: In Kapitel 2 werden zunächst die Grundlagen und der derzeitige Stand der Technik dargestellt. Daran anknüpfend werden in Kapitel 3 mithilfe verschiedener Quellen Technologieabhängigkeiten analysiert und klassifiziert. Mittels dieser Klassifizierung werden relevante Abhängigkeiten für das Technologiewissen spezifiziert und die selektierten Frontend-Technologien daraufhin untersucht.

Den Kern des Wissensaufbaus bildet Kapitel 4, in dem anhand von verschiedenen Kriterien exemplarisch Frontend-Technologien untersucht werden, um später Empfehlungen für deren Auswahl geben zu können. Die Umsetzung des Technologiewissens als prototypisches Web-Werkzeug in Kapitel 5 zeigt exemplarisch, wie dieses Technologiewissen einen Architekten bei seinen Entscheidungen unterstützen kann und wie das Werkzeug benutzt werden kann. Daraufhin wird in Kapitel 6 eine Evaluierung zu dem prototypischen Web-Werkzeug beschrieben. Dazu werden Entwickler anhand eines Anwendungsfalls das Werkzeug nutzen und bewerten.

Kapitel 7 fasst die Ergebnisse der Arbeit zusammen, zeigt deren Stärken sowie Schwächen auf und stellt mögliche zukünftige Ausarbeitungen dieser Art dar.

² Weitere Informationen zur INIT Solution GmbH im Anhang A.2

2 Stand der Technik

Um eine Entscheidungsunterstützung für einen Architekten im Bereich der Auswahl von Technologien zu entwickeln, müssen zunächst die Grundlagen der Softwarearchitektur erläutert werden und die Begrifflichkeiten für den exemplarischen Bereich der Frontend-Technologien für SAP HANA. Nachfolgend werden hierfür die Motive und Ziele der Softwarearchitektur aufgezeigt.

2.1 Motive und Ziele der Softwarearchitektur

Es sollen zunächst die Definitionen und die Eigenschaften für Softwarearchitektur dargestellt werden um darauf aufbauend die Chancen durch eine Entscheidungsunterstützung für Architekten auf zu zeigen.

2.1.1 Eigenschaften der Softwarearchitektur

„Software architecture is a growing but still young discipline; hence, it has no single, accepted definition.“ [Bass et al., 2012]

Obgleich der Begriff der Softwarearchitektur nur schwer allgemeingültig definiert werden kann, existieren einige vielversprechende Definitionsansätze. Für Shaw u. Garlan [1996] beinhaltet Softwarearchitektur die Beschreibung von Komponenten, aus denen sich ein System bildet, von Wechselwirkungen zwischen diesen Komponenten sowie von deren Mustern und den Bedingungen für diese Muster. Ein solches System kann auch in einem größeren System wieder genutzt werden.

Bass et al. [2012] definieren die Softwarearchitektur eines Computersystems als die Struktur bzw. die Menge an Strukturen, die Softwareelemente sowie die sichtbaren Eigenschaften und die Beziehungen untereinander. Das Institute of Electrical and Electronics Engineers (IEEE) berücksichtigt in seiner Definition zudem auch die Umwelt als einen zentralen Punkt der Softwarearchitektur:

„The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.“ [IEEE, 2007]

Vogel et al. betrachten Softwarearchitektur als ein Zusammenspiel der beiden Definitionen von Bass und des IEEE:1471-Standards. Denn wie für Bass sind für Vogel et al. [2009] sind die wichtigsten Punkte einer Softwarearchitektur:

-
- Software-Struktur bzw. die Software-Strukturen eines Systems
 - Softwarebausteine eines Systems
 - Eigenschaften der Softwarebausteine eines Systems
 - Beziehung zwischen den Softwarebausteinen eines Systems

Anhand dieser Definitionen ist zu erkennen, dass eine knappe und allgemeingültige Aussage über Softwarearchitektur nur schwer herauszuarbeiten ist. Dies liegt vor allem daran, dass die Softwarearchitektur eine umfangreiche Disziplin ist, die viele verschiedene Aktivitäten umfasst; so müssen etwa der Business-Case und die Anforderungen des Systems verstanden werden. Die Anforderungen werden nicht nur durch technische und organisatorische Faktoren bestimmt, auch die Stakeholder haben Einfluss darauf.

Die geschaffene oder auch ausgewählte Architektur muss an die Stakeholder, vor allem an die Entwickler, kommuniziert und auch dokumentiert werden. Die Entwickler sind hierbei besonders wichtig, da sie das Systems anhand dieser Architektur implementieren müssen. Neben den Anforderungen der Stakeholder muss eine Softwarearchitektur auch Qualitätsmerkmale erfüllen, welche entsprechend der Norm ISO/IEC FCD 25010 geregelt sind [Bass et al., 2012]. Speziell diese komplexen Anforderungen und der Druck, diese schneller und kostengünstiger bei gleichbleibend hoher Software-Qualität umzusetzen, macht Softwarearchitektur in den letzten Jahren immer wichtiger [Vogel et al., 2009].

Bei manchen Projekten entsteht die Softwarearchitektur auch ungeplant. Die Anforderungen werden implementiert und im Laufe der Zeit entsteht eine nicht näher bekannte Softwarearchitektur, was zur erheblichen Problemen führen kann. Da dem Architekten der Gesamtüberblick fehlt und das Projekt zu komplex wird, entstehen unvorhersehbare Risiken. Dadurch wird die Wartbarkeit eingeschränkt und die Performance verschlechtert sich, weshalb auch Änderungen länger brauchen, um umgesetzt zu werden [Vogel et al., 2009]. Je größer und komplexer die Projekte sind, desto wichtiger wird eine Softwarearchitektur, um diese Risiken zu vermeiden [Shaw u. Garlan, 1996].

Die Softwarearchitektur wird zudem stark von der Erfahrung des Architekten beeinflusst. Dieser muss die Chancen und Risiken der Architektur identifizieren. Einige der großen Herausforderungen der Softwarearchitektur sind die hohen Kosten für Änderungen und die Komplexität, die sich während der Evolution häufig erhöht [Jansen u. Bosch, 2005].

Entscheidungen während der Entwicklung charakterisieren die Architektur. Diese **Entwurfsentscheidungen** beeinflussen die Zukunft eines Projektes und die damit verbundene Softwareevolution. Die Erosion ist zum Teil auf den Wissensverlust zurückzuführen, der durch nicht dokumentierte Entscheidungen entsteht, da diese Entscheidungen implizit in die Architektur eingebettet sind [Jansen u. Bosch, 2005].

2.1.2 Chancen durch Entscheidungsunterstützung

„Software architecture is about making design decisions which potentially impose risk on product success.“ [Soliman et al., 2016].

Der im vorherigen Kapitel beschriebene Wissensverlust durch nicht dokumentierte Designentscheidungen zeigt die Bedeutsamkeit von Entwurfsentscheidungen. Auch für Bass et al. [2012] ist Softwarearchitektur das Ergebnis aus einer Menge an geschäftlichen und technischen Entscheidungen. Entscheidungen wiederum sind das Resultat aus dem Prozess der Architektur-Entwicklung oder auch der Softwareevolution [Gerdes et al., 2014].

Jansen u. Bosch definieren Architekturentscheidungen als:

„A description of the set of architectural additions, subtractions and modifications to the software architecture, the rationale, and the design rules, design constraints and additional requirements that (partially) realize one or more requirements on a given architecture“. [Jansen u. Bosch, 2005]

Werden diese Entwurfsentscheidungen jedoch nur unzureichend dokumentiert, sind Entscheidungsverletzungen möglich. Aufgrund von veralteten Entscheidungen, welche nicht entfernt wurden, können ein erhöhter Änderungsaufwand und infolgedessen hohe Kosten entstehen [Shaw u. Clements, 2006]. Dies bedeutet ebenfalls hohe Wartungskosten für ein Projekt und die Software droht zu erodieren [Jansen u. Bosch, 2006].

Durch diese Entwurfsentscheidungen werden auch weitere Entscheidungen beeinflusst. Der Einsatz einer Technologie beeinflusst das Projekt beispielsweise insofern, als andere Technologien nicht mehr eingesetzt werden können oder aber genutzt werden müssen; Entscheidungen, die weitere Entscheidungen beeinflussen, nennt man

Constraints [Kruchten, 2004]. Dadurch bedingen die bereits getroffenen Entwurfsentscheidungen und die Anforderungen eines Systems die weiteren Entscheidungen.

Der Prozess der Architekturentscheidung während der Softwareevolution wird allerdings kaum unterstützt, obwohl er durch Entscheidungsunterstützungen verbessert werden könnte. Dies ist, wie bereits erwähnt, auch wirtschaftlich wichtig, Neben zeitlichen Ersparnissen können durch geeignete Werkzeuge immense Kosten eingespart werden [Jansen u. Bosch, 2004].

Der Einsatz von externen Technologien in Projekten spart häufig viel Zeit und Kosten, da neue Features genutzt werden können, ohne diese von Grund auf neu zu implementieren. Allerdings haben sie auch Nachteile – neben den Anschaffungskosten und der geringen Anpassbarkeit einiger Features erschweren diese Technologien durch ihre Abhängigkeiten untereinander häufig die Entscheidungen der Softwarearchitekten. Nur durch kompetentes Verständnis darüber können die richtigen Entscheidungen getroffen werden. Die Technologien entwickeln sich jedoch so schnell, dass viele Architekten sich auf eine Technologie fokussieren und bei anderen Technologien Unterstützung benötigen [Chen u. Xing, 2016]. Mit der Zeit ist es jedoch durch neue Anforderungen nötig, Technologien auszutauschen oder auch zu ergänzen. Dies bedeutet für die Evolution, dass die getroffenen Entscheidungen und Constraints immer wieder überarbeitet werden müssen.

Derzeit werden Architekten bei Entwurfsentscheidungen in Bezug auf Abhängigkeiten zwischen einzelnen Technologien nur unzureichend durch Werkzeuge unterstützt. Neben den Werkzeugen, die von Jansen u. Bosch [2004] evaluiert wurden, existieren Werkzeuge, die Architekten bei dem Auffinden von Abhängigkeiten in ihrem Projekt unterstützen sollen. Einige Werkzeuge überprüfen Abhängigkeiten in Projekten, die mit dem Node Package Manager (npm) arbeiten. Eines der Werkzeuge kontrolliert, ob die Abhängigkeiten, welche in einer package.json mit ihrer Version spezifiziert sind, auch in dieser speziellen Version installiert wurden [Gołębiowski, 2014]. Ein weiteres von Maxogden [2017] überprüft, ob alle benutzten Module im Projektcode auch in der package.json definiert wurden und ob die definierten Abhängigkeiten auch benutzt wurden.

Aufgrund der häufigen Verwendung von Drittanbieter-Bibliotheken scannt das Werkzeug von Long [2017] Anwendungen und ihre abhängigen Bibliotheken, um darin enthaltene bekannte Schwachstellen zu identifizieren. Des Weiteren beschäftigen sich auch Build-Werkzeuge (siehe 3.1.1) mit Abhängigkeiten, jedoch muss der Entwickler

bei diesen Werkzeugen bereits wissen, welche Abhängigkeiten er benötigt, und diese definieren. Nur dann werden die Abhängigkeiten heruntergeladen.

Da jedoch noch kein geeignetes Werkzeug zur Entscheidungsunterstützung existiert, treffen Architekten ihre Entscheidungen vorwiegend auf Grundlage ihrer bisherigen Erfahrungen [Soliman u. Riebisch, 2014], der Herstellerangaben einer Technologie sowie deren Dokumentationen und der Beiträge von Internetforen wie *Stack Overflow* [Chen u. Xing, 2016].

Während der Softwareevolution muss ebenfalls bekannt sein, weshalb eine Technologie ausgewählt wurde und ob dieser Grund weiterhin besteht oder diese Entscheidung möglicherweise bereits obsolet ist [Gerdes et al., 2014; Jansen u. Bosch, 2005]. Wenn neue Technologien ausgewählt werden müssen, sei es für das initiale Design oder während der Softwareevolution, so müssen immer Alternativen miteinander verglichen werden. Diese Alternativen müssen zunächst recherchiert werden, um sie anschließend anhand der Qualitätsanforderungen, der benötigten Features und auch der Kosten miteinander vergleichen zu können [Gorton et al., 2015].

Doch die Gewinnung verschiedener Alternativen wird in den meisten Ansätzen zur Entscheidungsfindung nicht unterstützt; es wird immer mit einer Anzahl an Alternativen vorausgesetzt. Dies ist jedoch unrealistisch [Simon, 2002]. Daraus resultiert, dass eine Entscheidungsunterstützung bereits verschiedene Alternativen enthalten sollte, um den Architekten optimal zu unterstützen.

Nach Kruchten [2004] können Designentscheidungen in drei Hauptkategorien eingeordnet werden:

1. *Existence decisions (Existenzentscheidungen)*
besagen, dass ein Element oder Artefakt in dem System vorhanden sein wird. Im Gegensatz dazu stehen die *Ban decisions (Verbotsentscheidungen)* oder *Non-existence decisions (Nicht-Existenzentscheidung)*, bei denen das Element nicht im System existiert.
2. *Property decisions (Eigenschaftsentscheidungen)*
betreffen dauerhafte und übergreifende Eigenschaft oder Qualität des Systems. Dies können Design-Richtlinien oder auch Beschränkungen sein.
3. *Executive decisions (prozessbezogene Entscheidungen)*
sind leitende Entscheidungen, die vom Geschäftsumfeld getrieben werden und die Entwicklung, die Menschen, die Organisation und weitestgehend die Auswahl von Technologien und Werkzeugen beeinflussen.

Ungefähr 25 % aller Systementscheidungen sind prozessbezogene Entscheidungen, und die meisten davon sind Technologieentscheidungen [Miesbauer u. Weinreich, 2013; Drira, 2013]. Entscheidungsunterstützungen für Technologien helfen demzufolge im Bereich der prozessbezogenen Entscheidungen. Dies ist unter anderem deshalb wichtig, da diese Entscheidungen die Existenz- und Eigenschaftsentscheidungen zum Teil erfassen und erzwingen [Kruchten et al., 2006] und sich Entscheidungen auch gegenseitig beeinflussen können [Jansen u. Bosch, 2005].

Daraus resultiert, dass für eine effektive Entscheidungsunterstützung die nachfolgenden Aspekte wichtig sind:

- Der Architekt wird unterstützt, um eine Entscheidung treffen zu können.
- Dazu müssen bei Alternativen deren Abhängigkeiten, Features sowie Vor- und Nachteile dargestellt werden.
- Auch bei großem Wissensverlust, durch unzureichende Dokumentationen zur Architektur und implizitem Wissen in der Architektur [Zimmermann et al., 2009] muss es möglich sein, eine Entscheidung zu treffen.

2.2 SAP HANA: Ausprägungen und Eigenschaften

Da die Entscheidungsunterstützung für Architekten am Beispiel von Frontend-Technologien für SAP HANA dargestellt werden soll, werden nachfolgend zunächst die Grundlagen zu SAP HANA erläutert.

2.2.1 Ausprägungen von SAP HANA

Es wird immer wichtiger für Unternehmen, große Datenmengen, Big Data, in Echtzeit auswerten zu können, um etwa Erkenntnisse über das Verhalten von Kunden gewinnen zu können. Um diese riesen Datenmengen zu verarbeiten, hat SAP jahrelang verschiedene Technologien entwickelt.

Die firmeneigene Suchmaschine TREX war der erste Vorstoß in Richtung SAP-HANA-Datenbank, es folgten weitere wie SAP liveCache, die eine hybride Hauptspeicherdatenbank, die auf der relationalen SAP-Datenbank MaxDB basiert [Berg et al., 2014].

2010, nach einigen Jahren Entwicklung, folgte die SAP-HANA-Datenbank, eine komplette In-Memory-Lösung, basierend auf SAP TREX und SAP BIA [Färber et al., 2004]. Zwar gibt es bereits seit 1984 In-Memory-Datenbanken, doch waren diese für große

Systeme nicht bezahlbar [Neumann u. Schicker, 2014; Große Bley et al., 2012]. Erst die Reduzierung der Kosten für Speichersysteme und die stetige Erhöhung von deren Kapazitäten ermöglichten den Einsatz von In-Memory-Datenbanken. Diese Lösung mit der Fähigkeit des Massive Parallel Processing (MPP) macht es möglich, große Mengen von Daten gleichzeitig zu verarbeiten [Adlon, 2015].

Mittlerweile besteht SAP HANA aus weit mehr als nur einer Datenbank. Seit 2013 entwickelte sich SAP HANA immer mehr zu einer Plattform, bei der die Datenbanktechnologie im Zentrum steht. Seit 2015 ist SAP S/4 HANA eine Komplettlösung auf Basis von SAP HANA [Schmitz, 2015].

Mit SAP HANA Studio lässt sich das gesamte HANA-System administrieren. Es stellt damit eine zentrale Entwicklungsumgebung für die Erstellung von Datenmodellen sowie ein Administrations- und Monitoring-Werkzeug dar.

Bei SAP HANA handelt es sich deshalb um weit mehr als die SAP-HANA-Datenbank. Da die Vermittlung jeglicher Produkte rund um SAP HANA nicht für die Analyse der Frontend-Technologien und die verschiedenen Typen von Abhängigkeiten relevant ist, wird jedoch der Fokus der Arbeit für die weitere Ausarbeitung auf die SAP-HANA-Datenbank gelegt.

2.2.2 Architektureigenschaften von SAP HANA

Die Architektur wurde zusammen mit dem Hasso-Plattner-Institut entwickelt, weshalb HANA auch zunächst für „*Hasso’s New Architecture*“ stand. Mittlerweile steht SAP HANA jedoch für „*High-Performance Analytic Appliance*“ [tutorial_hana, 2017]. Es ist eine vollständige In-Memory-Lösung, mit welcher große Datenmengen verarbeitet werden können, ohne komplexe physische Datenmodelle erstellen zu müssen. In-Memory-Datenbanken setzen für die schnellere Verarbeitung darauf, die Daten in den RAM zu laden und Abfragen so aus dem Memory zu beantworten.

SAP HANA Architektur besteht, wie in Abbildung 1 zu erkennen, aus verschiedenen Diensten (Servern):

- einem Indexserver
- einem Preprocessorserver
- einem Nameserver
- einer XS Engine
- einem Statistik-Server

In-Memory-Computing Engine

Das Herzstück ist der Indexserver, siehe Abbildung 1, oder auch die In-Memory-Computing Engine (IMCE) [Uthaman, 2016]. Die IMCE beinhaltet die aktuellen Daten und die Engines für die Verwaltung von Datenzugriff und -speicherung.

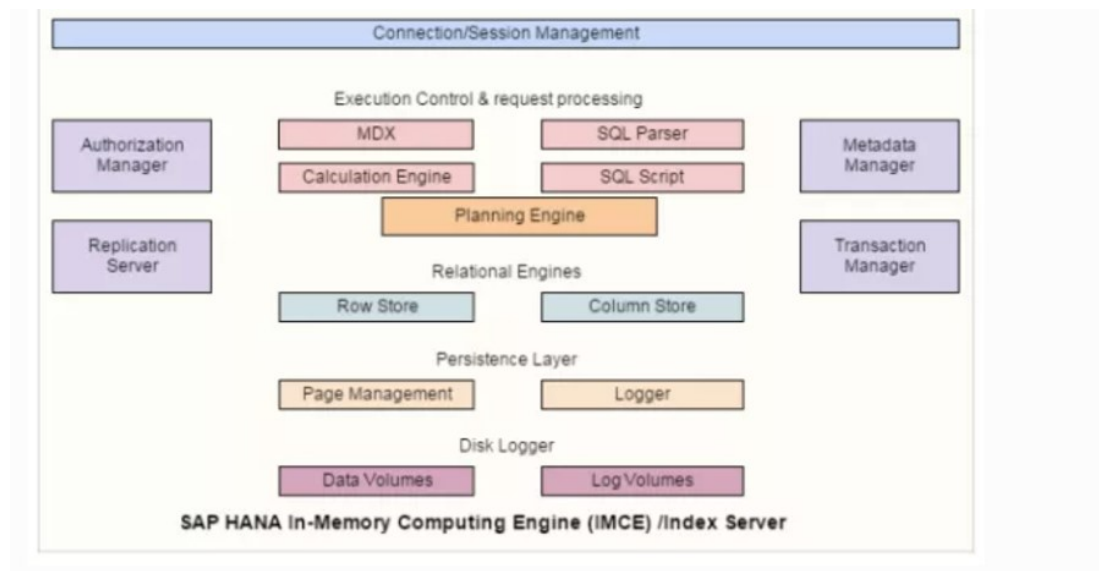


Abbildung 1: SAP-HANA-Datenbank-Architektur [Uthaman, 2016]

Connection/Session Management

Für die Anmeldung in SAP HANA steht das Connection-/Session-Management zur Verfügung, welches Sessions generiert und verwaltet. Für die Kommunikation werden SQL-Statements verwendet [Uthaman, 2016].

Der **Autorisierungs-Manager** arbeitet zusammen mit dem Session-Manager und überwacht, wer auf welche Daten zugreifen darf. Auch der **Metadaten-Manager** und der **Transaktions-Manger** arbeiten mit dem Session-Manager zusammen, ebenso wie der **Replication Server**, welcher die Daten vom Quellsystem repliziert [sapstudent, 2015].

Execution Controll & Request Processing

SQLScript, Multidimensional Expressions (MDX), SQL-Parser, eine effiziente Kalkulations-Engine und eine Planungs-Engine gehören zu einer wichtigen Komponente der IMCE – der Ausführungskontrolle und Anforderungsbearbeitung. Hier werden die Anfragen von Applikationen und Clients zu den richtigen Sub-Komponenten geleitet [tutorial_hana, 2017].

SAP HANA unterstützt zur Datenübertragung die Datenbanksprache SQL mit Hilfe von JDBC (Java Database Connectivity) und ODBC (Open Database Connectivity) und des

Weiteren MDX (Multidimensional Expressions) und die Laufzeitbibliothek SQLDBC [sapstudent, 2015].

Relational Engines

Die zeilen- und spaltenbasierte Speicherung wird durch relationale Engines verwaltet. Welche der beiden Speicher, oder welche Art der Kombination, verwendet werden soll, ist von dem Inhalt und dem System abhängig, welches implementiert werden soll [Berg et al., 2014]. Zudem kann zwischen den Variationen gewechselt werden, um Abfrageausdrucke mit Tabellen beider Layouts zu ermöglichen [Fäber, 2004].

Persistenzschicht & Disk Logger

Um eine persistente Datenerhaltung zu schaffen, enthält die IMCE eine Persistenzschicht, die das Disaster Recovery sicherstellt [tutorial_hana, 2017]. Dies sorgt für eine dauerhafte Speicherung auch nach einem Stromausfall. Dafür verwaltet die Persistenzschicht die Logs für alle Transaktionen. Der Hauptspeicher ist in Seiten unterteilt, welche markiert werden, sobald entsprechende Daten in einer Transaktion geändert werden. In regelmäßigen Abständen werden die Seiten in einen nichtflüchtigen Speicher geschrieben, wie etwa eine Festplatte. Zudem werden sämtliche Änderungen an Transaktionen im Datenbankprotokoll festgehalten, wodurch bei jeder ausgeführten Transaktion ein Protokolleintrag in den persistenten Speicher geschrieben wird.

Bei SAP HANA können geänderte Seiten in Savepoints gespeichert werden, die standardmäßig im Fünf-Minuten-Takt asynchron in den nichtflüchtigen Speicher geschrieben werden [Adlon, 2015]. Eine Transaktion wird erst zurückgeführt, wenn der synchron geschriebene Protokolleintrag in den persistenten Speicher geschrieben wurde. Durch diesen Vorgang wird die Dauerhaftigkeit der Transaktionen gewährleistet, um das **ACID-Prinzip** einzuhalten [Berg et al., 2014].

Das **ACID-Prinzip** muss eingehalten werden, damit die Verarbeitung von Daten einer Datenbank bestimmte Eigenschaften besitzen. So müssen alle Transaktionen *atomar* sein, dies bedeutet, dass sobald ein Teil einer Transaktion fehlschlägt, die gesamte Transaktion fehlschlägt und der Datenbank-Status unverändert bleibt. Zudem dürfen nur gültige Daten abgelegt werden, um die *Konsistenz* zu wahren. Verletzen Transaktionen diese Regeln, wird die gesamte Transaktion rückgängig gemacht, um die Datenbank in einem Status zu sichern, der den Regeln entspricht. Zu diesen Regeln gehört auch, dass eine Transaktion immer *isoliert* werden muss, um Störungen zwischen einzelnen Transaktionen zu verhindern, und dass Transaktionen *dauerhaft* sind [Berg et al., 2014].

Preprocessor-Server

Der Indexserver, oder auch IMCE, nutzt den Preprocessor-Server für die Textdaten-Analyse sowie zum Extrahieren von Informationen aus diesen Daten [tutorial_hana, 2017].

Name-Server

Die Topologie des SAP-HANA-Systems, also die gesamte Systemlandschaft, enthält der Name-Server. Er enthält darüber hinaus alle notwendigen Informationen zu laufenden Komponenten.

XS-Engine

Mithilfe des XS-Clients erleichtert die XS-Engine Java- und HTML-basierten Anwendungen den Zugriff auf das HANA-System.

Statistik-Server

Durch den Statistik-Server werden alle Komponenten auf ihre Konstitution überprüft und er ist verantwortlich für das Sammeln von Daten wie Systemressourcen, Angaben zu deren Verbrauch und der Gesamtleistung des Systems. Durch seine Analysen erhebt er auch historische Daten, um Performance-Probleme zu überprüfen.

SAP HANA setzt im Unterschied zu anderen Datenbanken auf eine hybride In-Memory-Datenbank-Lösung, bei der zeilen-, spalten- und objektbasierte Datenbanktechnologien kombiniert werden [Berg et al., 2014]. Dadurch ist es SAP HANA möglich, Daten in fünf Nanosekunden zu lesen statt in üblichen fünf Millisekunden [Tutorialspoint_hana, 2017]. SAP HANA ist dementsprechend 1 000-mal schneller ist.

Für diese Performance-Verbesserung nutzt SAP HANA unter anderem eine spaltenbasierte Organisation. Deren Vorteil ist, dass nicht alle Daten gelesen werden müssen, sondern nur die relevanten Spalten des Auswahlprozesses. Zudem ist es möglich, jede Spalte als Index oder Schlüssel für den Datenabruf zu nutzen. Der Nachteil liegt allerdings in der Aktualisierung. Hierbei muss zunächst die richtige Spalte und dann die richtige Zeile gefunden werden, was die Effizienz des Schreibprozesses mindert. Dieses Problem behebt SAP HANA, indem nicht aktualisiert wird, sondern eingefügt. Bei einer Aktualisierung wird also eine neue Zeile mit den entsprechenden Informationen eingefügt [Berg et al., 2014].

Diese Methodik ist vor allem für den Bereich des Reportings, bei dem meist nur einzelne Spalten einer Zeile genutzt werden, von Vorteil. Das Lesen von nicht relevanten Daten wird reduziert bis eliminiert und der Prozess wird dadurch wesentlich schneller und effizienter.

Die Idee für die spaltenbasierten Indizes stammt daher, dass Daten in Tabellen sich wiederholen und Muster bilden. Durch die Indexerstellung kann so die Datenmenge reduziert werden. SAP HANA verwendet eine *light-weight*-Komprimierung, die Dictionary-Komprimierung. Dabei wird für jeden Wert eine eindeutige ID erzeugt, welche komprimiert in einer weiteren Tabelle gespeichert wird. Dies erhöht die Performanz, da eine Abfrage nun zur Folge hat, dass die komprimierten IDs gelesen werden [Neumann u. Schicker, 2014].

Die Datenredundanzen in einer Spalte können so bereinigt werden, was die Komprimierungsrate und somit auch die Performanz erhöht. Den Speicher möglichst gering zu halten ist auch deshalb nötig, weil die Daten im Hauptspeicher gehalten werden.

2.3 Frontend-Technologien

Frontend bezeichnet die Oberfläche eines Systems. Ein Frontend ist dementsprechend das, was der Nutzer sieht. Jede Software hat ein Frontend; es ist der Teil, über den der Nutzer seine Eingaben macht [Hery-Moßmann, 2016].

Frontend-Technologien sind also Technologien, die eine Oberfläche erstellen und dabei von einer Backend-Technologie indirekt unterstützt werden, indem diese beispielsweise mit benötigten Ressourcen kommuniziert.

Dabei kann sich das Frontend lokal auf dem Rechner befinden oder auch als Web-Frontend vorliegen, welches im Browser aufgebaut wird. Im Bereich von SAP sind die ersten Web-Frontends im SAP GUI mit WebDynpro ABAP oder Business Services Page implementiert worden [Antolovic, 2016].

Mit SAP Screen Personas können rollenbasierte Benutzeroberflächen erstellt werden. Wirklich angekommen in der Webentwicklung ist SAP allerdings erst mit SAP UI Development Toolkit for HTML5, kurz SAPUI5.

Das Framework SAPUI5, welches auf der Open-Source-Variante OpenUI5 aufbaut, ist gezielt für Weboberflächen entwickelt worden. Im selben Atemzug mit SAPUI5 wird häufig auch SAP Fiori genannt, welches allerdings aus der Technologie SAPUI5 und sogenannten Styling Guidelines besteht. SAP Fiori ist also keine alleinstehende Frontend-Technologie.

Durch die Benutzung von JavaScript mit SAPUI5 ergeben sich Entwicklern, die vorher nur mit SAP-Standards gearbeitet haben, völlig neue Möglichkeiten [referenz, 2015]. Um Daten aus dem Backend zu laden und diese auf der Oberfläche anzuzeigen, nutzt SAPUI5 standardmäßig OData-Services.

3 Abhängigkeiten zwischen Technologien

In Softwareprojekten werden häufig externe Technologien eingesetzt, um deren Features zu nutzen und dadurch Zeit und Kosten zu sparen. Die eingesetzten Technologien besitzen jedoch meist Abhängigkeiten zu anderen Technologien, welche abermals weitere Abhängigkeiten besitzen. Wenn nun etwa neue Anforderungen an die Software gestellt werden, kann es unter Umständen nötig sein, eine der eingesetzten Technologien durch eine andere auszutauschen. Auch andere Szenarien können zu einem Technologieaustausch führen, wie die folgenden:

- Die Technologie ist nicht mehr vorhanden und muss deshalb ausgetauscht werden.
- Die Technologie wird vom Entwickler nicht mehr aktualisiert,
 - dadurch entstehen Sicherheitsrisiken,
 - wodurch Bugs nicht mehr behoben werden.
- Es existiert kein Support mehr für die Technologie, um Probleme schnell beheben zu können.
- Die Lizenzkosten erhöhen sich oder es werden erstmals Kosten erhoben.
- Die Anforderungen an die Software haben sich geändert und diese können nicht durch die derzeitige Technologie abgedeckt werden.
- Die weiteren Abhängigkeiten der Technologie können nicht mehr eingehalten werden.
- Die Vorbedingungen haben sich durch einen Plattform- und/oder Betriebssystemwechsel geändert und können nicht mehr erfüllt werden.
- Die Technologie ist nicht kompatibel mit einer Technologie, die eingesetzt werden muss.
- Geforderte Softwarequalitäten werden nicht (mehr) erfüllt, da
 - diese erst nach der Entwicklung bewertet werden konnten (beispielsweise dauert der Datenzugriff zu lange),
 - die Software zu viel Ressourcen verbraucht,
 - mehr Benutzer auf die Software zugreifen und die Performance darunter leidet.

Diese Szenarien führen in vielen Fällen zu einem Austausch der Technologie; in manchen ist es jedoch auch möglich, die vorhandene Technologie zu erweitern, um die neuen Anforderungen abzudecken. Sowohl ein Technologieaustausch als auch eine Erweiterung kann schnell zu Problemen führen, wie etwa zu einem erhöhten War-

tungsaufwand, Fehlern in der Software bis hin zu einem völlig funktionsunfähigen System. Diese Probleme entstehen vorwiegend, weil das Wissen zu den eingesetzten Technologien nicht vollständig bzw. gar nicht vorhanden ist.

Um solche Risiken zu vermeiden, wäre es wichtig, dass ein Architekt ein gutes Verständnis der Technologielandschaft hat, um die richtige Auswahl für das Projekt zu treffen. Er muss die Abhängigkeiten zwischen den einzelnen Technologien kennen und wissen, welche Features der Technologien für welche Zwecke benutzt werden. Zudem müssen Vorbedingungen bekannt sein, beispielsweise ob einzelne Technologien nur auf bestimmten Servern laufen oder ein bestimmtes Betriebssystem erfordern. Des Weiteren müssen ihm Abhängigkeiten bekannt sein, die andere Technologien ausschließen, da diese nicht kompatibel sind. Doch Softwarearchitekten kennen sich in den meisten Fällen nicht mit einer Vielzahl an Technologien aus. Sie kennen vielmehr einzelne spezielle Technologien und nutzen diese häufig immer wieder in verschiedenen Projekten [Chen u. Xing, 2016].

Wie in Kapitel 2.1.2 beschrieben, existieren bereits Werkzeuge, die Architekten bei dem Auffinden von Abhängigkeiten in ihrem Projekt unterstützen sollen. Dies zeigt, wie wichtig Abhängigkeiten in der Softwareentwicklung sind. Hierbei werden allerdings nur vorhandene Anwendungen darauf untersucht, ob die explizit genutzten Abhängigkeiten zu Modulen vorhanden sind, ob diese noch in der richtigen Version heruntergeladen werden müssen bzw. ob die vorhandenen Abhängigkeiten im Code auch wirklich genutzt werden oder ob es bekannte Schwachstellen in den verwendeten Abhängigkeiten gibt.

Allerdings geben dieser Werkzeuge dem Architekten keine Kenntnis darüber, wie die Abhängigkeiten miteinander verknüpft sind. Dies ist jedoch wichtig, um zu erkennen, welche Folgen ein Austausch einer Technologie hat. Wenn eine Technologie eine exakte Alternative zu einer anderen Technologie darstellt, so kann sie möglicherweise einfach ausgetauscht werden. Es existieren jedoch nur selten exakte Alternativen einer Technologie. Wenn der neuen Technologie also Features fehlen, müssen diese wieder durch weitere Technologien oder eigene Implementierungen ersetzt werden. Zudem können auch weitere Technologien des Projektes abhängig von der auszutauschenden Technologie sein; dann müsste auch die abhängige Technologie ausgetauscht werden.

Darüber hinaus ist ohne Dokumentation nicht immer klar, für welche Features die Technologie eingesetzt wurde oder dass die Technologie nur aufgrund der Abhängigkeit zu einer anderen benötigten Technologie eingesetzt wurde.

Deshalb ist das Wissen über die Technologielandschaft einer Software beim Technologieaustausch wichtig für den Architekten. Denn wenn keine genaue Dokumentation darüber vorhanden ist, ist es schwierig zu sagen, wieso sich für eine bestimmte Technologie entschieden wurde und welchen Einfluss diese auf andere haben könnte. Aber auch bei Neuentwicklungen kann die Darstellung von Abhängigkeiten dem Architekten die Auswahl einer Technologie erleichtern. Durch eine Darstellung der Abhängigkeiten wird er sich der Komplexität besser bewusst. Zudem ist er in der Lage, auf Technologien zu verzichten, welche viele Abhängigkeiten mit sich bringen, wenn es eine Technologie gibt, die den Anforderungen entspricht, aber weniger Abhängigkeiten hat.

Da dieselben Technologien in unterschiedlichen Projekten mitunter auch unterschiedliche Abhängigkeiten nutzen, wird davon ausgegangen, dass unterschiedliche Arten von Abhängigkeiten existieren. Für diese Abhängigkeiten gelten Eigenschaften, die regeln, wann diese Abhängigkeiten benötigt werden oder nicht.

Um eine Entscheidungsunterstützung für Architekten zur Auswahl von Technologien zu erstellen, welche Abhängigkeiten berücksichtigt, ist es deshalb erforderlich, diese verschiedenen Arten von Abhängigkeiten zu klassifizieren. Um eine solche Klassifizierung zu erarbeiten, wird das nachfolgende Vorgehensmodell verfolgt:

1. Untersuchung wissenschaftlicher Arbeiten nach Technologieabhängigkeiten
2. Klassifizierung dieser Abhängigkeiten
3. Weitere Quellen nach Abhängigkeiten untersuchen:
 - a. GitHub-Projekte (vorwiegend Build-Dateien)
 - b. Dokumentationen von Technologien
 - c. Foren (exemplarisch Stack Overflow)
4. Identifizierung der Abhängigkeiten aus der Literatur, welche auch in der Praxis vorkommen
5. Überprüfung dieser Abhängigkeiten daraufhin, ob sie für die Auswahl von Technologien relevant sind
6. Überprüfung, ob relevante Abhängigkeiten fehlen, um den Architekten zu unterstützen
7. Anhand der Analyse: Klassifizierung der Abhängigkeiten von Technologien

3.1.1 Quellen der Abhängigkeiten

Abhängigkeiten nach Kruchten und Zimmermann

Für den ersten Schritt des Vorgehensmodells wird die Literatur nach Arten von Abhängigkeiten untersucht. Dazu werden die wissenschaftlichen Arbeiten von Kruchten und von Zimmermann analysiert.

Kruchten et al. [2004] beschäftigen sich in ihrem Paper mit Abhängigkeiten von Entwurfsentscheidungen. Sie beschreiben einen immensen Wissensverlust durch implizite Entwurfsentscheidungen. Um diesem Wissensverlust entgegenzuwirken, definiert die Arbeit Abhängigkeiten im Rahmen von Entwurfsentscheidungen. Durch diese Ontologie sollen Zusammenhänge zwischen Problemen, deren Möglichkeiten und deren Ergebnis aufgezeigt werden. Kruchten et al. [2004] kommen in ihrem Paper zu folgenden möglichen Abhängigkeiten zwischen Entwurfsentscheidungen:

Constrains: Eine Entscheidung B ist an eine Entscheidung A gebunden. Dies bedeutet, wenn eine Entscheidung A gefallen ist, dann ist auch die Entscheidung B gefallen.

Forbids: Eine Entscheidung verhindert, dass eine weitere Entscheidung getroffen wird. Die Zielentscheidung B kann nicht getroffen werden, solange die Entscheidung A nicht getroffen wurde. Die Entscheidung B muss also in einem Zustand höher als 0 sein, wenn die Entscheidung A in einem Zustand von 0 ist.

Enables: Eine Entscheidung A macht eine Entscheidung B möglich. Es bedeutet aber nicht, dass sich auch für Entscheidung B entschieden werden muss. Zudem kann sich auch für Entscheidung B entschieden werden, ohne dass Entscheidung A getroffen wird.

Subsumes: Eine Entscheidung A umfasst eine Entscheidung B, da Entscheidung A die Verallgemeinerung von Entscheidung B ist.

Conflicts with: Eine Entscheidung A und eine Entscheidung B schließen sich gegenseitig aus.

Overrides: Eine Entscheidung A zeigt eine Ausnahme von Entscheidung B an. Dies kann einen Sonderfall oder auch einen Bereich darstellen, in dem Entscheidung B nicht zutrifft.

Comprises (Is Made of, Decomposes into): Wenn Entscheidung A eine sehr umfangreiche und komplizierte Entscheidung ist, kann diese durch weitere Entscheidungen bestehen. Diese Beziehung ist stärker als „Constrains“.

Is an Alternative to: Entscheidungen A und B sind sich sehr ähnlich und gehören zum gleichen Problem, sind aber verschiedene Entscheidungen.

Is Bound to (stark): Eine Entscheidung A erzwingt die Entscheidung B und andersherum. Die Entscheidungen sind also aneinandergebunden und werden zum gleichen Zeitpunkt getroffen.

Is Related to (schwach): Die Entscheidungen stehen in irgendeiner Art von Beziehung.
Dependencies: Beschreibt die Abhängigkeit, wenn eine Entscheidung B eine Entscheidung A erzwingt, eine Entscheidung B in Entscheidung A zerfällt oder eine Entscheidung A eine Entscheidung B überschreibt.

Zimmermann et al. [2009] entwickeln, aufbauend auf den oben genannten Abhängigkeiten von Kruchten et al. und ihren eigenen älteren Arbeiten, ein erweitertes UML-Modell zur Entscheidungsunterstützung. Sie gehen davon aus, dass Probleme und Lösungen von Architekten häufig in einem betrachtet werden und dies zu einem Mangel bei der Entscheidungserfassung führt. Deshalb definieren Zimmermann et al. [2009] in ihrem Paper Entwurfsentscheidungen als Fragen (Issues), Alternativen, Ergebnisse und unterscheiden verschiedene Typen von Abhängigkeitsbeziehungen. Durch die Abhängigkeiten soll die Entscheidungsfindung besser organisiert werden.

Laut Zimmermann ist dessen so entstandenes Modell zur Entscheidungsunterstützung so allgemein gehalten, dass es auch für andere Wissensgebiete interessant ist. Ähnlich wie bei Kruchten et al. [2004] wurden so folgende Abhängigkeitsbeziehungen erforscht:

Influences: Erfasst übergreifende Angelegenheiten zwischen Fragen.

RefinedBy: Eine Frage verfeinert eine weitere Frage.

DecomposeInto: Eine Entscheidung A zerfällt in die Entscheidung B und B ist so die Zerlegung von Entscheidung A.

Forces: Drückt aus, dass die Entscheidung A zu einer Alternative zwangsläufig eine weitere Alternative (Entscheidung B) in einem anderen Problem auswählt. Die Entscheidungen erzwingen sich also.

InCompatibleWith: Zeigt, dass bestimmte Kombinationen von Alternativen nicht zusammenarbeiten können.

CompatibleWith: Zeigt das die Kombination von Alternativen möglich ist.

Triggers: Die Entscheidung A für eine Alternative löst ein anderes Problem aus und damit die Themengruppe, die dieses Problem enthält.

HasOutcome: Beziehung zwischen einem Problem und seinem Ergebnis. Immer eine 1:n-Beziehung, bei der alle Ergebnisse unterschiedliche Namen haben.

Tabelle 1: Abhängigkeiten von Kruchten und Zimmermann zusammengefasst

Kruchten	Zimmermann	Zusammengefasst
IsBoundTo (beidseitig)	Influences	<i>Bidirektionale Abhängigkeit</i> A erzwingt B & B erzwingt A
Subsumes, Comprises, Overrides	RefinedBY	<i>Erweiterung</i> B erweitert A
Constraints	Forces	<i>Direktionale Abhängigkeit</i> A erzwingt B oder B erzwingt A
Conflict with (Forbids)	InCompatibleWith	<i>Nicht Kompatibel</i> A und B sind nicht kompatibel; A verbietet B
Enables	Triggers	<i>Ermöglichung</i> A ermöglicht B
IsAlternativeTo	HasOutcome	<i>Alternativen</i> A ist eine Alternative zu B
Is Related To		<i>Undefinierte Abhängigkeit</i> A und B sind in irgendeiner Form abhängig
	CompatibleWith	<i>Kompatibel</i> A und B sind kompatibel zueinander

Da die Abhängigkeiten von Zimmermann, aber auch von Kruchten sehr allgemein gehalten sind [Zimmermann et al., 2009], sollten diese auch auf Technologieabhängigkeiten übertragen werden können. Um dies zu verifizieren, werden weitere Quellen untersucht. Zunächst sollen dazu GitHub-Projekte auf Abhängigkeiten untersucht werden und anschließend Dokumentationen von Technologien sowie das Forum „Stack Overflow“.

GitHub-Projekte/Maven-Projekte

GitHub-Projekte werden als Quelle ausgewählt, um existierende Projekte aus der Praxis zur Verifizierung zu verwenden. Aufgrund der Fülle an Projekten auf GitHub³ werden vorwiegend Projekte untersucht, die mit Build-Werkzeugen entwickelt wurden. Build-Werkzeuge unterstützen bei der Entwicklung eines Projektes und binden beispielsweise Abhängigkeiten für das Projekt ein. Diese Art von Werkzeugen können so

³ 25 Millionen Software-Projekte [Jackson, 2015]

Abhängigkeiten initial oder bei Aktualisierungen herunterladen und der Entwickler muss dies nicht selbst tun. Beispiele für diese Werkzeuge, die den Build-Prozess von Projekten unterstützen, sind Maven, sbt und bei Frontend-Entwicklung häufig auch Bower oder der Task-Runner Grunt. Anhand der vielen verschiedenen Werkzeuge, die zur Unterstützung von Softwareprojekten vorhanden sind, ist gut zu erkennen, wie wichtig die Thematik während der Softwareentwicklung und auch während der Evolution ist.

Nach Popp [2013] und der Analyse im Rahmen der Arbeit, wurde festgestellt, dass die verschiedenen Build-Prozesse sich sehr ähneln. Es gibt zwar einige Unterscheidungen in der Definierung der Abhängigkeiten, jedoch sind dieselben Arten von Abhängigkeiten in den Build-Dateien zu finden⁴. Deswegen wird im Weiteren von Build-Dateien gesprochen, die exemplarisch anhand von Maven-pom.xml-Dateien dargestellt werden.

Maven stellt ein Framework zum Projektmanagement dar. Es ist dadurch weit mehr als nur ein Build-Management-Werkzeug, als das es häufig beschrieben wird [Bachmann, 2009]. Maven hilft Entwicklern, vorwiegend Java-Programme standardisiert zu erstellen und zu verwalten, wodurch viele Schritte des Softwareerstellungsprozesses automatisiert werden. Es beruht dabei auf vier Grundsätzen, die heute als „state of the art“ in der Softwareentwicklung gelten: Konvention vor Konfiguration, Wiederverwendbarkeit, deklarative Ausführung und einheitliche Organisation von Abhängigkeiten [Bachmann, 2009]. Gerade der letztgenannte Grundsatz ist eine der größten Stärken von Maven [Popp, 2013] und macht Maven-Projekte relevant für die Analyse von Abhängigkeitstypen.

Informationen für ein durch Maven erstelltes Softwareprojekt werden in einer XML-Datei namens pom.xml gespeichert. Pom steht für Project Object Model und enthält alle Informationen zu einem Projekt. Es sind alle Softwareabhängigkeiten angegeben, die das Projekt zu anderen Softwareprojekten hat. Diese Abhängigkeiten werden aufgelöst, indem Maven ermittelt, ob die Datei im lokalen Verzeichnis bereits vorhanden ist. Ist die Datei lokal vorhanden, wird sie verwendet, ohne sie ins Projektverzeichnis zu kopieren. Ist sie das allerdings nicht, versucht Maven die Datei aus dem Internet ins lokale Verzeichnis zu kopieren.

⁴ Andere Build-Werkzeuge, siehe Anhang A.5

Da die Abhängigkeiten zuvor in die pom.xml eingetragen werden müssen, können diese mithilfe der Subelemente ArtifactId, ihrer Versionsnummer und der GroupId unter <http://search.maven.org/> eingesehen werden. Hierzu ist es jedoch nötig, die Abhängigkeiten von vornherein zu kennen.

Alle transitiven Abhängigkeiten⁵ müssen nicht in der pom.xml angegeben werden, da Maven die pom.xml der abhängigen Technologien liest und so die weiteren Abhängigkeiten erhält [Bahdanovich, 2013].

Für die Untersuchung der Build-Dateien wurden etwa 100 GitHub-Projekte⁶ untersucht. Für den Vergleich der Build-Dateien in GitHub-Projekten wurde nach folgenden Begriffen gesucht: *Maven, Bower, Grunt, sbt, package.json, Dependency, pom.xml, sapui5, openui5, play, angular*.

Zum größten Teil beschränkte sich die Suche auf die Filterung nach **Repositories** (Verzeichnissen), allerdings wurde teilweise auch unter **Code** nach den Begriffen gesucht.

Tabelle 2: Abhängigkeiten in Build-Dateien

Abhängigkeiten aus Literatur	Build-Dateien
<i>Bidirektionale Abhängigkeit</i> A erzwingt B & B erzwingt A	In Build-Dateien existieren keine bidirektionalen Abhängigkeiten. Diese Abhängigkeit würde einen Zyklus darstellen, der nicht durchlaufen werden kann. Jeder Technologie fehlt die andere, weshalb diese Abhängigkeit verboten ist. Wenn sie doch existiert, gibt der Build-Prozess eine Fehlermeldung aus [mszalbach, 2013; Makoto, 2014].

⁵ Transitive Abhängigkeit: Eine Technologie A besitzt eine Abhängigkeit zu einer Technologie B, welche wiederum von einer Technologie C abhängig ist. C ist aber nicht abhängig von A oder B. Dann ist A transitiv abhängig von C.

⁶ Die relevanten GitHub-Projekte sind im Anhang A.3 aufgelistet.

<p><i>Direktionale Abhängigkeit</i> A erzwingt B</p>	<p>Nur Abhängigkeiten, die wirklich benötigt werden. Beim Play Framework etwa <code>play_2.11</code>. Nachweis in zwei Projekten mit <code>pom.xml</code></p> <pre><dependencies> <dependency> <groupId>com.type- safe.play</groupId> <artifactId>play_2.11</arti- factId> <version>\${play2.ver- sion}</version> </dependency> </dependencies></pre>
<p><i>Erweiterung</i> B verfeinert A</p>	<p>Die Abhängigkeit muss nicht zwangsläufig implementiert werden, nur wenn die Technologie B genutzt werden soll. Sie kann Technologie A dann erweitern. Beispielsweise wenn Junit-Tests durchgeführt werden sollen:</p> <pre><!--https://mvnrepository.com/arti- fact/junit/junit --> <dependency> <groupId>junit</groupId> <artifactId>junit</artifactId> <version>4.11</version> <scope>test</scope> </dependency></pre> <p>[Caijiahao, 2016] Zudem können in Maven transitive Abhängigkeiten als optional definiert werden; wenn also eine Technologie A von B abhängig ist und B von C. Wenn die Technologie B nun Technologie C als optional markiert, wird Technologie A die Technologie C nicht verwenden [Tutorialspoint, 2017].</p>
<p><i>Nicht kompatibel</i> A und B sind nicht kompatibel; A verbietet B</p>	<p>Nicht kompatible Technologien werden nicht in Build-Dateien aufgeführt, da diese nicht genutzt werden. Allerdings können nicht kompatible Versionen durch <code><exclusions></code> abgebildet werden.</p>

	<pre> <dependency> <groupId>com.sap.netweaver.cloud</groupId> <artifactId>neo-sdk-core-api</artifactId> <version>\${nw.cloud.sdk.version}</version> <scope>system</scope> <systemPath>C:\Tools\SDK-JavaEE6WebProfile/api/neo-sdk-core-api-\${nw.cloud.sdk.version}.jar</systemPath> <exclusions> <exclusion> <artifactId>slf4j-api</artifactId> <groupId>org.slf4j</groupId> </exclusion> </exclusions> </dependency> </pre> <p>[Grail, 2014]</p> <p>Zudem ist es durch <code><excluded></code> möglich, transitive Abhängigkeiten auszuschließen. Wenn also eine Technologie A von B abhängig ist und B von C abhängt, kann A die Technologie C ausschließen [Tutorialspoint, 2017].</p>
<p><i>Ermöglichung</i> A ermöglicht B</p>	<p>Wenn eine Technologie A eine Technologie B benötigt, resultiert daraus, dass die Technologie B die Technologie A ermöglicht. Eine direktionale Abhängigkeit der Technologie A von der Technologie B bedeutet also auch eine Ermöglichung der Technologie A, wenn die Technologie B bereits eingesetzt wird. Dieser Fall ist dadurch unter der direktionalen Abhängigkeit von Technologie A zu finden.</p>
<p><i>Alternativen</i> A ist eine Alternative zu B</p>	<p><i>Alternative Technologien werden in Build-Dateien nicht dargestellt, da sich zu diesem Zeitpunkt schon für eine Technologie entschieden wurde. Allerdings können alternative Versionen durch eine bestimmte Syntax abgebildet werden. In diesem Fall handelt es sich um die Versionen, für die $2.11.0 \leq x < 2.12.0$ erfüllt ist</i></p>

	<pre><dependency> <groupId>org.eclipse.emf</groupId> <artifactId>org.eclipse.emf.ecore.xmi</artifactId> <version>[2.11.0,2.12.0)</version> </dependency></pre> <p>[Maven, 2017].</p>
<p><i>Undefinierte Abhängigkeit</i> A und B sind in irgendeiner Form abhängig</p>	<p>Jegliche Art von Abhängigkeit in Build-Dateien:</p> <pre><dependency> <groupId>org.apache.olingo</groupId> <artifactId>odata-server-api</artifactId> <version>\${odata.version}</version> </dependency></pre> <p>[ysokol, 2017]</p> <p><i>Es können aber auch inkompatible Technologien oder Alternativen sein, diese wären nicht in Build-Dateien zu finden (siehe nicht kompatible Technologien bzw. Alternativen), außer es handelt sich um Versionen.</i></p>
<p><i>Kompatibel</i> A und B sind kompatibel</p>	<p>Jegliche Art von Abhängigkeit in Build-Dateien und alle, die nicht inkompatibel sind.</p> <pre><dependency> <groupId>org.mockito</groupId> <artifactId>mockito-all</artifactId> <version>1.10.19</version> </dependency></pre> <p>[Jeong,2015]</p>

Neben den oben aufgeführten Abhängigkeiten existieren in Build-Dateien zudem noch die bereits beschriebenen transitiven Abhängigkeiten. Diese werden automatisch durch Maven aufgelöst, da die folgenden Abhängigkeiten durch die definierte Abhängigkeit bereits bekannt sind. Diese transitiven Abhängigkeiten sind wie eine direkte Abhängigkeit von einer Technologie zu sehen. Für weitere Forschungsarbeiten wäre es deshalb sinnvoll zu untersuchen, ob die transitiven Abhängigkeiten genutzt werden könnten, um daraus automatisiert Abhängigkeiten von Technologien zu analysieren.

Auch die Häufigkeit von gemeinsam verwendeten Technologien und das Nichtvorkommen von Technologien kann für weitere Forschungsarbeiten genutzt werden, um daraus Wissen für die Technologielandschaft zu ziehen. So könnten dadurch häufige Kombinationen analysiert werden und Architekten die Entscheidung für eine Technologie erleichtern. Durch das Nichtvorkommen könnten nicht kompatible Technologien gefunden werden.

Zudem kann es neben den definierten Abhängigkeiten wichtig sein zu definieren, in welchem Stadium eine Abhängigkeit relevant ist. Dies ist in Maven durch die Definition mit `<scope>` möglich. So können Abhängigkeiten beispielsweise nur während des Testens genutzt werden:

```
<dependency>
  <groupId> org.xmlunit</groupId>
  <artifactId> xmlunit-core </artifactId>
  <version>2.1.1</version>
  <scope>test</scope>
</dependency>
```

Quellcode-Beispiel 1: Beispiel für Scope

Bei Kruchten [2004] wird die Möglichkeit, den Scope zu deklarieren, auch als nützlich empfunden. Da die Analyse zu der Benutzung des Scopes aufgrund des zeitlich hohen Aufwandes den Rahmen dieser Arbeit übersteigt, wäre dies auch für zukünftigen Arbeiten interessant.



Dokumentationen

In Dokumentation zu Technologien werden vom Hersteller häufig Vor- und Randbedingungen für die Technologie genannt, die nötig sind, damit die Technologie funktioniert. Deshalb werden für die Analyse auch verschiedene Dokumentationen zu Technologien untersucht, darunter auch einige Technologien, welche später spezifischer untersucht werden sollen: *Foundation Framework, Laravel, Twitter Bootstrap, Play Framework, Semantic UI, Atlassian JIRA, SAPUI5, OPENUIS, AngularJs, Angular, Spring Framework, SAP HANA*⁷.

⁷ Quellen der Dokumentation im Anhang unter A.4 aufgelistet

Bei der Analyse dieser Dokumentationen sind folgende Abhängigkeiten zu finden:

Tabelle 3: Abhängigkeiten Dokumentationen

Abhängigkeiten aus Literatur	Dokumentationen
<i>Bidirektionale Abhängigkeit</i> A erzwingt B & B erzwingt A	<p>Da in der Praxis keine zyklischen Abhängigkeiten zwischen Technologien bestehen sollten, dürfen auch in Dokumentationen keine derartigen Abhängigkeiten abgebildet werden. Daraus resultiert, dass sich hierfür auch keine Beispiele in Dokumentationen finden lassen.</p>
<i>Erweiterung</i> B verfeinert A	<p>Die Hersteller des Playframeworks geben an, dass öffentliche Module genutzt werden können, welche die Features des Play Frameworks erweitern [Playframework_modules, 2017].</p> <p>Das Foundation-Framework integriert bereits Angular:</p> <p>Foundation Integrations </p> <div data-bbox="702 996 837 1131">  </div> <p>Angular Foundation 6</p> <p>The awesome folks at Pinecone created an Angular port for Foundation. Angular.js assists with creating single-page applications, one-page web applications that only require HTML, CSS, and JavaScript on the client side.</p> <p>Learn More</p> <p>[Foundation, 2017]</p>
<i>Direktionale Abhängigkeit</i> A erzwingt B	<p>Eine Voraussetzung, welche in fast allen Dokumentationen zu Technologien, die auf Java basieren, zu finden sein wird, ist, dass Java installiert ist:</p> <h2>Prerequisites</h2> <p>Play requires Java 1.8. To check that you have the latest JDK, please run:</p> <pre>java -version</pre> <p>You should see something like:</p> <pre>java version "1.8.0_121" Java(TM) SE Runtime Environment (build 1.8.0_121-b13) Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)</pre> <p>[Play, 2017]</p> <p>Bei Webtechnologien ist häufig auch wichtig, welche Voraussetzungen der Server besitzt:</p>

	<h2># Server Requirements</h2> <p>The Laravel framework has a few system requirements:</p> <ul style="list-style-type: none"> • PHP ≥ 5.4, PHP < 7 • Mcrypt PHP Extension • OpenSSL PHP Extension • Mbstring PHP Extension • Tokenizer PHP Extension <p>As of PHP 5.5, some OS distributions may require you to manually install the PHP JSON extension. When using Ubuntu, this can be done via <code>apt-get install php5-json</code>.</p> <p>[Laravel, 2017]</p> <p>Die Hersteller von Twitter Bootstrap weisen darauf hin, dass alle Java-Plugins jQuery verwenden und dieses integriert sein muss:</p> <div> <p>jQuery required</p> <p>Please note that all JavaScript plugins require jQuery to be included, as shown in the starter template. Consult our bower.json to see which versions of jQuery are supported.</p> </div> <p>[Bootstrap, 2017]</p>
<p><i>Nicht kompatibel</i> A und B sind nicht kompatibel; A verbietet B</p>	<p>Nicht kompatible Technologien werden in Dokumentationen häufig nicht angegeben, da dies nicht förderlich ist für die Verbreitung der Technologien. Allerdings wird oft angegeben, ob die Technologien in den verschiedenen Browsern funktionieren:</p> <h3>What Won't Work? ⚠</h3> <ul style="list-style-type: none"> • The Grid: Foundation's grid uses <code>box-sizing: border-box</code> to apply gutters to columns, but this property isn't supported in IE8. • Desktop Styles: Because the framework is written mobile-first, browsers that don't support media queries will display the mobile styles of the site. • JavaScript: Our plugins use a number of handy ECMAScript 5 features that aren't supported in IE8. <p>[Foundation, 2017]</p>
<p><i>Ermöglichung</i> A ermöglicht B</p>	<p>Technologie B benötigt Technologie A, wodurch die Technologie B ermöglicht wird, sobald Technologie A benutzt wird.</p> <p>Diese Art der Abhängigkeit wäre also in der Dokumentation von B zu finden, da dies eine direktionale Abhängigkeit von B darstellt. Die <i>Ermöglichung</i> liegt also vor, wenn A und B kooperieren und A in der Dokumentation auf B hinweist.</p>
<p><i>Alternativen</i> A ist eine Alternative zu B</p>	<p>In Dokumentationen werden generell eigentlich keine Alternativen zu der jeweiligen Technologie aufgezeigt, weil es sich negativ auf deren Marktanteil auswirken könnte. Dies wäre nur in No-budget-Projekten möglich oder wenn andere Alternativen für die Verwendung im Zusammenhang mit der Technologie aufgezeigt werden sollen.</p>

	<p>Ein Beispiel dafür lässt sich in der Twitter-Bootstrap-Dokumentation finden:</p> <p>„Bootstrap verwendet Autoprefixer, um mit den CSS Vendor Präfixen klarzukommen. Falls du Bootstrap von der Less/Sass-Quelle kompilieren möchtest und nicht unser Gruntfile verwendest, musst du Autoprefixer selbst in deinen Build-Vorgang integrieren. Falls du vorkompilierte Bootstrap-Dateien oder unser Gruntfile verwendest, musst du dir darüber keine Sorgen machen, da Autoprefixer dort bereits eingebunden ist“ [Bootstrap, 2017].</p> <p>Zudem gibt es noch den Fall, dass Alternativen als unzureichend beschrieben werden: „Other frameworks deal with HTML’s shortcomings by either abstracting away HTML, CSS, and/or JavaScript or by providing an imperative way for manipulating the DOM. Neither of these address the root problem that HTML was not designed for dynamic views“ [AngularJs, 2017].</p>
<i>Undefinierte Abhängigkeit</i> A und B sind in irgendeiner Form abhängig	Undefinierte Abhängigkeiten sind in der Regel nicht vorhanden, da in Dokumentationen angegeben wird, welchen Zweck die Abhängigkeiten verfolgen.
<i>Kompatibel</i> A und B sind kompatibel	<p>Kompatible Technologien werden teilweise in Dokumentationen abgebildet:</p> <p>„Almost 100% compatible with old Play Groovy template implementation (See note below for more info)“, [https://www.playframework.com/modules].</p> <p>jQuery 3.0 Support We now support jQuery 3.0 out of the box. Learn more about jQuery 3.0 to see why you should switch.</p> <p>[semantic UI, 2017]</p>

Neben den Abhängigkeiten in der Literatur werden in Dokumentationen auch häufig Wörter wie „recommended“, also empfohlen, verwendet. Die Technologie ist dann keine direktionale, aber eine empfohlene Variante einer optionalen Abhängigkeit, wie in folgendem Beispiel:

„Having reviewed this position recently, we believe there are benefits for customers in using PostgreSQL. In addition to being tested to the same standard

as all of our databases, PostgreSQL is used; [...] That's why we now recommend you choose PostgreSQL for your JIRA server instance" [Atlassian, 2017]

Zudem wird in Dokumentationen auch erwähnt, auf welcher Art von Technologie eine Technologie basiert:

„Based on proven web standards (HTML5, CSS, OData, XML)" [openui5, 2017]

Stack Overflow

Entwickler suchen Informationen über das Web meist auf zwei Arten, entweder durch Suchmaschinen wie Google oder über Foren wie Stack Overflow. Chen u. Xing [2, 2016] untersuchten die Korrelation beider Arten. Dabei zeigte sich, dass Stack Overflow als eine wichtige Informationsquelle genutzt werden kann, weshalb das Forum in dieser Arbeit für die Analyse der Abhängigkeiten genutzt wird.

Stack Overflow ist ein Frage-Antwort-Portal und kann auf verschiedene Arten zur Untersuchung von Abhängigkeiten genutzt werden. Die erste Möglichkeit besteht darin, mithilfe von Suchbegriffen auf Stack Overflow nach speziellem Wissen zu Abhängigkeiten zu suchen, siehe Abbildung 2.

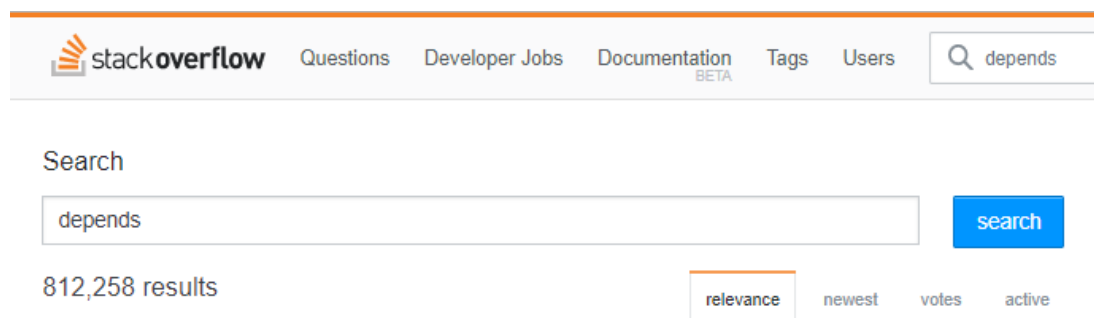


Abbildung 2: Suche nach „depends“ auf Stack Overflow

Zum anderen können aber auch Abfragen über Stack Exchange vorgenommen werden. Dadurch kann die Datenbank von Stack Overflow nach Fragen und/oder Antworten ausgelesen werden, je nachdem, welche Abfrage man verwendet wie in Abbildung 3:

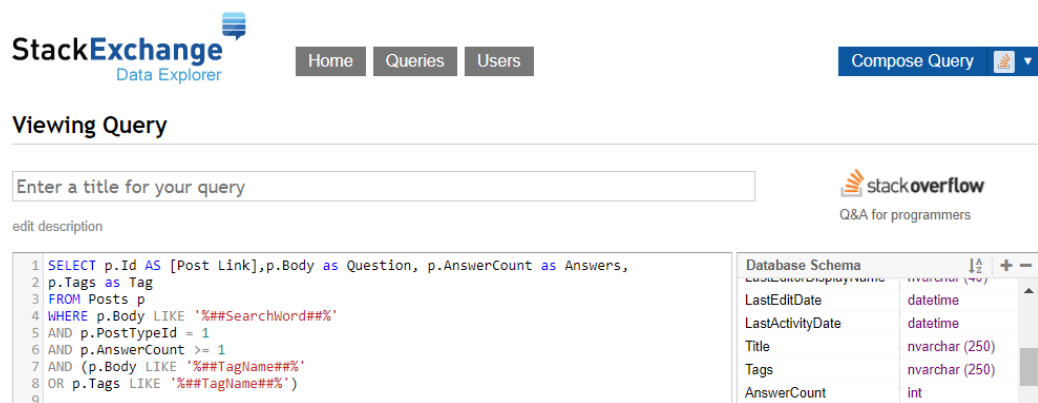


Abbildung 3: Beispiel einer Abfrage über Stack Exchange

Für die Analyse wurde zunächst an einer geeigneten Abfrage für Stack Exchange gearbeitet. Dieses Verfahren⁸ erwies sich jedoch als äußerst zeitaufwendig, und da der Fokus dieser Arbeit nicht auf einem automatisierten Verfahren liegt, wird im Weiteren mit der ersten Möglichkeit gearbeitet, indem manuell nach Wissen auf Stack Overflow gesucht wird.

Für die Analyse wurde Stack Overflow nach verschiedenen Suchbegriffen durchsucht:

- Depends (on)
- Needs
- (not) compatible (with)
- Work(s) (not/with)

Dabei wurden etwa 250 verschiedene Threads analysiert, wobei viele davon nicht weiter relevant waren. Bei einer manuellen Analyse dieser Threads wurden unter anderem folgende Abhängigkeiten aus Tabelle 4 gefunden:

Tabelle 4: Abhängigkeiten Stack Overflow

Abhängigkeiten aus Literatur	Stack Overflow
<i>Bidirektionale Abhängigkeit</i> A erzwingt B & B erzwingt A	Da es keine zyklischen Abhängigkeiten geben sollte, wird hier darauf aufmerksam gemacht und erklärt, wie dies gelöst werden kann. „A cyclic dependency in eclipse indicates that there is a cycle in the buildpaths between projects in Eclipse.

⁸ Weiteres dazu Anhang A.6

	<p>So if you have 5 projects, say A, B, C, D and E, then a cyclic dependency could be that:</p> <ol style="list-style-type: none"> 1. A requires B in its build-path 2. B requires D in its build-path 3. D requires A in its build-path <p>Hence A->B->D->A is a cycle.</p> <p>Because of this cycle, Eclipse does not know which project to compile first.</p> <p>You need to refactor your code to remove this cyclic dependency. Or if the actual code doesn't have such dependencies, remove the build-path entries which are unnecessary" [Advani, 2013]</p>
<i>Erweiterung</i> B verfeinert A	<p>„SAP Fiori is a Design Guide for building Web-based App that are easy to use, responsive, delightful, etc. The technology for SAP Fiori is based on SAP UI5.“ [Niehues, 2015]</p> <p>„UI5 does use several parts of jQuery UI internally, like the ‚position‘ plugin, but it does not use its UI control mechanism, as it relies on the DOM being there already, which does not work well for controls that are more complex than say tabstrips and also not for very big apps“ [Akudev, 2015].</p>
<i>Direktionale Abhängigkeit</i> A erzwingt B	<p>„Bootstrap needs jquery to run“ [Almis, 2015]</p>
<i>Nicht kompatibel</i> A und B sind nicht kompatibel; A verbietet B	<p>Für Versionen:</p> <p>„Bootstrap 3 is not compatible with jQuery 3.0 and above at the moment. So latest version of jQuery that can be used with Bootstrap 3 is 2.2.4.“ [Jakob, 2016].</p>
<i>Ermöglichung</i> A ermöglicht B	<p><i>Kein Beispiel gefunden</i></p>
<i>Alternativen</i> A ist eine Alternative zu B	<p>Alternativen werden auf Stack Overflow aufgezeigt:</p> <p>„[B]elow are a number of Ext JS alternatives currently available“ [Dascalescu, 2008]</p>
<i>Undefinierte Abhängigkeit</i> A und B sind in irgendeiner Form abhängig	<p>„It's irrelevant. You need to ensure that you can access your DBMS of choice from your Java code. That's all. Olingo is nothing else than a Java framework. It's a way of exposing your application's resources. Therefore, if</p>

	your Java code can read the resources from your Oracle database, then your application can expose these resources using Olingo“ [C.N., 2015]
<i>Kompatibel</i> A und B sind kompatibel	„I started an app with Play Framework (Scala) and AngularJS v1.3.“ [Roca, 2015]

Außer für die Ermöglichung und die bidirektionale Abhängigkeit konnten für alle Abhängigkeiten aus der Literatur auch auf Stack Overflow Beispiele gefunden werden. Für spätere Arbeiten wäre es jedoch sinnvoll, die Methode der Abfrage über Stack Exchange so zu verbessern, dass nur noch relevante Threads angezeigt werden und der Vorgang so automatisiert werden kann.

Insgesamt wurden durch die Analyse in den verschiedenen Quellen folgende Abhängigkeiten gefunden bzw. nicht gefunden:

Tabelle 5: Gesamtüberblick über vorkommende Abhängigkeiten

Übergreifender Name	Wissenschaftliche Arbeiten	Build-Dateteien	Dokumentationen	Stack Overflow
<i>Bidirektionale Abhängigkeit</i>	✓	✗	✗	✗
<i>Erweiterung</i>	✓	✓	✓	✓
<i>Direktionale Abhängigkeit</i>	✓	✓	✓	✓
<i>Nicht kompatibel</i>	✓	✗	✓	✓
<i>Ermöglichung</i>	✓	✗	✗	✗
<i>Alternativen</i>	✓	✗	✗	✓
<i>Undefinierte Abhängigkeit</i>	✓	✓	✗	✓
<i>Kompatibel</i>	✓	✓	✓	✓

In Tabelle 5 wird deutlich, dass eine *bidirektionale Abhängigkeit* nur in der Literatur, nicht aber in der Praxis vorkommt. Dies ist darauf zurückzuführen, dass sich die Literatur mit Abhängigkeiten von Entscheidungen im Allgemeinen und nicht speziell für Technologien beschäftigt. Bei Entscheidungen sind gegenseitige Beeinflussungen durchaus möglich. In der Softwareentwicklung sind solch zyklischen Beziehungen allerdings verboten, weshalb sich keine derartigen Abhängigkeiten in der Praxis finden lassen.

Da die Ermöglichung eine andere Sichtweise auf die *direktionale Abhängigkeit* darstellt, ist es nicht verwunderlich, dass diese Abhängigkeit nicht in der Praxis gefunden wird. Wie bereits erwähnt, ist es nicht möglich, diese Art der Abhängigkeit in Build-Dateien nachzuweisen. In Dokumentationen wären diese eher unwahrscheinlich, aber es ist dennoch denkbar, dass ein Hersteller es als einen Vorteil herausstellt, dass mit seiner Technologie eine andere ermöglicht wird. Und auch bei Stack Overflow wären Threads mit Antworten dieser Art denkbar. Jedoch konnten keine Beispiele in dem untersuchten Material gefunden werden.

3.1.2 Relevante Abhängigkeiten

Nach der Identifizierung der Abhängigkeiten werden diese nun nach ihrer Relevanz für die Entscheidungsunterstützung gefiltert. Dazu wird geprüft, ob die Abhängigkeiten zielführend für die Arbeit des Architekten sein könnten.

Tabelle 6: Relevanz der Abhängigkeiten

Abhängigkeiten aus Literatur	Dokumentationen
<i>Bidirektionale Abhängigkeit</i> A erzwingt B & B erzwingt A	Da zyklische Abhängigkeiten bei Technologien nicht vorkommen dürfen und sie nur in der Literatur und nicht in der Praxis zu finden sind, sind diese <u>nicht relevant</u> für die Arbeit des Architekten zur Auswahl von Technologien.
<i>Erweiterung</i> B verfeinert A	<u>Relevant</u> – da es sich hierbei um Erweiterungsmöglichkeiten handelt, können diese interessant sein, wenn der Technologie bestimmte Features fehlen.
<i>Direktionale Abhängigkeit</i> A erzwingt B	Die Abhängigkeit ist für den Architekten <u>relevant</u> und wohl die wichtigste Art der Abhängigkeit, da diese Abhängigkeit für die Technologie immer benötigt wird.
<i>Nicht kompatibel</i>	<u>Relevant</u> , da keine Technologien miteinander benutzt werden können, die nicht kompatibel miteinander sind.

A und B sind nicht kompatibel; A verbietet B	Hierbei wird jedoch nur der Bereich der Webanwendungen betrachtet, also Technologien, bei denen der Entwickler davon ausgehen würde, dass diese miteinander funktionieren. Ansonsten würde diese Art der Abhängigkeit zu unübersichtlich. Auch nach Kruchten [2004] sind Nicht-Existenzentscheidung wichtig, weshalb nicht kompatible Technologien in die Klassifizierung aufgenommen werden sollten.
<i>Ermöglichung</i> A ermöglicht B	Die Art der Abhängigkeit wurde in der Analyse der Praxis-Quellen nicht gefunden, kann jedoch <u>relevant</u> für den Architekten sein, da dieser sich so eher für eine Technologie entscheiden kann, wenn beispielsweise bereits Technologie B benutzt wird und Technologie A die Technologie B benötigen würde. Die Alternative zu Technologie A wäre aber Technologie C und diese würde zusätzlich Technologie D benötigen. So ermöglicht Technologie B die Technologie A.
<i>Alternativen</i> A ist eine Alternative zu B	Alternativen sind <u>relevant</u> . Es könnte beispielsweise möglich sein, dass die vorgeschlagene Technologie nicht genutzt werden kann, etwa weil die Lizenzkosten zu hoch sind. In diesem Fall wäre es sinnvoll, Alternativen zu der Technologie aufzuzeigen.
<i>Undefinierte Abhängigkeit</i> A und B sind in irgendeiner Form abhängig	<u>Nicht relevant</u> , da jegliche undefinierten Abhängigkeiten in den anderen definierten Abhängigkeiten auftauchen sollten.
<i>Kompatibel</i> A und B sind kompatibel	Da alle Abhängigkeiten, welche nicht kompatibel sind, aufgezeigt werden sollten, kann diese Abhängigkeit <u>vernachlässigt</u> werden, weil automatisch alle weiteren kompatibel sind. Zudem wäre eine Darstellung aller möglichen kompatiblen Technologien zu unübersichtlich, da dies weit über hundert bis zu tausende Technologien im Bereich der Webanwendungen sein könnten.

Nach der Prüfung der Abhängigkeiten auf ihre Relevanz ist zu untersuchen, ob noch mögliche Abhängigkeiten aus den anderen Quellen existieren, welche in der Literatur nicht genannt wurden, aber dennoch relevant für den Architekten sind.

Die Empfehlungen, welche häufig in Dokumentationen durch den Hersteller angegeben werden, sind ebenfalls relevant. Sie zeigen dem Architekten, dass eine Technologie B vom Hersteller in Verbindung mit seiner Technologie A empfohlen wird. Auch das Wissen darum, dass Technologien auf anderen Technologien basieren, kann wertvoll sein, da es für die Entwickler einfacher ist, mit einer Technologie zu arbeiten, die beispielsweise auf JavaScript basiert, sofern er bereits Erfahrung mit dieser Programmiersprache besitzt.

Eine weitere Abhängigkeit, die in der Literatur nicht beachtet wird, besteht darin, dass ein Feature von der Technologie A eine Technologie B benötigt. Technologie B wäre nur dann notwendig, wenn das Feature genutzt wird. Damit liegt eine direktionale Abhängigkeit vor. Diese direktionale Abhängigkeit darf jedoch nur bei Verwendung des Features betrachtet werden. Andernfalls könnte ein Architekt sich aufgrund dieser Abhängigkeit gegen die Technologie A und für eine Technologie C entscheiden, obwohl er das Feature der Technologie A gar nicht benötigt und die Abhängigkeit somit nicht relevant wäre. Ein Beispiel für diese Art der Abhängigkeit ist das Play Framework. Nur wenn das Play Framework im Zusammenhang mit Java und nicht mit Scala genutzt wird, muss die pom.xml eine Abhängigkeit von einer Java-Bibliothek besitzen:

```
<!-- only if using Java -->
<dependency>
  <groupId>com.typesafe.play</groupId>
  <artifactId>play-java_2.11</artifactId>
  <version>${play2.version}</version>
</dependency>
```

Quellcode-Beispiel 2: Beispiel für Feature-Direktionale

Wenn jedoch Scala benutzt wird, existiert diese Abhängigkeit nicht. Dies erweitert die Klassifizierung um eine weitere Abhängigkeit, eine Feature-Direktionale Abhängigkeit.

Zudem wäre es auch vorstellbar, dass eine Technologie A eine weitere Technologie benötigt, allerdings mit der Bedingung verknüpft, dass die weitere Technologie B oder C sein muss. Hier liegt also eine austauschbare direktionale Abhängigkeit vor. Leider konnte hierfür in der Praxis kein Beispiel gefunden werden.

3.1.3 Klassifizierung von Abhängigkeiten

Durch die Analyse der Quellen wurden verschiedene Abhängigkeiten gefunden. Jedoch unterscheiden sich einige noch stark voneinander. Einige der gefundenen Ab-

hängigkeiten stehen zwar im Zusammenhang mit Technologien, jedoch sind die Technologien nicht abhängig im direkten Sinne. Deshalb werden sie nochmals gruppiert und dadurch noch spezifischer zusammengefasst. Diejenigen Abhängigkeiten, welche eine tatsächliche Abhängigkeit beschreiben, gehören weiterhin der Gruppe Abhängigkeit an. Die Abhängigkeiten hingegen, welche keine existierende Abhängigkeit darstellen, werden fortan als Beziehungen bezeichnet.

Abhängigkeiten:

1. Direktionale: Technologie A benötigt Technologie B, um zu funktionieren. Technologie B wird also bei dem Gebrauch von Technologie A immer mitverwendet.
2. Austauschbare Direktionale: Technologie A benötigt Technologie B oder Technologie C, um zu funktionieren. Zwar konnte diese Abhängigkeit durch die Analyse nicht gefunden werden, aber da sie durchaus vorstellbar ist, wird sie mit in die Klassifizierung aufgenommen.
3. Feature-Direktionale: Bestimmte Features von Technologie A benötigen Technologie B. Die Technologie B wird nur dann verwendet, wenn das zugehörige Feature in dem Projekt verwendet werden soll.

Beziehungen:

1. Ermöglichung: Technologie A ermöglicht Technologie B, da die Technologie B eine direktionale Abhängigkeit von A aufweist.
2. Empfehlung: Die Entwickler von Technologie A empfehlen die Technologie B.
3. Erweiterung: Eine Technologie B erweitert Technologie A, häufig besteht dabei eine direktionale Abhängigkeit der Technologie B von A; dies muss allerdings nicht der Fall sein. (*Die Erweiterung war zunächst ein Teil der Abhängigkeiten, aufgrund der Bewertung durch die Entwickler wurde dies aber überdacht und die Erweiterung den Beziehungen zugeordnet.*)
4. Basierend auf: Eine Technologie A basiert auf der Technologie B. Dies kann zum Beispiel bedeuten, dass eine Technologie eine bestimmte Programmiersprache verwendet.
5. Nicht kompatibel: Technologie A funktioniert nicht mit Technologie B.
6. Häufige Kombination: Eine Technologie A wird häufig mit Technologie B genutzt.
7. Alternative: Eine Technologie B ist eine Alternative zu A.

Die Abhängigkeit der Ermöglichung kann als eine redundante Abhängigkeit zu der direktionalen Abhängigkeit gesehen werden, da alle Technologien, die eine Technologie B benötigen, auch dadurch ermöglicht werden, wenn Technologie B schon im Einsatz ist. Diese Redundanz wird jedoch in Kauf genommen, um für das prototypische Werkzeug einen möglichst schnellen Datenzugriff zu gewährleisten. Ansonsten müsste die Abfrage hierfür zunächst alle Technologie ausfindig machen, welche von Technologie B abhängig sind, anstatt einfach die Technologien von B auszulesen. Zudem kann so eine zyklische Abfrage unterbunden werden, welche das Werkzeug abstürzen lassen würde.

Insgesamt kann gesagt werden, dass nicht alle Abhängigkeiten aus der Literatur auch in der Praxis zu finden sind, vorwiegend aufgrund der Allgemeingültigkeit der klassifizierten Abhängigkeiten aus der Literatur. Zudem existieren Technologieabhängigkeiten in der Praxis, welche nicht von Kruchten und Zimmermann et al. beschrieben wurden, wie etwa Empfehlungen zur Verwendung oder auch häufige Kombinationen. Diese Beziehungen könnten in der Literatur unter dem Punkt der ‚Is Related to‘ Abhängigkeit aufgeführt werden. Für diese Arbeit zur Entwicklung eines Prototyps zur Entscheidungsunterstützung ist es jedoch wichtig die genauen Unterschiede zu identifizieren. Zudem erhält der Architekt durch diese Beziehungen zusätzlichen Informationen, die ihm helfen können eine Entwurfsentscheidung zu treffen.

Die existierenden Abhängigkeiten werden in den verschiedenen Quellen unterschiedlich abgebildet, so geben Build-Dateien zunächst keine genaue Auskunft darüber, welche Art von Abhängigkeit vorliegt, nur im Vergleich mit vielen weiteren Build-files oder mit anderen Quellen ergibt sich aus den Build-Dateien eine identifizierbare Abhängigkeit. Aus diesem Grund wäre auch eine automatische Generierung des Wissens aus vielen Build-Dateien sinnvoll.

Im Vergleich zu Build-Dateien definieren Dokumentationen die Abhängigkeiten genauer, denn es gibt in den meisten Fällen eine Begründung zur der benötigten Abhängigkeit oder einer Beziehung. In Foren dagegen ist ein Aufschluss über die abgebildete Abhängigkeit unterschiedlich schwierig zu treffen, da es zu einem großen Teil auf den Wissenstand des Verfassers eines Posts ankommt. Aber auch hier wäre eine Automatisierung des Wissens durch Stack Exchange interessant.

3.1.4 Bewertung der Klassifizierung

Um die Klassifizierung zu überprüfen, wurde diese mithilfe von drei Mitarbeitern der Firma INIT Solution GmbH bewertet. Die Mitarbeiter sind ein Juniorentwickler und

zwei Seniorentwickler mit langjähriger Berufserfahrung im Bereich SAP. Beide Senior Entwickler übernehmen teilweise in Projekten auch die Aufgaben eines Architekten, da bei INIT noch kein ausgewiesener Architekt angestellt ist.

Um die Klassifizierung zu bewerten wurde ihnen zunächst die Thematik der Arbeit und die Vorgehensweise erläutert. Darauf aufbauend wurde die Klassifizierung der Abhängigkeiten und Beziehungen dargestellt und erklärt. Dann wurden sie dazu befragt:

- ob sie die dargestellte Klassifizierung als relevant erachten;
- ob sie möglicherweise Abhängigkeiten oder auch Beziehungen darin für nicht relevant halten;
- ob Abhängigkeiten oder Beziehungen existieren, welche nicht in der Klassifizierung vorkommen, welche sie allerdings als relevant erachten.

Generell lässt sich sagen, dass keiner der Befragten nach näherer Betrachtung Abhängigkeiten oder Beziehungen in der Klassifizierung vermisste und dass die Befragten von einer sehr umfangreichen und vor allem auch relevanten Klassifizierung sprach. Der Junior-Entwickler, welcher seit einem Jahr bei der INIT Solution GmbH arbeitet und bereits langjährige Weberfahrung mitbringt, würde die Klassifizierung als vollständig erachten. Allerdings würden ihn die Erweiterungen einer Technologie womöglich erst später interessieren, wenn er mit der Technologie arbeitet.

Der zweite Entwickler, welcher häufig auch Tätigkeiten eines Architekten übernimmt, befindet die Klassifizierung als „vollumfänglich relevant“. Er bemerkt nur, welche Faktoren ihn bei der Auswahl einer Technologie außerdem interessieren. So ist für ihn wichtig:

- was die Technologie kostet, sowohl in der Entwicklung als auch im Einsatz später beim Kunden;
- wie zeitaufwendig es ist, die Technologie zu implementieren;
- wie es mit Softwarequalitätsattributen, wie etwa der Wartbarkeit, bei der Technologie aussieht;
- ob der Kunde zusätzliches Know-how für die Verwendung der Software mit dieser Technologie benötigt;
- Risikobewertung: Wie zukunftssträftig ist die Technologie beispielsweise? Ist sie in ein paar Jahren veraltet?

Da diese Anmerkungen jedoch nichts mit den Abhängigkeiten zu tun haben, können sie zunächst außer Acht gelassen werden. Jedoch wäre es sinnvoll, einige dieser Anmerkungen mit in das Werkzeug einzuarbeiten.

Auch der dritte Senior-Entwickler befand die Arbeit der Klassifizierung als sehr umfangreich, allerdings war ihm unklar,

- ob die Erweiterungen zu den Abhängigkeiten zählen sollten oder zu den Beziehungen, da für ihn eine Abhängigkeit eine Muss-Abhängigkeit bedeutet. Eine optionale wäre eher eine Kann-Beziehung.
- Zudem verwies er auf weitere Abhängigkeiten, welche sich jedoch in den schon klassifizierten Abhängigkeiten verbergen. Als Beispiel nannte er Beziehungen, die sich in transitive Abhängigkeiten aufschlüsseln lassen. Transitive Abhängigkeiten verteilen sich jedoch auf die direktionalen Abhängigkeiten der jeweiligen Technologie, um Redundanzen im Datenmodell zu vermeiden. Zudem fehlte ihm die autarke Technologie, womit gemeint ist, dass generell keine Abhängigkeiten vorhanden sind.

Insgesamt wurde dadurch deutlich, dass sich die hier erarbeitete Klassifizierung für Entwickler als umfangreich und angemessen darstellt. Der Vorschlag, die Erweiterungen den Beziehungen zuzuordnen und nicht den Abhängigkeiten, wurde nach genauerer Betrachtung umgesetzt. Die Anmerkungen zu den weiteren Kriterien, welche für die Auswahl von Technologien wichtig sind, wurden für die weitere Bearbeitung der Arbeit mit aufgenommen.

4 Technologieanalyse

Um die Klassifizierung der Abhängigkeiten für eine Entscheidungsunterstützung zu evaluieren, soll ein Prototyp erstellt werden. Hierfür muss zunächst eine Grundlage von Technologiewissen geschaffen werden. Für dieses spezifische Technologiewissen müssen jedoch einige Annahmen getroffen werden, um die Arbeit einzuschränken:

- Der Umfang des spezifischen Technologiewissens ist aufgrund des hohen manuellen Aufwands für die Technologieanalyse klein gehalten und auf exemplarische Daten beschränkt. Dies könnte einen negativen Einfluss auf die Allgemeingültigkeit haben.
- Da das Technologiewissen als Teil des Architekturwissens gesehen werden kann und kein allgemeingültiger Ansatz der Modellierung von Architekturwissen existiert, kann keine vollständige Betrachtung garantiert werden. Hierzu muss die Forschung einen allgemeingültigen Ansatz erarbeiten, welcher für zukünftige Arbeiten genutzt werden kann.
- Die Modellierung der Daten beruht auf einem Lösungsansatz von Soliman et al. [2015], der bereits in der Forschung akzeptiert wurde.

4.1 Kriterien für Technologiewissen

Nach der allgemeinen Technologieanalyse, um Abhängigkeiten zu klassifizieren, folgt nun eine spezifische Analyse exemplarisch ausgewählter Frontend-Technologien für SAP HANA. Als Grundlage für diese Analyse dient das ursprüngliche Metamodell von Soliman et al. [2015] und das erweiterte Modell von Fechner [2016]⁹. Das Modell wird verwendet, da es sich bereits mit dem Lösungsraum für Technologien beschäftigt und durch die Erweiterungen von Fechner die Features der Technologie mit einbezogen werden. Dies ist wichtig, da eine Technologie immer auf der Grundlage ihrer Features ausgewählt wird.

Für die Arbeit wurden alle nicht relevanten Teile des Modells bereits entfernt:

⁹ Original Metamodelle, siehe A.7

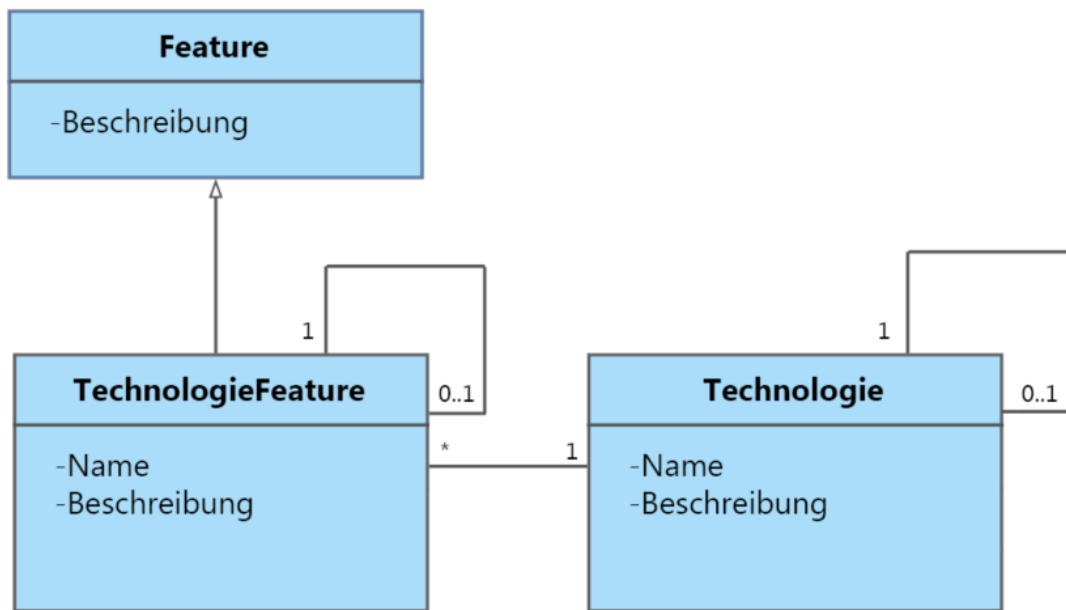


Abbildung 4: Grundlegendes Metamodell

In dem Modell (Abbildung 4) stehen Technologien und ihre Features in einer direkten Beziehung. Technologien können eine unbegrenzte Anzahl an Features besitzen. Technologie-Features sind Instanzen des abstrakten Elements Features. Dies ist notwendig, da Features sich im Namen zwar unterscheiden können, aber dennoch dieselben Funktionalitäten aufweisen. Dies muss auch im Modell dargestellt werden, um so gleichartige Features identifizieren zu können.

Für das weitere Vorgehen muss das Metamodell um Elemente ergänzt werden, welche für den ursprünglichen Zweck und auch in der Erweiterung nicht relevant waren.

Abhängigkeiten wurden zwar in dem erweiterten Modell schon bedacht, jedoch konnte jede Technologie nur eine Abhängigkeit von einer anderen Technologie haben, da weitere Abhängigkeiten nach Zimmermann et al. [2009] die Komplexität erhöhen würden. Da jedoch bereits festgestellt wurde, dass mehr als nur eine Abhängigkeit bestehen kann, muss diese Beziehung verändert werden. Dies bedeutet, dass das Metamodell um ein Element zur Beschreibung von **Abhängigkeiten** ergänzt werden muss. Zudem können auch Features Abhängigkeiten besitzen (Feature-Direktionale), weshalb eine Beziehung zwischen den Technologie-Features und den Abhängigkeiten bestehen muss. So kann jedes Feature unendlich viele Abhängigkeiten besitzen. Aber jede Abhängigkeit gehört zu einem Feature oder einer Technologie.

Neben den Abhängigkeiten können Technologien **Beziehungen** besitzen, weshalb noch ein Element zur Beschreibung der klassifizierten Beziehungen hinzugefügt werden

muss. Diese beiden Elemente sind wichtig, um Zusammenhänge zwischen Technologielandschaften zu erkennen und abwägen zu können, welche Technologie ausgewählt werden sollte.

Neben den bereits klassifizierten Kriterien sind noch andere Eigenschaften wichtig, um den exemplarischen Prototyp möglichst hilfreich aufbauen zu können. So interessieren den Architekten die **Vor- und Nachteile** einer Technologie. Im Rahmen der Technologieanalyse sollten diese deshalb betrachtet werden, um empfohlene Technologien besser vergleichen zu können. Unter Vor- und Nachteile können beispielsweise Aspekte wie eine gute Dokumentation, Lizenzkosten oder auch eine gute Community vermerkt werden. Deshalb wird das Element Technologie ergänzt um die Eigenschaften Vor- und Nachteile sowie **Links**, um weitere Informationen erhalten zu können.

Aus diesen Anforderungen ergibt sich ein erweitertes Metamodell, das als Grundlage für die weitere Analyse dient:

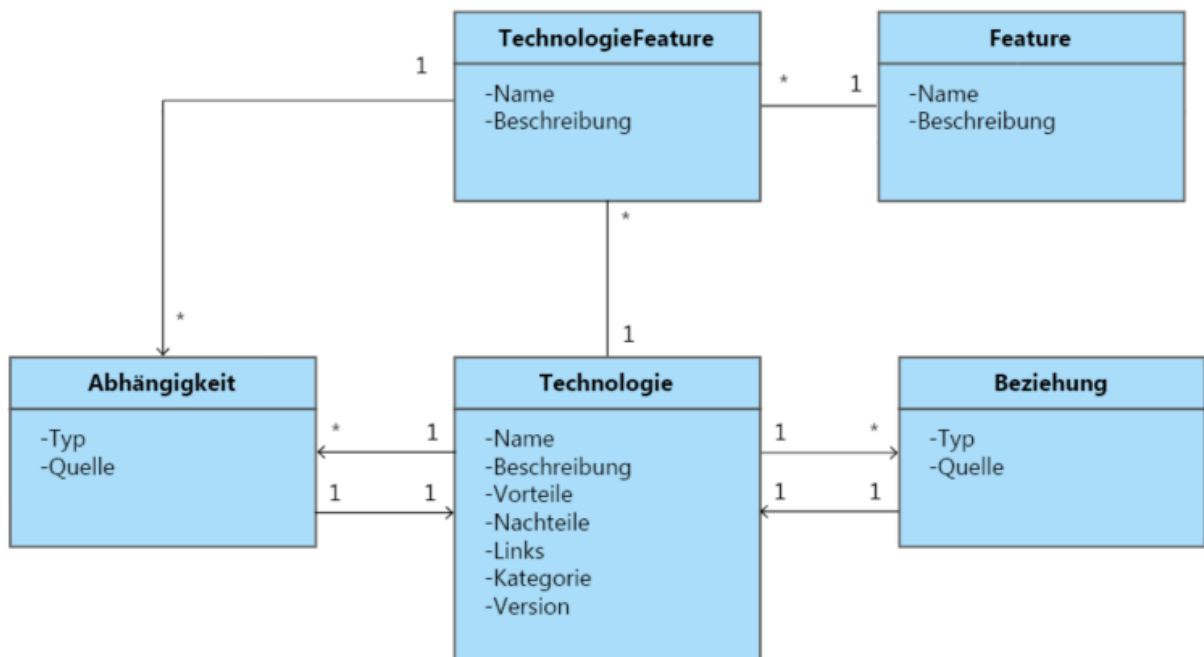


Abbildung 5: Erweitertes Metamodell

Die Erweiterungen aus Abbildung 5 werden nachfolgend erläutert. Die Abhängigkeitsbeziehung von Technologie zu Technologie wurde aufgehoben und dafür das Element

Abhängigkeiten hinzugefügt, welches den Typ und die Quelle der Abhängigkeit beschreibt. Zudem besitzt das Element Technologie nun noch weitere Eigenschaften, und auch Features haben eine Beziehung zu Abhängigkeiten.

Um das erweiterte Metamodell zu validieren, werden nachfolgend exemplarisch Frontend-Technologien selektiert und Technologiewissen für diese erfasst. Dadurch sollen Stärken und Schwächen des Modells aufgezeigt werden.

4.2 Selektion expliziter Frontend-Technologien

Wegen des hohen Aufwands einer Technologieanalyse und der großen Auswahl an Frontend-Technologien können entweder nur einige wenige detailliert untersucht werden oder viele nur allgemein. Da für eine Empfehlung jedoch die gesamte Technologielandschaft relevant ist, werden einige wenige Technologien selektiert und analysiert, um das Technologiewissen exemplarisch zu erheben.

Für das exemplarische Technologiewissen wird bei der Eingrenzung der Frontend-Technologien die Bedingung getroffen, dass die ausgewählten Frontend-Technologien mit einer SAP-HANA-Datenbank ohne weiteres Backend funktionieren müssen.

Da im SAP-Umfeld für den Zweck der Webentwicklung standardmäßig auf SAPUI5 gesetzt wird und die Entwickler der Firma INIT bereits damit arbeiten, wird in dieser Arbeit als erstes **SAPUI5** untersucht. Von SAPUI5 gibt es auch eine Open-Source-Version OpenUI5. Da diese aber sehr ähnlich zu SAPUI5 ist, wird sie als weitere Technologie ausgeschlossen. Um einen Vergleich zu einem anderen JavaScript-Framework zu ziehen, wird **AngularJS** als weitere Technologie analysiert. Um auch eine Technologie mit einer anderen Programmiersprache zu verwenden, wird das **Play Framework** ausgewählt, welches Java und/oder Scala verwendet. Neben diesen Technologien werden auch die Technologien, die sich als Abhängigkeiten aufzeigen, weiter untersucht.

4.3 Eigenschaften der Technologien

Nachfolgend werden die Technologien SAPUI5, AngularJS und das Play Framework auf ihre Kriterien untersucht. Dazu werden, wie bei der Analyse der Abhängigkeiten, auch die Literatur zu den jeweiligen Technologien, Build-Dateien, Dokumentationen sowie

Stack-Overflow-Einträge analysiert. Aus Gründen der Übersichtlichkeit werden nicht alle Abhängigkeiten näher erläutert¹⁰.

Neben den direkten Abhängigkeiten der Technologien werden diese auch nach möglichen Features untersucht. Für weitere Informationen zu der Thematik der Features und der Extraktion von Features kann die Arbeit „*Identifikation von Technologie-Features in existierenden Softwaresystemen*“ von Fechner [2016] herangezogen werden.

Da SAP HANA über JDBC, ODBC, JSON und OData zugänglich ist [openSAP,2017], müssen die ausgewählten Frontend-Technologien über mindestens eine dieser Möglichkeiten verfügen.

4.3.1 SAPUI5

Untersucht wird die zum Zeitpunkt der Arbeit aktuellste Version SAPUI5 1.46.6. SAPUI5 ist in Verbindung mit SAP HANA standardmäßig nutzbar. Es ist möglich, die SAPUI5-Applikation auf einem externen Frontend-Server zu lagern und per SAP NetWeaver Gateway mit dem SAP-Backend zu kommunizieren, welches in Verbindung mit einer SAP-HANA-Datenbank steht. Da es in dieser Arbeit nur um die Verbindung mit der Datenbank SAP HANA und nicht um das SAP-Backend geht, wird die zweite Variante genauer betrachtet. Hierbei wird SAPUI5 zusammen mit HANA XS Advanced genutzt [Engelbrecht u. Wegelin, 2016], welcher den Server für die SAPUI5-Applikation direkt auf der SAP-HANA-Datenbank darstellt. Da SAPUI5 standardmäßig mit der Kommunikation mittels OData ausgestattet ist, können die OData-Services sofort zur Kommunikation mit SAP HANA genutzt werden.

Abhängigkeiten

In der Literatur werden vorwiegend diejenigen Abhängigkeiten erläutert, auf denen SAPUI5 basiert: HTML5, CSS3 und JavaScript bzw. die Bibliothek jQuery [Antolovic, 2016]. Auch der Zusammenhang mit SAP Fiori wird häufig erläutert. Hierbei handelt es sich um eine SAPUI5-Anwendung, welche spezifische Gestaltungsrichtlinien erfüllt und deshalb als SAP Fiori bezeichnet wird [Engelbrecht u. Wegelin, 2016]. Auch die Kommunikation wird beschrieben, so kann SAPUI5 Daten mittels AJAX an den Webserver senden und mittels OData, JSON und XML Daten empfangen und übermitteln.

¹⁰ Alle untersuchten Abhängigkeiten können im entwickelten Prototypen anhand eines Graphens betrachtet werden, siehe Kapitel 5.

Da SAPUI5 noch relativ neu ist und aus dem SAP-Umfeld stammt, existieren bisher kaum GitHub-Projekte, die mit Maven erstellt wurden; lediglich ein Projekt wurde bei der Suche gefunden. Allerdings wurden aufgrund der geringen Anzahl an Projekten auch Projekte auf Abhängigkeiten untersucht, welche ohne Build-Werkzeug erstellt wurden.

Um ein Projekt als ein SAPUI5-Projekt zu nutzen, muss zunächst SAPUI5 geladen und initialisiert werden:

```
<script src=http://<server>>:<port>>/resources/sap-ui-  
core.js"  
id="sap-ui-bootstrap"  
data-sap-ui-them="sap.belize"  
data-sap-ui-libs="sap.m">  
</script>
```

Quellcode-Beispiel 3: Initialisierung SAPUI5

SAPUI5-Projekte nutzen immer interne Bibliotheken, welche wie folgt eingebunden werden:

```
<mvc:View  
controllerName="sap.ui.unified.sample.MenuItemEvent-  
ing.MenuItemEventing"  
xmlns:l="sap.ui.layout"  
xmlns:u="sap.ui.unified"  
xmlns:mvc="sap.ui.core.mvc"  
xmlns="sap.m">
```

Quellcode-Beispiel 4: Einbindung interner Bibliotheken

Diese Beispiele zeigen, wie interne Bibliotheken eingebunden werden; so nutzt dieses Projekt die Bibliotheken: sap.ui.layout, sap.ui.unified sap.ui.core.mvc und sap.m.

Diese Bibliotheken sollten nur dann eingebunden werden, wenn sie auch genutzt werden. Es sind also Feature-Direktionale, welche vorwiegend auf den verwendeten UI-Elementen basieren.

Externe Bibliotheken können zum einen in der Index-Datei manuell eingebunden werden. Zum anderen werden manche externen Bibliotheken auch automatisch geladen: SAPUI5 bietet die Kommunikation mittels OData an, wozu keine eigenen Implementierungen nötig sind und auch keine Abhängigkeiten bestehen, die explizit heruntergeladen werden müssen. Trotzdem nutzt SAPUI5 für die Verwendung von OData-Services auch externe Bibliotheken. So wird automatisch die Bibliothek **Data.js** eingebunden und aufgerufen, siehe Abbildung 6. Da diese Abhängigkeit nicht immer verwendet

wird, handelt es sich hierbei um eine Feature-Direktionale, die nur in Verbindung mit dem Feature OData-Datenkommunikation eingebunden wird.

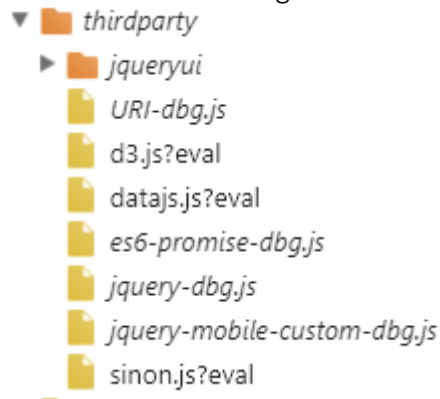


Abbildung 6: Verwendung von Data.Js in SAPUI5-Projekten

Die Dokumentation von SAPUI5 ist sehr umfangreich und bietet dadurch gegenüber anderen Technologien einen Vorteil, da Entwickler bei Fehlern auch die Dokumentation zu Rate ziehen können.

Allerdings wird auf die genauen Abhängigkeiten von SAPUI5 in der Dokumentation kaum eingegangen. Es sind jedoch unter Lizenzen die verschiedenen Third-Party-Technologien zu finden, welche SAPUI5 auf der Webseite verwendet. Viele davon werden allerdings nur dann genutzt, wenn bestimmte Features von SAPUI5 genutzt werden (siehe Data.js). Welches Feature welche Technologie nutzt, wird dagegen nicht dokumentiert.

Stattdessen stellt die Dokumentation dar, dass nicht alle eigenen Bibliotheken miteinander verwendet werden können, siehe Abbildung 8:

Abbildung 7 zeigt, dass jede Bibliothek auf der linken oder rechten Seite mit Bibliotheken aus der Mitte zu nutzen ist. Allerdings dürfen die äußeren Bibliotheken nicht miteinander genutzt werden. Warum diese Inkompatibilität besteht, wird nicht weiter erläutert.

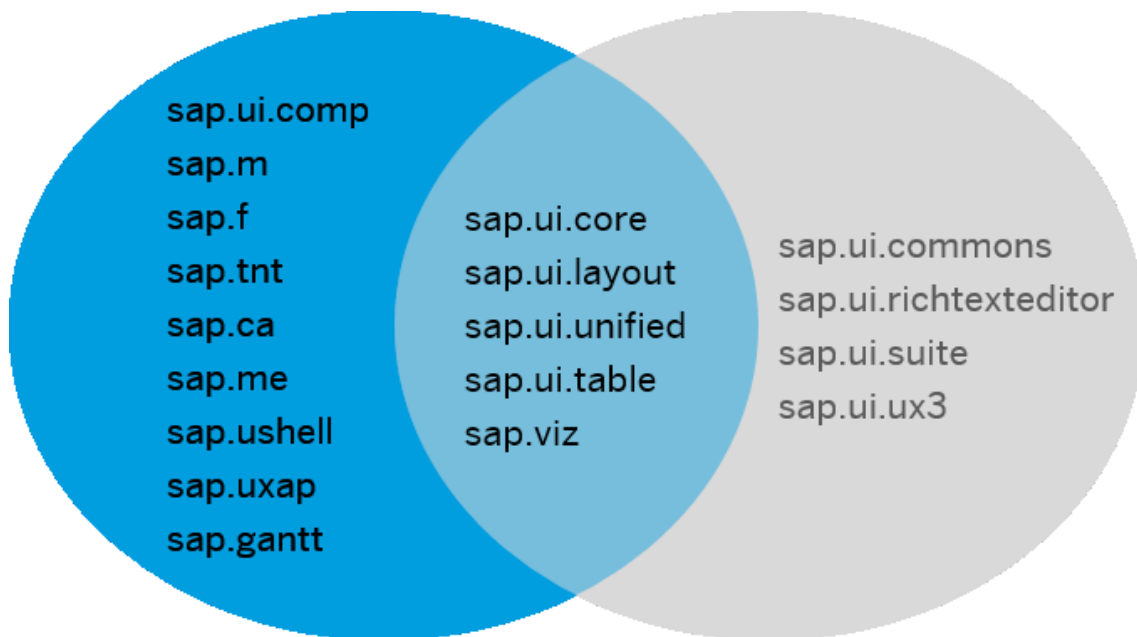


Abbildung 7: SAPUI5-Bibliotheken [SAPUI5, 2017]

Insgesamt werden laut Dokumentation 39 Third-Party-Technologien¹¹ in SAPUI5 genutzt [SAPUI5, 2017].

Für die Analyse bei Stack Overflow wurden 75 Threads untersucht. In diesen werden vorwiegend Abhängigkeiten der Kompatibilität geklärt. So wird unter anderem aufgeführt, dass SAPUI5 mit datajs, AngularJs und PhantomJs in Verbindung mit QUnit kompatibel ist. Diese Kompatibilität ist nicht verwunderlich, da SAPUI5 auf JavaScript aufbaut und diese Technologien ebenfalls, weshalb sie generell kompatibel sein sollten.

Features

SAPUI5 wurde nach dem Ansatz Mobile First entwickelt und bietet deshalb die vielfältigen 226 UI-Steuerungen, welche nach Hardware gruppiert sind (Telefon, Tablets und Desktop-PCs) [SAPUI5, 2017] und als ein wesentliches Feature betrachtet werden sollten. Zudem werden 627 Icons mitgeliefert, welche einfach verwendet werden können [SAPUI5, 2017].

Die standardmäßige Unterstützung der MVC-Architektur ist ebenfalls ein wesentliches Feature von SAPUI5 [Antolovic, 2016]. Des Weiteren kann zwischen den verschiedenen Ansichten XML, HTML, JavaScript und JSON gewählt werden. Die Datenanbindung ist sowohl mit OData als auch mit JSON- oder XML-Modellen möglich. Durch den Internationalisierung Standard L18n und einen Rechts-nach-Links-Sprachsupport wird die Internationalisierung unterstützt.

¹¹ Eine vollständige Liste, dieser Abhängigkeiten befindet sich im Anhang A.8

Da SAPUI5 in der Regel die jQuery-Bibliothek (auch ohne jQuery möglich) benutzt, können die Features von jQuery ergänzend zu denen von SAPUI5 gesehen werden, wie:

- Manipulation des Document Object Models
- Animationen und Effekte
- Ajax-Funktionalität

Zudem werden auch jQuery Mobile und jQueryUI automatisch eingebunden, welche SAPUI5 um weitere Features ergänzen, vorwiegend im Bereich der Benutzung von mobilen Anwendungen [SAPUI5, 2017].

Randbedingungen

In der Dokumentation werden auch Informationen zu den Randbedingungen von SAPUI5 genannt, so wird eine Matrix für die verschiedenen Browser und Plattformen aufgestellt, welche von SAPUI5 unterstützt werden (siehe Anhang A.8).

Vor- und Nachteile

Neben den vielen Features, die SAPUI5 bietet, zählt zu den Vorteilen vor allem die ausführliche Dokumentation, welche auch Tutorials und Beispielanwendungen beinhaltet. Auch die große Auswahl an Steuerungselementen und Icons sowie die gut ausgearbeitete User Experience können positiv gesehen werden.

Ein großer Vorteil gegenüber anderen Technologien ist jedoch die standardmäßige Kommunikation mit ODATA-Services, da hierfür nichts mehr implementiert werden muss.

Ein großer Nachteil ist bei SAPUI5 die Erweiterbarkeit. So sind andere Technologien wesentlich einfacher anzupassen; dadurch wirkt SAPUI5 recht starr. Zudem ist der Zugang zu Informationen aufgrund der SAP-Zugehörigkeit begrenzt; für vieles werden teure Lizenzen benötigt. SAPUI5 wird mit anderen Produkten zusammen angeboten, sodass keine genaue Lizenzgebühr für SAPUI5 genannt werden kann. Allerdings ist es auch möglich, OpenUI5 zu nutzen, für das keine Lizenzgebühren anfallen. OpenUI5 unterscheidet sich von SAPUI5 dadurch, dass der Open-Source-Variante einige Features fehlen.

4.3.2 AngularJs

AngularJs ist ein Framework, welches von Google Inc. entwickelt wurde. Untersucht wurde die AngularJs-Version 1.6.5. Es wurde bewusst nicht auf den Nachfolger Angular gesetzt, da AngularJs, ebenso wie SAPUI5, ein JavaScript-Framework nutzt. Der Nachfolger baut stattdessen auf TypeScript auf. Um AngularJs mit SAP HANA zu verwenden, kann auf die Datenübermittlung durch JSON zurückgegriffen werden.

Abhängigkeiten

Die Templates von AngularJs-Anwendungen werden mit HTML beschrieben. Dadurch ist HTML eine direktionale Abhängigkeit von AngularJs, da dies immer in Verbindung mit AngularJs verwendet wird. So werden dem üblichen HTML5 neue Tags, die Direktiven von AngularJs, hinzugefügt, um dessen Features zu nutzen [Green u. Seshadri, 2013].

Zudem basiert AngularJs, wie bereits erwähnt, auf JavaScript [Jain et al., 2014].

In dem Paper von Balasubramanee et al. [2013] wird die Verwendung von AngularJs und Bootstrap beschrieben. Dies zeigt, dass die beiden Technologien miteinander kompatibel sind und AngularJs durch Bootstrap um eine bessere User Experience und UI-Elemente erweiterbar ist.

In Build-Dateien wird Angular häufig mit Modulen und Bibliotheken verwendet, welche nur für AngularJs verwendet werden können. Dies wären dann Erweiterungen, die AngularJs auch als direktionale Abhängigkeiten besitzen und die durch AngularJs auch ermöglicht werden. Zudem werden in Projekten häufig AngularJs und Bootstrap zusammen verwendet, um das User Interface ohne viel Zeitaufwand im Bereich CSS zu gestalten. Dies wird auch in der Dokumentation dargestellt [angularjs Docs, 2017].

Die Dokumentation von AngularJs ist ähnlich umfangreich wie die SAPUI5-Dokumentation. Es werden viele kompatible Technologien aufgelistet, inklusiver vorhandener Tutorials aus dem Netz. Insgesamt werden in der Dokumentation 43 Technologien benannt, welche in Verbindung mit AngularJs verwendet werden können¹².

Es ist neben der Datenübermittlung durch JSON auch möglich, mittels OData mit einer SAP-HANA-Datenbank zu kommunizieren. Dazu benötigt AngularJs entweder DataJs oder das BreezeJs als Erweiterung [H., 2015]. Jedoch können die Services nicht ohne

¹² Siehe Anhang A.9

Weiteres verwendet werden. Jegliche Responses müssen manuell geparkt werden, wie sich in Projekten der INIT Solution herausstellte.

Features

Eines der Features von AngularJS ist der Aufbau nach dem MVC-Muster [Green u. Seshadri, 2013]. Ein weiteres wichtiges Feature ist die Zwei-Wege-Datenanbindung und die Möglichkeit, Animationen zu erstellen [AngularJs, 2017].

Vor- und Nachteile

Ein klarer Vorteil, vor allem gegenüber SAPUI5, ist die Erweiterbarkeit. Durch die Erweiterung etwa mit d3.js können wesentlich schönere Diagramme erstellt werden als es standardmäßig möglich wäre. Zudem kann die Appstruktur verändert und selbst definiert werden.

Allerdings liegt ein großer Nachteil bei der Verwendung von AngularJs mit SAP HANA darin, dass die Anwendung zwar mit OData-Service kommunizieren kann, aber alle Datentypen, die zurückkommen, selbst geparkt werden müssen. Dies ist vor allem sehr zeitaufwendig und kann bei späteren Änderungen zu Fehlern führen.

Bei dem Vergleich der Abhängigkeiten von SAPUI5 und AngularJs fällt auf, dass SAPUI5 jegliche Abhängigkeiten selbst steuert, sobald eine interne Bibliothek verwendet wird (siehe DataJs). Bei AngularJs dagegen müssen die Bibliotheken nicht nur eingebunden, sondern auch dem Projekt manuell hinzugefügt werden, ebenso wie weitere Abhängigkeiten, die die Bibliothek möglicherweise aufweist. Dieses Merkmal könnte bei späteren Arbeiten mit in die Klassifizierung aufgenommen werden.

4.3.3 Play

Die Analyse des Play Frameworks basiert auf der zum Zeitpunkt der Arbeit aktuellsten Version 2.6.2. Play ist inspiriert von Ruby on Rails und wurde in Scala geschrieben¹³. Es erleichtert die Entwicklung von Web-Applikationen mit Scala und Java. Es ist also möglich, eine Anwendung sowohl in Java als auch in Scala zu implementieren.

Auch das Play Framework ist nach dem MVC-Muster aufgebaut und wird von den Entwicklern der Sprache Scala als Framework empfohlen.

¹³ die Version 1.0 noch in Java

Abhängigkeiten

In der Literatur wird vorwiegend auf den Vorteil eingegangen, dass sowohl in Scala als auch in Java programmiert werden kann. Da Play in Scala geschrieben wurde, basiert es auch auf Scala und verwendet Java als Feature-Direktionale [Ward u. Leroux, 2014]. Bei Play ist vor allem die Nutzung von Java bzw. Scala sehr speziell, da man sich entweder für eine Sprache entscheiden kann oder auch beide nutzen kann [Play, 2017], dies stellt eine Feature-Direktionale da. Wenn es sich dabei nicht um ein Feature handeln würde, könnte man es auch als Austauschbare-Direktionale verstehen.

Die Dokumentation von Play zeigt vorwiegend die Kombination mit Modulen auf, die ähnlich wie bei AngularJs nur für Play entwickelt wurden [Play, 2017]¹⁴. Da es nicht wie die anderen Technologien in JavaScript geschrieben ist, ist es teilweise schwieriger JavaScript Bibliotheken zu nutzen. Jedoch gibt es beispielsweise für das Nutzen von Twitter Bootstrap andere Implementationen um dies zu ermöglichen [Hurtado, 2017].

Features

Neben dem wohl wichtigsten Feature, dass Play sowohl mit Java als auch mit Scala verwendet werden kann, verfügt Play auch über eine MVC-Architektur. Zudem ist es sehr leistungsfähig und es besteht die Möglichkeit zur Internationalisierung [Play, 2017].

Randbedingungen

Alle Versionen des Play Frameworks ab 2.5 benötigen Java 8 JVM [Play, 2017].

Vor- und Nachteile

Der große Vorteil von Play besteht in der Möglichkeit, sich zwischen Java oder Scala zu entscheiden oder auch beides zu nutzen. So ist es je nach Kenntnisstand des Entwicklers möglich, zwischen verschiedenen Sprachen zu wechseln, wenn etwa ein Entwickler Java beherrscht und der zweite nur Scala. Zudem ist es eine Open-Source-Software und wird durch die Community mit Modulen erweitert.

Der Nachteil besteht allerdings darin, dass es einfacher ist SAPUI5 und AngularJs mit JavaScript Bibliotheken zu verändern. Nicht alle JavaScript Bibliotheken können für das Play Framework verwendet werden. Diese werden jedoch im Webbereich häufig zur Erweiterung genutzt.

¹⁴ Siehe Anhang A.10

5 Prototypische Implementierung

Nachdem im letzten Arbeitsschritt exemplarisch Technologien auf ihre Abhängigkeiten untersucht wurden, soll nun ein Prototyp zur Entscheidungsunterstützung implementiert werden. Dazu werden zunächst die Nutzerszenarien für dieses Werkzeug aufgezeigt um anhand dessen die Architektur und die Technische Implementierung dieser Funktionalitäten zu erläutern.

5.1 Nutzerszenarien

Um den Prototyp zu implementieren, wird zunächst aufgezeigt, welche Szenarien hier möglich sein sollen. Der Nutzer ist in der Regel Architekt oder – bei kleineren Unternehmen – auch der Entwickler selbst. Er hat ein spezifisches Wissen von einer vergleichsweise kleinen Menge an Technologien, mit denen er bereits gearbeitet hat. Nun bekommt er die Aufgabe, in einem bestehenden Projekt eine Technologie auszutauschen oder sich in einem neuen Projekt für eine Technologie zu entscheiden.

Gerade im Bereich der Webentwicklung gibt es zahlreiche Frameworks und Bibliotheken, sodass die Entscheidung hier schwerfällt. Es ist häufig nicht klar, welche Vor- und Nachteile die Technologien bieten, und der Vergleich von deren Features ist sehr zeitintensiv.

Das Werkzeug soll dem Architekten die Möglichkeit geben, anhand seiner Anforderungen Empfehlungen für Technologien zu erhalten, und seine Entscheidung so unterstützen. Die Technologien, die ihm empfohlen werden, kann er sich genauer ansehen. Aber auch ohne eine Empfehlung kann er sich Technologien ansehen. In beiden Fällen wird ihm eine Visualisierung der Abhängigkeiten dargestellt, eine Beschreibung, die Vor- und Nachteile der Technologie, und Links zu Dokumentationen der Technologie sowie andere nützliche Referenzen. Des Weiteren werden häufige Kombinationen, Empfehlungen durch den Entwickler, Ermöglichkeiten und Alternativen aufgezeigt. Dies ist nicht nur zur Informationsgewinnung interessant, sondern kann auch für die spätere Dokumentation hilfreich sein und dafür verwendet werden.

5.2 Architektur der Implementierung

Zur Validierung des Technologiewissens und des Nutzens eines Werkzeuges zur Unterstützung eines Architekten wird ein Prototyp implementiert, welcher das spezifische Wissen aus der Technologieanalyse beinhaltet und exemplarisch die Arbeit mit solch einem Werkzeug darstellt.

Das Technologiewissen basiert auf der Technologieanalyse aus Kapitel 4. Es wurde strukturiert in dem Datenmodell in Abbildung 6 (siehe Kapitel 4) abgelegt. Mithilfe des spezifischen Technologiewissens soll das Werkzeug exemplarisch darstellen, dass der Prozess der Technologieauswahl durch ein solches Werkzeug vereinfacht werden könnte.

Das Werkzeug wurde nach dem Model-View-Controller-Muster (MVC) implementiert, um spätere Änderungen und Erweiterungen zu vereinfachen. Bei dem MVC-Muster wird die Software in drei Teile geteilt: das Model (Datenmodell), die View (Präsentation) und den Controller (Steuerung). Die Architektur des Prototyps wird im Anhang A.11 dargestellt.

Das Datenmodell enthält die Daten des Technologienwissens für das Werkzeug und ist unabhängig von den beiden anderen Komponenten. Dies vereinfacht es, die Daten für spätere Zwecke weiterzuverwenden und nur Controller und View neu zu implementieren, wenn das Werkzeug später in der Praxis beispielsweise in Java implementiert werden soll.

5.3 Technische Implementierung

Das Werkzeug¹⁵ wurde exemplarisch als Web-Werkzeug implementiert. Diese Entscheidung beruht auf der Grundlage der Verwaltung von Architekturwissen nach Hansen et al. [1999]. Hierbei wird zwischen dem Codification-Ansatz, bei dem das Wissen für die Allgemeinheit zugänglich ist, und dem Personalization-Ansatz unterschieden, bei dem verschiedene Wissensbasen entstehen, auf die nur einzelne Individuen Zugriff haben. Für Hansen et al. [1999] sollte das Bestreben der Wissenschaft jedoch in der Zugänglichkeit für die Allgemeinheit liegen.

Dementsprechend wird die Anwendung im Internet für die Allgemeinheit zugänglich und für jeden sofort nutzbar sein, ohne langwierigeren Installationsprozess. Das Datenmodell bildet das Datenbankverwaltungssystem MySQL in Kombination mit dem

¹⁵ Eine ausführbare Version findet sich unter: <http://depends-thesis.de> und das Projekt kann auch unter Github <https://github.com/Jennyfa/masterprojekt> heruntergeladen und mithilfe der Anleitung in Anhang A.12 lokal installiert werden.

Speichersubsystem InnoDB. Es wird hierbei auf SAP HANA als Datenbank für das Datenmodell verzichtet, da hierfür im Regelbetrieb Lizenzkosten anfallen und bei einer Testversion die Datenbank nach 21 Tagen gelöscht wird.

Für das Frontend wird das Framework AngularJs in Kombination mit dem CSS-Framework Twitter Bootstrap genutzt, da AngularJs nach dem MVC-Muster verwendet werden kann und die Implementierung mit MySQL bereits bekannt war. Der Datenzugriff erfolgt mittels PHP 7.0 und durch das Dateiformat JSON.

Die Startseite (Abbildung 8) bietet die Möglichkeit, sich über die Abhängigkeiten genauer zu informieren und sich die Arbeit als PDF herunterzuladen, um sich in die Thematik einzulesen.

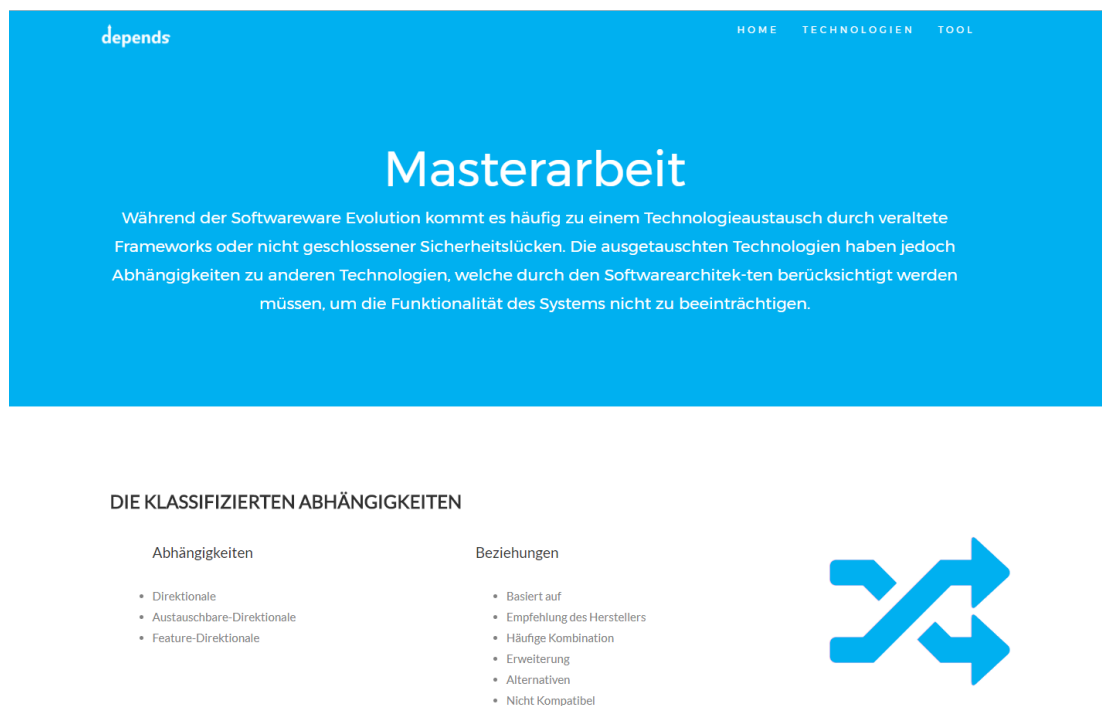


Abbildung 8: Startseite

Damit das Werkzeug seine grundlegendste Funktion der Entscheidungsunterstützung, (siehe Abbildung 9) umsetzen kann, werden im „ToolController“ zunächst die Formularefelder automatisch mit den Daten aus dem Datenmodell gefüllt.

depends

HOME TECHNOLOGIEN TOOL

ENTSCHEIDUNGSUNTERSTÜTZUNG

Handelt es sich um ein neues Projekt
oder ein bestehendes?

☐ Neuimplementierung
 ☐ Erweiterung

Welche Programmiersprachen sind
dem Team bekannt?

JavaScript
PHP
Java
Scala
...

Welche neuen Features werden
benötigt:

Analytics
Animationen
Authentication/Login
Captcha
...

Welche Technologien werden
eingesetzt:

SAPUI5
AngularJS
data.js
D3
...

Welche Features werden bereits
genutzt:

Analytics
Animationen
Authentication/Login
Captcha
...

Welche Datenbank wird genutzt:

...

Empfehlung erhalten

Zurücksetzen

Abbildung 9: Werkzeug zur Entscheidungsunterstützung

Nach der Auswahl der Anforderungen durch den Nutzer werden die definierten Anforderungen aus der View durch AngularJs an den Controller übertragen. Im Controller werden die Anforderungen abgeglichen und per PHP mittels JSON Anfragen an die Datenbank gesendet. Dadurch werden die Voraussetzungen ausgelesen und die derzeitig verwendeten Technologien mit der ausgewählten Programmiersprache und den geforderten Features verglichen. So wird überprüft, ob alle benötigten Features in den verwendeten Technologien enthalten sind. Ist dies nicht der Fall, werden für die Features weitere Technologien gesucht, welche sich nach Möglichkeit mit den ausgewählten Programmiersprachen decken. Zudem wird geprüft, ob diese auch die alten verwendeten Features nutzen. Wenn zusätzlich noch SAP HANA als Datenbank ausgewählt wurde, wird überprüft, welche Technologien über geeignete Datenbankverbindungen verfügen.

Dabei werden jegliche Möglichkeiten zunächst als Ergebnis gespeichert und bei wiederholtem Aufruf wird durch die Funktion „getRating“ deren Bewertung hochgesetzt. Danach werden mittels der Funktion „getTechCombination“ mögliche Kombinationen von Technologien überprüft und diese zusammen mit deren Rating in der Tabelle aufgelistet (siehe Abbildung 10).





depends			
HOME TECHNOLOGIEN TOOL			
Empfehlung			
Ihre Kriterien: JavaScript Analytics Angular.js Animationen			
Suche nach Technologie			
Rating	Technologie	Übereinstimmung	Details
★★★★★ 4 von 4	 Angular.js In Kombination mit: angularartics angularartics	JavaScript, Animationen Analytics Analytics	
★★★☆☆ 1 von 4	 SAPUI5	JavaScript	

Abbildung 10: Ergebnis der Abfrage

Bei diesen Ergebnissen besteht jedoch aufgrund der geringen Menge an Technologie-wissen kein Anspruch auf Vollständigkeit oder Gültigkeit.

Nachdem der Nutzer Empfehlungen erhalten hat, kann er sich weitere Details zu der empfohlenen Technologie ansehen und sich genauer über deren Abhängigkeiten informieren, indem er auf den Button unter „Details“ klickt. Dadurch wird der „Detail-Controller“ aufgerufen, welcher die Inhalte für die Detailseite aus der Datenbank ausliest. Hierbei werden alle gespeicherten Informationen aus der Tabelle Technologien für die jeweilige Technologie und deren Beziehungen ausgelesen. Dazu werden unter Kombinationen und Alternativen jedoch nicht nur die Technologien ausgelesen, für die diese Beziehungen definiert sind, sondern auch diejenigen, bei denen die Technologie als „dependsOn“ definiert ist (siehe Abbildung 11). Dadurch müssen Alternativen und Kombinationen nur jeweils einmal definiert werden und keine redundanten Informationen gespeichert werden.










				b_id	b_type	b_source	t_id	dependsOn
				14	Alternative	Dokumentation	1	19
				43	Kombination	Projekte	1	54

Abbildung 11: Auszug aus Datenbank-Tabelle Beziehung

Zudem wird in der View noch der Controller für die Visualisierung, „VisuController“, aufgerufen. Dieser fasst alle Abhängigkeiten der Technologie zusammen und erstellt mithilfe der D3-Bibliothek eine Visualisierung (siehe Abbildung 12).

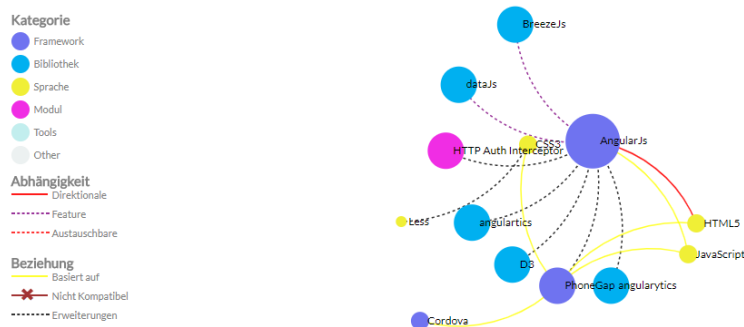


Abbildung 12: Visualisierung der Abhängigkeiten

Durch die Verwendung der Beziehung „Ermöglichung“ kann bei diesem Graphen ein Zyklus vermieden werden. Würde der Controller alle direktionalen Abhängigkeiten für eine Technologie auslesen und umgekehrt, wo diese eine direktionale Abhängigkeit darstellt, so würde ein Zyklus entstehen. Dies könnte das Werkzeug zum Absturz bringen. Deshalb kann mithilfe der Beziehung „Ermöglichung“ die Beziehung ausgelesen werden und alle weiteren Abhängigkeiten, die darauffolgen, werden außer Acht gelassen und nicht ausgelesen.

Das Routing der Seite erlaubt es, jede Detailseite für eine bestimmte Technologie als Lesezeichen zu speichern, inklusive des vollständigen Namens der Technologie, welche gerade angezeigt wird. Dadurch ist das Lesezeichen später leichter wiederzufinden und die URLs können auch in Dokumentationen eingebunden werden.

Zudem ist es durch den „DbController“ möglich, sich alle Technologien als Tabelle anzeigen zu lassen. Dieser liest aus der Datenbanktabelle Technologien alle vorhandenen Einträge aus (siehe Abbildung 13).





















depends			HOME	TECHNOLOGIEN	TOOL
VORHANDENE TECHNOLOGIEN					
Suche nach Technologie					
Technologie	Kategorie	Details ansehen			
Ace	library				
angular-mobile-nav	library				
 Angular.js	Framework				
angularartics	library				
angularytics	library				
 Breeze.js	library				
Canvg	library				
 Cordova	Framework				
Crossroads	library				
 CSS3	language				
 D3	library				
data.js	library				
es6-promise	library				
Esprima	library				
fastclick.js	library				

Abbildung 13: Alle Technologien, die in der Datenbank vorhanden sind

6 Feedback und Bewertung

Um den entwickelten Prototyp und die Klassifizierung der Abhängigkeiten zu bewerten, soll Feedback von Entwicklern der INIT Solution GmbH eingeholt werden.

Das Feedback soll durch eine offene Befragung der Entwickler stattfinden. Hierzu wird ihnen eine knappe Einführung in die Thematik gegeben und eine Aufgabe, welche sie durch die Zuhilfenahme des Werkzeuges bearbeiten sollen. Nach der Verwendung des Werkzeuges werden ihnen offene Fragen gestellt¹⁶. Diese offenen Fragen dienen der Möglichkeit, den Entwicklern Raum für Kritik und Verbesserungsvorschläge zu geben. Das Ziel dieser Befragung ist es, den Prototyp zu bewerten und so Möglichkeiten zur Verbesserung des Werkzeuges zur Entscheidungsunterstützung aufzuzeigen sowie Schwächen bei der Klassifizierung der Abhängigkeiten zu identifizieren und Aufschluss über die Allgemeingültigkeit des Konzepts zu geben.

Dadurch soll es späteren Arbeiten ermöglicht werden, diese Schwächen zu verhindern und das Werkzeug zielführender zu entwickeln.

Die Hypothese zu der Befragung lautet, dass das Werkzeug den Entwicklern zwar hilft, die Empfehlungen jedoch aufgrund des begrenzten Technologiewissen noch nicht realistisch sind und die Allgemeingültigkeit darunter leidet.

Die Befragung wurde mit sechs Entwicklern durchgeführt. Die Entwickler sind so ausgewählt, dass diese ein unterschiedliches Wissen in den Bereichen Webentwicklung, Frontend-Technologie und SAP HANA besitzen, um ein möglichst umfassendes Feedback zur über die Qualität der Empfehlung durch den Prototyp zu erhalten.

Der Ablauf der Befragung ist wie folgt gegliedert:

- Den Entwicklern wird eine kurze Einführung in die Thematik gegeben
- Ihnen wird daraufhin eine Aufgabe gestellt, welche sie mit Hilfe des Werkzeuges lösen sollen
- Wenn sie fertig sind, werden ihnen offene Fragen gestellt, wie beispielsweise:
 - Wie Ihnen das Werkzeug hilft?
 - Wie das Werkzeug verbessert werden könnte?
 - Wie viel Zeit Sie für die Arbeit manuell benötigen würden?

¹⁶ Aufgabe und Fragebogen inklusive der Antworten der Entwickler, siehe Anhang A.12

Das Feedback der Entwickler¹⁷ zu dem Prototyp ergibt ein einheitliches Bild, obwohl die Entwickler verschiedene Wissensstände aufweisen.

Zum einen hat sich die Hypothese bestätigt, dass der Umfang als zu klein erachtet wird und die Entwickler dies bei der Empfehlung bzw. auch bei der Auswahl von Anforderungen bemängeln. Die Verbesserungsvorschläge werden nachfolgend erläutert.

Für die Verbesserung der Empfehlungen durch den Prototyp bitten die Entwickler auch um zusätzliche Auswahlkriterien:

- Eine dynamische Anpassung der Gewichtung für die Auswahlkriterien durch den Anwender
- Die Möglichkeit, bestimmte Technologien generell ausschließen zu können
- Nur Open-Source-Technologien in der Empfehlung zu erhalten
- Angaben zur Hardware machen zu können
- Auswählbarkeit von Offline-Verhalten.
- Eine Angabe darüber, für welches Gerät, die Anwendung optimiert werden soll

Vorschläge zur Verbesserung und Erweiterung des dargestellten Wissens:

- Man sollte das Werkzeug der Community zur Verfügung stellen, um möglichst viel Wissen zu erhalten.
- Eine Möglichkeit für den Nutzer, selbst Anmerkungen zu den Technologien zu machen, eine Art Kommentarfunktion
- Eine detailliertere allgemeine Beschreibung der Technologien
- Darstellung als Beispiel-Code, wie etwa die Datenbankbindung umgesetzt wird

Die Usability des Prototyps wurde zwar stark gelobt und häufig betont, dass dieser sehr einfach zu bedienen sei, trotzdem gab es einige Verbesserungsvorschläge:

- Das Gelb in der Visualisierung ist etwas schwer erkennbar.
- Das man im Werkzeug, wenn man bereits Empfehlung erhalten hat, wieder zurück zu seinen Anforderungen gelangen kann.
- Hilfetexte,
 - wie man die Mehrfachauswahl des Werkzeuges nutzt;
 - wie man die Auswahl rückgängig macht;
 - was die Features jeweils bedeuten;
 - weshalb der Unterschied zwischen Neuimplementierung und Erweiterung in den Auswahlmöglichkeiten entsteht;

¹⁷ Antworten siehe Anhang A.12

-
- was die unterschiedlichen Abhängigkeiten in der Visualisierung bedeuten.
 - Die Visualisierung ist manchmal etwas unübersichtlich, da die Knoten für die einzelnen Technologien, sich manchmal überschneiden, jedoch hilft es, diese zu bewegen.
 - Den Namen auch als Link benutzen zu können, wäre in der Liste der Technologien intuitiver (Verbessert).
 - Suchfeld im Multiselect und Vergrößerung der Auswahlfenster.
 - Alphabetische Sortierung (Verbessert).

Des Weiteren wurden dabei einige Fehler entdeckt:

- Das Rating funktioniert manchmal nicht richtig.
- Die Seite springt zur Startseite, wenn sie wieder nach oben scrollen soll (Verbessert).
- Im Internet Explorer wird die Visualisierung nicht dargestellt (Kompatibilitätsproblem).

Dieses Feedback sollte für weitere Arbeiten in diesem Bereich genutzt werden, um Entscheidungsunterstützungen weiter zu verbessern.

Insgesamt kann zur Bewertung des Werkzeugs gesagt werden, dass die Entwickler einheitlich sehr positiv auf den Prototyp reagiert haben. Sie waren sehr angetan von der einfachen Bedienung des Werkzeuges und der Visualisierung der Abhängigkeiten.

Sie gaben an, sie würden das Werkzeug bei Projektstarts, Updates oder Erweiterungen in einem Projekt verwenden, wodurch ihre Arbeit mehrmals im Jahr stark vereinfacht werden könnte und Fehlentscheidungen reduziert werden könnten. Der Aufwand für die Beschaffung dieser Informationen zu den Abhängigkeiten einer Technologie wurde von den meisten Entwicklern auf mehrere Tage für eine Technologie geschätzt. Dass mehrere Technologien zunächst verglichen werden müssen, stellt somit einen erheblichen Kostenfaktor dar. Zudem werden diese Informationen durch die Entwickler häufig durch Googeln und durch das Wissen von Kollegen beschafft, was sich ebenfalls sehr zeitaufwendig gestalten kann.

Des Weiteren könnte sich ein Entwickler das Werkzeug auch als gute Informationsquelle für neue Technologien vorstellen, welche ihm noch nicht bekannt sind. Dies

deckt sich mit den Meinungen der anderen Entwickler, die sich gern mit den Abhängigkeiten von Technologien auseinandersetzen würden, wenn diese so ansprechend aufbereitet wären wie im Prototyp.

Das Werkzeug erweist sich also auch bei einer geringen Menge an Technologiewissen als nützlich und die Entwickler können sich ein Werkzeug mit einer breiten Masse an Wissen zu den verschiedensten Technologien durch den Prototyp gut vorstellen. Sie würden ein Werkzeug dieser Art gern verwenden, wodurch auch die Allgemeingültigkeit für andere Bereiche außer dem der Webtechnologien bewiesen werden kann.

7 Fazit und Ausblick

Abschließend sollen die Ergebnisse der Arbeit noch einmal zusammengefasst werden und deren Stärken und Schwächen dargestellt werden, sowie ein Ausblick für weitere Arbeiten zu der Thematik gegeben werden.

7.1 Fazit einer Entscheidungsunterstützung

Im Mittelpunkt dieser Arbeit stand die Analyse der Zusammenhänge zwischen Technologien bestehender Anwendungen bei Entwurfsentscheidungen während der Softwareevolution sowie die Ermittlung von Informationen zu exemplarischen Technologien. Dadurch sollten die Fragen beantwortet werden, welche Abhängigkeiten aus der Literatur sich in anderen Quellen wie Dokumentationen, Build-Dateien und Foren wiederfinden lassen und wie diese Abhängigkeiten in den verschiedenen Quellen abgebildet werden.

Die Ergebnisse zeigen, dass sich Abhängigkeiten aus der Literatur nicht vollständig in den anderen Quellen wiederfinden lassen. Dies ist auf die Allgemeingültigkeit der Klassifizierung aus der Literatur zurückzuführen: Weshalb im speziellen Bereich der Technologieentscheidungen nicht alle Abhängigkeiten vorhanden sind. Zudem zeigt sich, dass Abhängigkeiten in der Praxis bestehen, welche in der Literatur bisher nicht beschrieben wurden.

Gezeigt hat sich zudem, dass die klassifizierten Abhängigkeiten in den Quellen unterschiedlich abgebildet werden. In Build-Dateien gibt es zunächst keine eindeutige Unterscheidung zwischen den Abhängigkeiten. Nur im Kontext mit anderen Build-Dateien ist ein Aufschluss darüber möglich, um welche Art der Abhängigkeit oder Beziehung es sich handelt. Dokumentationen dagegen verweisen meist schon auf die Verwendung der Abhängigkeit und geben so einen klaren Anhaltspunkt um die Abhängigkeit benötigt wird oder nicht. In Foren werden häufig optionale Abhängigkeiten erläutert und auch Empfehlungen gegeben, welche Technologien gut mit einer funktionieren.

Diese Arbeit hat gezeigt, dass Architekten durch ein Werkzeug zur Entscheidungsunterstützung hilfreich unterstützt werden können. Sie würden Zeit und somit auch Geld in Projekten einsparen und unnötige Fehler vermeiden.

Bei der Analyse für das exemplarische Technologiewissen konnten auch Schwächen identifiziert werden. So könnte die Klassifizierung der Feature-Direktionalen noch unterschieden werden in automatische und manuelle Feature-Direktionalen, da manche Technologien die Abhängigkeiten automatisch herunterladen und bei anderen diese manuell hinzugefügt werden müssen.

Auch durch die Befragung der Entwickler konnten die Grenzen der Arbeit aufgezeigt und Verbesserungen für zukünftige Arbeiten identifiziert werden, welche im Folgenden näher erläutert werden.

7.2 Zukünftige Relevanz von Entscheidungsunterstützungen für Technologien

Für zukünftige Forschungsarbeiten wurden in der Arbeit bereits an einigen Stellen potenzielle Themen aufgezeigt. Diese sollen hier noch einmal verdeutlicht werden.

Der wichtigste Schritt, um den Nutzen des Werkzeuges in der Praxis zu erhöhen, wäre die Erweiterung des Technologiewissens für den Prototyp. Denn nur durch mehr Wissen können exaktere und hochwertigere Aussagen getroffen werden. Zudem kann durch eine Erweiterung an Technologiewissen in folgenden Arbeiten ein Anspruch auf Allgemeingültigkeit erhoben werden.

Für die Erweiterung des Wissens sollte möglichst eine automatisierte Variante entwickelt werden, um Build-Dateien nach Abhängigkeiten zu untersuchen. Dadurch wäre es auch denkbar darzustellen, welche Technologien häufig oder gar nicht miteinander verwendet werden. Dies wäre höchstwahrscheinlich auch ein Ansatz, um nicht kompatible Technologien aufzuzeigen. Durch diese Automatisierung könnte der Prototyp eine Art Recommendation-System wie bei dem Onlinehändler Amazon aufbauen. Dieses System könnte dem Architekten zeigen, welche Technologien von anderen Architekten häufig miteinander benutzt werden, wie etwa:

Von 1 000 Projekten nutzen 950 Technologie A und nur 50 Technologie B.

Zudem könnten die transitiven Abhängigkeiten, welche bei Maven automatisch heruntergeladen werden, für weitere Forschungsarbeiten sinnvoll genutzt werden, um damit automatisiert Abhängigkeiten von Technologien zu analysieren.

Neben dieser Automatisierung könnte auch das Wissen aus Stack Overflow automatisiert in das Werkzeug einfließen. Dazu kann Stack Exchange genutzt werden. Im Rahmen dieser Arbeit stieß die Suche nach einer gültigen Abfrage an ihre Grenzen, da die

Ergebnisse zu viele nicht relevante Einträge besaßen. Jedoch könnte untersucht werden, ob eine Automatisierung durch eine Abfrage möglich wäre, die nur relevante Ergebnisse liefert, mit denen anschließend das Werkzeug befüllt werden könnte.

Auch Verbesserungen im Bereich der Usability wären sinnvoll, um das Werkzeug intuitiver und dadurch effizienter zu gestalten und den Architekten besser zu unterstützen und einzubinden. Hierzu wäre es sinnvoll, die Vorschläge der Entwickler einzubinden und unter anderem eine Kommentarfunktion einzurichten, die es dem Architekten ermöglicht, Anmerkungen zu den gelieferten Ergebnissen oder auch zu bestimmten Technologien zu machen, die anderen Entwicklern helfen. In einem weiteren Schritt könnten auch diese Anmerkungen bei der Empfehlung eine Rolle spielen.

Da Dokumentationen während des Entwicklungsprozesses häufig zu kurz kommen, könnten die Architekten sich möglicherweise die Daten zu den Technologien herunterladen. Dabei wäre es auch sinnvoll, den Abhängigkeitsgraphen der Technologie als Bild herunterladen zu können. Dadurch könnten verschiedene Notwendigkeiten auch anderen Stakeholdern besser erklärt werden.

Um überhaupt einen Überblick darüber zu erhalten, wie Interessant das Thema in der Forschung und auch der Praxis ist, wäre es sinnvoll, ein Analysewerkzeug wie Google Analytics einzubinden. So könnte beispielsweise der Verbleib auf der Seite analysiert werden, um dadurch die häufig verwendeten Funktionen zuerst zu verbessern.

Neben den bereits implementierten Funktionen sind noch weitere denkbar, wie:

- Vergleich zwischen Technologien (per Klick auswählbar)
- Kundenbewertung der Technologie (auch mit in der Ergebnistabelle verzeichnen)
- Scope mit aufnehmen
- Das Konzept von Tobias Fechner zur Feature-Extraktion mit dem Werkzeug verbinden
- Softwarequalitätsattribute zu den Technologien analysieren

Der Vorschlag, Quellcode-Beispiele darzustellen, könnte durch die Einbindung von Hello-World-Anwendungen der jeweiligen Technologie umgesetzt werden. So können deren Abhängigkeiten direkt in der Praxis dargestellt werden. Dazu könnte das Projekt unter <http://todomvc.com/> genutzt werden, welches für verschiedene Frameworks bereits eine Anwendung für To-do-Listen bereitstellt.

Der Ansatz von Williams u. Dabirsiaghi [2014], welcher sich mit bekannten Schwachstellen von Drittanbieter-Bibliotheken beschäftigt, könnte in zukünftigen Arbeiten genutzt werden, um das Werkzeug an diesen Schwachstellen zu ergänzen und so bessere Empfehlungen geben zu können, welche diese Sicherheitsmängel berücksichtigen.

A Anhang

A.1 IT-Markt

In Abbildung 14 ist zu erkennen, dass sich seit 2005 (544 Milliarden Euro) der Umsatz im Markt für Software und IT-Services weltweit bis 2015 bereits fast verdoppelt hat.¹⁸

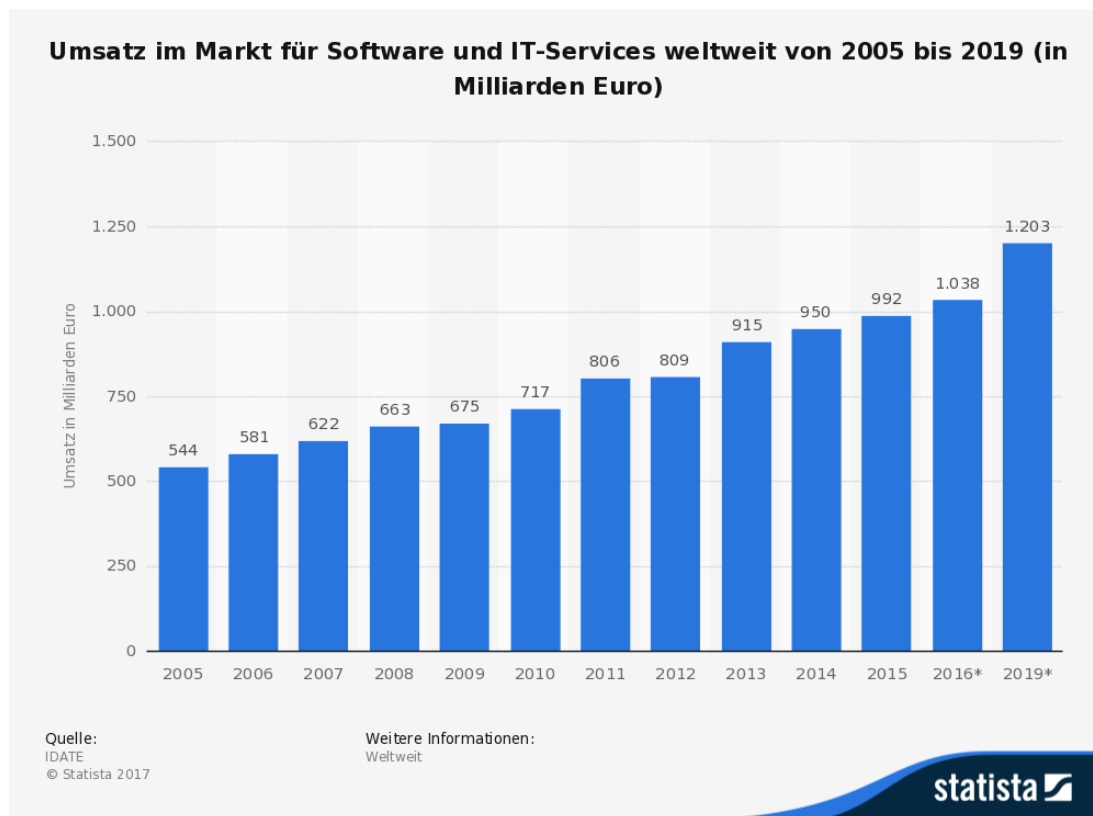


Abbildung 14: Umsatz im Markt für Software und IT-Services [Statista, 2017]

A.2 INIT Solution GmbH

Für diese Arbeit konnten Mitarbeiter, der Firma INIT Solution GmbH (INIT) befragt werden, aufgrund der Werkstudentenanstellung der Verfasserin.

INIT ist eine Softwareentwicklungs- und beratungsfirma mit derzeit 78 Mitarbeitern in Stuttgart, Bremerhaven, Valencia und Hamburg. Die Firma wurde 1996 von Peter Cepok und Thomas Winkelsdorf gegründet und arbeitet vorwiegend im Bereich SAP

¹⁸ Die mit * versehenen Jahreszahlen sind Prognosen.

mit namenhaften Kunden wie Daimler AG, Mercedes-Benz Bank AG und verschiedenen Netzbetreibern. Weitere Informationen zu INIT unter: <http://init-software.de/>

A.3 GitHub Projekt

In der Arbeit verwendete GitHub Projekte. Vorwiegend mit Maven erstellte Projekte, einige aber auch mit sbt, Grunt oder Bower. Für SAPUI5 sind auch Projekte enthalten, die ohne Build-Werkzeug erstellt wurden.

1. <https://github.com/qmacro/SAPUI5-Fiori>
2. <https://github.com/sschober/netty-example>
3. <https://github.com/Ekryd/sortpom>
4. <https://github.com/woodyalen202/playmvn>
5. <https://github.com/myfear/play-java-maven>
6. <https://github.com/ysokol/mavenproject>
7. <https://github.com/agrz/karma-openui5-maven-sample>
8. <https://github.com/eirslett/frontend-maven-plugin>
9. <https://github.com/ManuelB/blueprint>
10. <https://github.com/cgrail/SapUI5-for-Maven>
11. <https://github.com/6of5/UI5SplitApp-Boilerplate>
12. <https://github.com/release-engineering/pom-manipulation-ext>
13. <https://github.com/playframework/playframework>
14. <https://github.com/mitsuruog/sapui5-showroom>
15. <https://github.com/denisenepraunig/sapui5-hanaxs-examples/tree/master/todoapp>
16. <https://github.com/denisenepraunig/sapui5-hcp-examples>
17. <https://github.com/ui5experts/ui5-toolbox>
18. <https://github.com/janmattfeld/bachelor>
19. <https://github.com/caijiahao/springMvcPlusMongo>
20. <https://github.com/release-engineering/pom-manipulation-ext/blob/master/pom.xml>

A.4 Dokumentationen

Dokumentation, die für die Arbeit untersucht und verwendet wurden sind:

Foundation Framework unter: <http://foundation.zurb.com/sites/docs/> (abgerufen am 04.09.2017)

Twitter Bootstrap unter: <https://bootstrapdocs.com/v3.3.6/docs/> (abgerufen am 04.09.2017)

Play unter: <https://www.playframework.com/documentation/2.6.x/Home> (abgerufen am 04.09.2017)

Laravel unter: <https://laravel.com/docs/5.4/installation> (abgerufen am 04.09.2017)

Semantic UI unter: <https://semantic-ui.com/introduction/getting-started.html> (abgerufen am 04.09.2017)

Atlassian JIRA unter: <https://confluence.atlassian.com/jira/jira-documentation-1556.html> (abgerufen am 04.09.2017)

SAPUI5 unter: <https://sapui5.hana.ondemand.com/> (abgerufen am 04.09.2017)

OPENUI5 unter: <https://openui5.hana.ondemand.com/> (abgerufen am 04.09.2017)

AngularJs unter: <https://docs.angularjs.org/guide> (abgerufen am 04.09.2017)

Angular unter: <https://angular.io/docs> (abgerufen am 04.09.2017)

Spring Framework unter: <https://spring.io/docs> (abgerufen am 04.09.2017)

SAP HANA unter: <https://archive.sap.com/documents/docs/DOC-60319> (abgerufen am 04.09.2017)

A.5 Andere Build-Tools

Sbt

Sbt ist ein Build-Management Werkzeug für Scala Projekte. Da die Sprache Scala recht neu ist, können nur wenige Build-Werkzeuge benutzt bzw. nur eingeschränkt benutzt werden. Sbt erzeugt ein build.sbt in der die Informationen zu den Abhängigkeiten des Projektes gespeichert werden.

Bower

Das Paketverwaltungswerkzeug Bower hilft vorwiegend Frontend-Entwicklern Abhängigkeiten zu verwalten. In einer package.json sind alle Abhängigkeiten zu anderen Technologien zu finden.

npm is awesome as a package manager. In particular, it handles sub-dependencies very well: if my package depends on `request` version 2 and `some-other-library`, but `some-other-library` depends on `request` version 1, the resulting dependency graph looks like:

```
├─ request@2.12.0
└─ some-other-library@1.2.3
   └─ request@1.9.9
```

This is, generally, great: now `some-other-library` has its own copy of `request` v1 that it can use, while not interfering with my package's v2 copy. Everyone's code works!

Abbildung 15: „Peer Dependencies“ [Node.js, 2017]

Grunt

Grunt ist ein Task-Runner und soll eigentlich dabei helfen Aufgaben, welche häufig wiederholt werden, zu automatisieren. Aber auch Grunt verwendet das package.json für Abhängigkeiten.

A.6 Stack Exchange

Bei der Untersuchung von Stack Overflow wurde auch die Möglichkeit betrachtet über Stack Exchange mit Hilfe von Abfragen Stack Overflow Threads auszulesen. Dafür wurden für die Arbeit Abfragen entwickelt, welche Threads aufzeigen sollten in denen Abhängigkeitsbeziehungen abgebildet werden. Ein Beispiel für solch eine Abfrage ist die nachfolgende:

```
SELECT p.Id AS [Post Link],p.Body as Question, p.AnswerCount as Answers,  
p.Tags as Tag  
FROM Posts p  
WHERE p.Body LIKE '###SearchWord###'  
AND p.PostTypeId = 1  
AND p.AnswerCount >= 1  
AND (p.Body LIKE '###TagName###'  
OR p.Tags LIKE '###TagName###')
```

Quellcode-Beispiel 5: Abfrage Beispiel

In dem Quellcode-Beispiel 5 ist eine Abfrage dargestellt, welche alle Post mit ihrer Id, der Frage und der Antworten Anzahl selektiert. In der die Variable „SearchWord“ enthalten ist, der Post einer Frage entspricht, Antworten existieren und die Variable „TagName“ in der Frage selbst oder als Tag vorkommen. Die Variablen können selbst angegeben werden, dadurch können mit der Abfrage verschiedene Begriffe gesucht werden, im Zusammenhang mit bestimmten Technologien.

SearchWord

depend

TagName

angularjs

Run Query

Cancel

Options:

☐ Text-only results
 ☐ Include execution plan

Switch sites:

Results

Messages

Download CSV

Post Link	Question	Answers	Tag
Karma + Angular undefined error	<p>I just started angular testing with karma a...	1	<javascript><angularjs><unit-testing><karma...
angular gulp - how to split js file in dist folder ...	<p>I'm using <a href="https://github.com/Swii...	2	<javascript><angularjs><node.js><gulp>
How to build the yeoman "angular-generator" ...	<p>I created my angular application with "yeo...	4	<angularjs><gruntjs><yeoman>
Loading Angularstrap datepicker CSS from B...	<p>Angularstrap say that it's only dependenci...	1	<angularjs><twitter-bootstrap><angular-strap>
How can i inject a mock service into a unit tes...	<p>I have a simple angularjs filter (it takes an...	1	<javascript><unit-testing><angularjs><mocki...
Using Angular to pull data from MongoDB ba...	<p>I am new to Angular and Mongo, and am ...	1	<javascript><angularjs><forms><mongodb><...
(AngularJS) Only one less file for entire Webs...	<p>I am a beginner with AngularJS and I hav...	2	<css><angularjs><less>
How to dynamically add a decorator after ang...	<p>Is there a way in angularjs to dynamically ...	1	<angularjs>
angular UI bootstrap throws error	<p>I am very new to angular-Js. I am trying t...	1	<javascript><jquery><angularjs><twitter-boot...
how to prefill data with angucomplete plugin?	<p>I am using angular with angucomplete plu...	1	<angularjs>
Ionic : no content / white screen using interce...	<p>I successfully managed to use interceptor...	1	<angularjs><cordova><ionic-framework><ion...
CSRF Token Security	<p>i have disabled the laravel CSRF Token s...	1	<javascript><angularjs><laravel-5>
Comparing Two Arrays with ng-repeat	<p>I am trying create multiple checkboxes us...	1	<javascript><angularjs><ng-repeat>
Angular 1.4.0 \$timeout is not happening after ...	<p>I just upgraded to angular 1.4.0 and am h...	2	<angularjs>
AngularJS share session data between contr...	<p>I'm new to AngularJS and i don't know wh...	2	<javascript><angularjs><session>
AngularUI modal custom scope	<p>I wanted to define a custom scope for the...	2	<angularjs><angular-ui-bootstrap>

9699 rows returned in 247206 ms

Abbildung 16: Ergebnis der Abfrage

In Abbildung 16 sieht man die hohe Anzahl an Ergebnissen bei einer solchen Abfrage anhand der Anzahl der „Rows“ rechts unten im Bild. Diese manuell für eine qualitative Analyse auszuwerten, würde sehr lange dauern, weshalb die Analyse durch Stack Exchange an ihre Grenzen gelangte. Deshalb wäre dies ein interessanter Ansatz für zukünftige Arbeiten um automatisiert Wissen über Abhängigkeiten aus Stack Overflow zu generieren.

A.7 Metamodelle

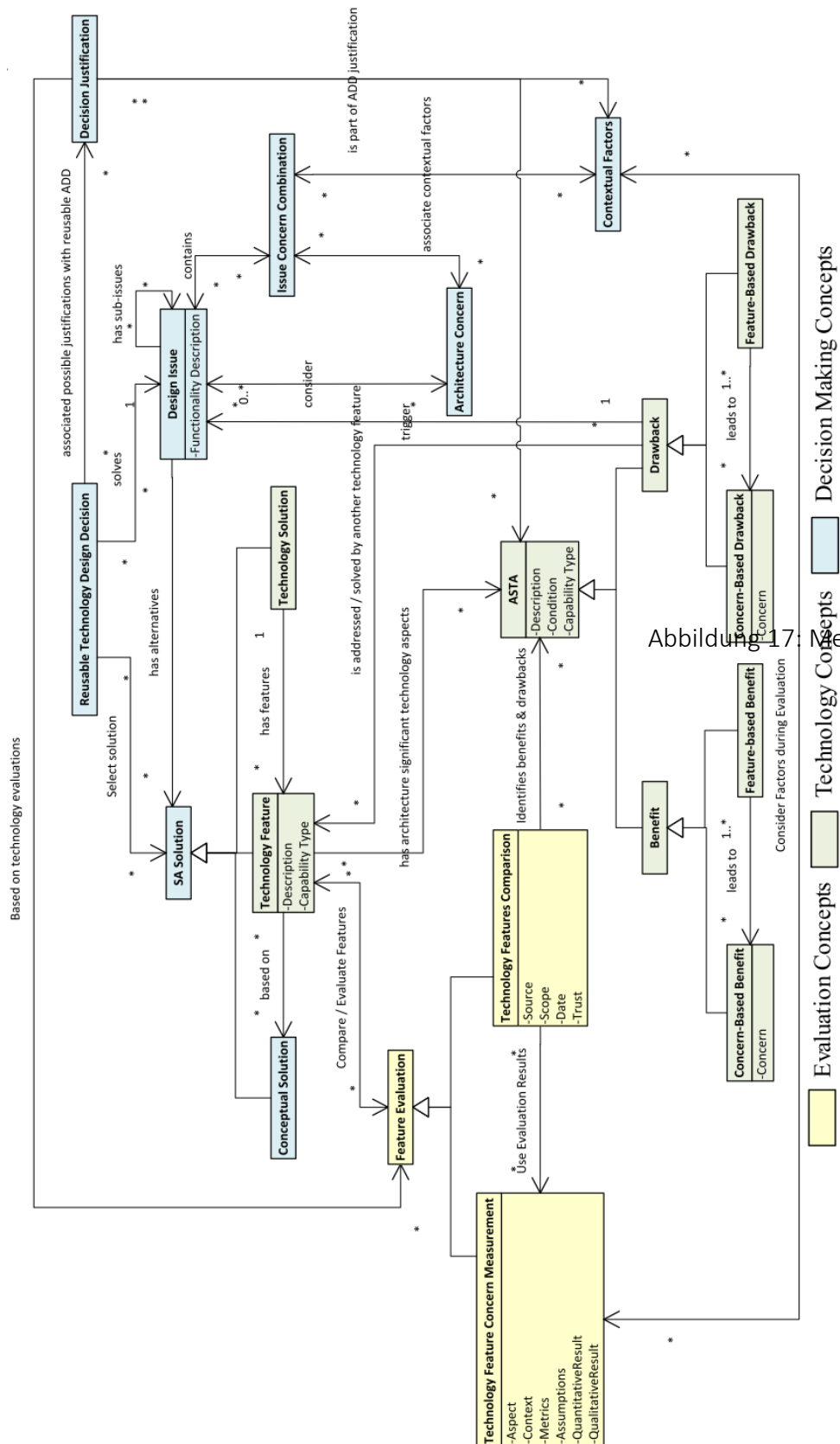


Abbildung 17: Metamodell [Soliman, 2015]

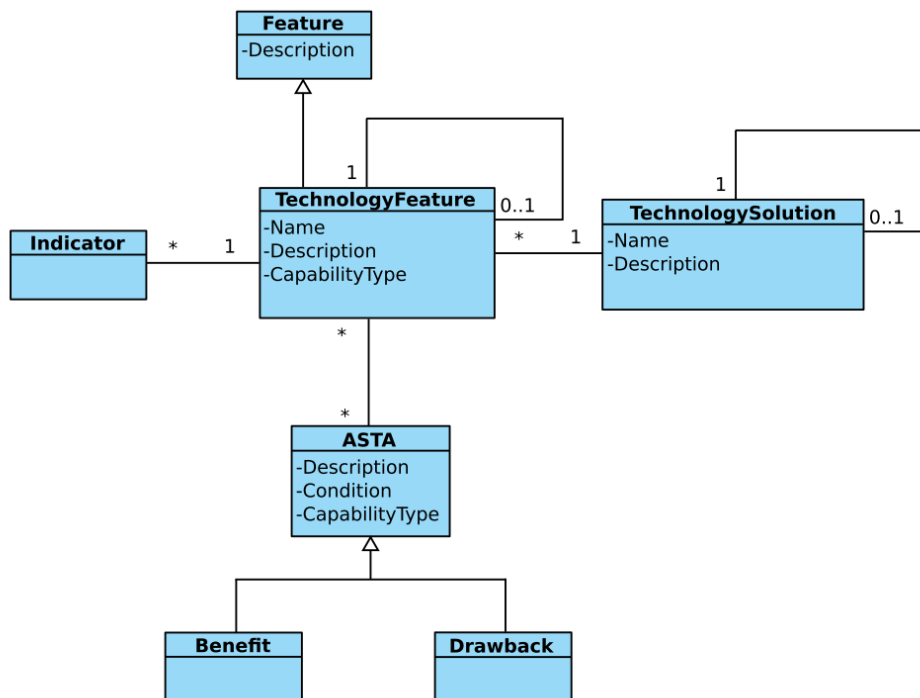


Abbildung 18: Erweitertes Metamodell [Fechner, 2016]

A.8 SAPUI5

Dokumentation

Aus Dokumentation von SAPUI5, unter <https://sapui5.hana.ondemand.com/> ('i' Menü oben links > About > Included Third-Party Software) (abgerufen am 05.09.2017).

- | | |
|---------------------------------|------------------------------|
| 1. Ace (Ajax.org Cloud9 Editor) | 21. JS-Signals |
| 2. Array.prototype Polyfills | 22. JSZip |
| 3. Blanket.js | 23. Junit Reporter for QUnit |
| 4. Canvg | 24. LESS |
| 5. Crossroads.js | 25. Mobify.js |
| 6. Cubi.org-iscroll | 26. MobiScroll |
| 7. Cubi.org – swipeview | 27. Punycode.js |
| 8. D3 | 28. Qunit Compsite |
| 9. DataJS | 29. RaphaelJs |
| 10. Es6-promise | 30. requireJS |
| 11. Esprima | 31. RGBColor Static |
| 12. Flexie.js | 32. SinonJS |

- | | |
|-----------------------------------|---|
| 13. Google-Caja JS HTML Sanitizer | 33. Top Down Operator Precedence |
| 14. Google-code-prettify | 34. Underscore |
| 15. Handlebars | 35. Unicode Common Locale Data Repository |
| 16. Hasher.js | 36. Unorm.js |
| 17. jQueryMobile | 37. URI.js |
| 18. jQuery resize event | 38. Xlsx |
| 19. jQuery UI Touch Punch | 39. Zynga Scroller |
| 20. jQuery, QUnit, jQueryUI | |

Von Stack Overflow:

- | | |
|------------------|----------------|
| • Datajs | • Webpack |
| • Angularjs | • Kendo UI |
| • PhantomJs | • jQuery 2.2.3 |
| • Qunit | • jQueryUI |
| • Babeljs Plugin | |

Plattform- und Browserunterstützung

Tabelle: Matrix zur Browser- und Plattformunterstützung

Platform	Device Category	Platform Version	Safari	Web View	Internet Explorer	Microsoft Edge	Google Chrome	Mozilla Firefox	SAP Fiori Client
Windows	Desktop	Windows 7, 8.1	-	-	11	-	Latest version	Latest and Extended Support Release (ESR)*	-
		Windows 10	-	Latest version		Latest Current Branch for Business			Latest version
	Touch	Windows 8.1	-	-	11	-	Latest version	Latest and Extended Support Release (ESR)*	Up to version 1.5.3

Platform	Device Category	Platform Version	Safari	Web View	Inter- net Ex- plorer	Microsoft Edge	Google Chrome	Mozilla Firefox	SAP Fi- ori Cli- ent
		Windows 10	-	Latest version		Latest Current Branch for Business			Latest version
Windows Phone	Phone	Windows 10 Mobile	-	Latest version	-	Latest version	-	-	Latest version
macOS X	Desktop	Latest 2 versions	Latest 2 versions	-	-	-	Latest version	-	-
iOS	Phone and Tablet	As of version 9	Latest 2 versions	Latest version	-	-	-	-	Latest version
Android	Phone and Tablet	As of version 4.4	-	-	-	-	Latest version	-	Latest version

Quelle: „Browser and Platform Support“, unter: <https://sapui5.hana.ondemand.com/#docs/guide/74b59efa0eef48988d3b716bd0ecc933.html> (abgerufen am 04.09.2017).

Zudem wird dargestellt wie lange bestimmte Geräte unterstützt werden:

Tabelle 7: Geräteunterstützung

Platform	Device	End of Support Date
iOS Apple always supports the 2 latest releases of the iOS operating system. SAPUI5 supports Apple mo-	Apple iPhone 5 SE	31 March 2019
	Apple iPhone 6	19 September 2017
	Apple iPhone 6s	25 September 2018
	Apple iPhone 7	7 September 2019
	Apple iPad Air 2	22 October 2017

Platform	Device	End of Support Date
bile devices 3 years from the vendor device release date, except defined otherwise.	Apple iPad Mini 3	24 October 2017
	Apple iPad Pro	9 September 2018
Android Android OS based devices are very fragmented in matters of operating system variants and hardware diversity. SAPUI5 supports the following Android reference devices until 3 years from vendor device release date.	Samsung S5	11 April 2017
	Samsung S6	10 April 2018
	Samsung S7	10 March 2019
	Samsung Galaxy Tab Pro 10.1	27 February 2017
Windows Phone SAPUI5 supports the following Windows Phone reference devices until 3 years from vendor device release date.	Nokia Lumia 930	4 July 2017
	Nokia Lumia 950	20 November 2018

Quelle: „Browser and Platform Support“ unter: <https://sapui5.hana.ondemand.com/#docs/guide/74b59efa0eef48988d3b716bd0ecc933.html> (abgerufen am 04.09.2017).

A.9 AngularJS

Aus der Dokumentation von AngularJs, unter: <https://docs.angularjs.org/guide/external-resources> (abgerufen am 06.09.2017):

- UI Bootstrap
- Ace
- UI.sortable
- LoDash
- Momentjs
- MathJax
- Angular Translate
- Cordova
- PhoneGap
- CRUD
- Hood.io
- Mean
- Rails
- Meteor
- PHP
- MySQL
- SQLite
- Laravel 4 PHP Web
- MCript
- ES6

-
- Ng-annotate
 - Babel plugin
 - Angularytics
 - Angulartics
 - Stacktrace.js
 - Seo.js
 - Prerender.io
 - Brombone
 - Seo4Ajax
 - Django
 - Firebase
 - Cloud Endpoints
 - Webpack
 - Typescript
 - Dart
 - CoffeScript
 - Facebook Javascript SDK
 - SVG
 - NVD3.js
 - Raphael.js
 - Dr.js
 - Socket.io
 - Omni Binder

A.10 Play

Play Module, unter: <https://www.playframework.com/documentation/2.6.x/Module-Directory> (abgerufen am 10.09.2017).

API Hosting

- swagger-play
- iheartradio/play-swagger
- zalando/play-swagger
- mohiva/swagger-codegen-play-scala

Authentication (Login & Registration) and Authorization (Restricted Access)

- Silhouette (Scala)
- Deadbolt 2 Plugin
- Play-pac4j (Java and Scala)
- Authentication and Authorization module (Scala)
- Play! Authenticate (Java)
- SecureSocial (Java and Scala)

Datastore

- Flyway plugin
- MongoDB Jongo Plugin (Java)
- MongoDB Morphia Plugin (Java)
- MongoDB ReactiveMongo Plugin (Scala)

- Play-Hippo
- Play-Slick
- Redis Plugin (Java and Scala)
- ScalikeJDBC Plugin (Scala)
- Redis Cache Plugin (Java and Scala)

Deployment

- WAR Module

Page Rendering

- Play Pagelets

Localization

- FolderMessages plugin
- JsMessages
- Messages Compiler Plugin (Scala)
- Play I18n HOCON

Performance

- Google's HTML Compressor (Java and Scala)
- Memcached Plugin

Task Schedulers

- Akka Quartz Scheduler
- play-akkjobs

Templates and View

- Google Closure Template Plugin
- HTML5 Tags module (Java and Scala)
- Scalate
- PDF module (Java)
- Play-Bootstrap (Java and Scala)
- Thymeleaf module (Scala)
- Handlebars templates (Java and Scala)

Utilities

- Emailer Plugin (Java and Scala)

-
- Geolocation (Java)
 - JSONP filter
 - Sitemap Generator (Java)
 - play-guard (Scala)

Cloud services

- Amazon SES module (Scala)
- Amazon S3 module (Scala)
- Pusher
- Push Notifications module (Java)

A.11 Architektur des Prototyps

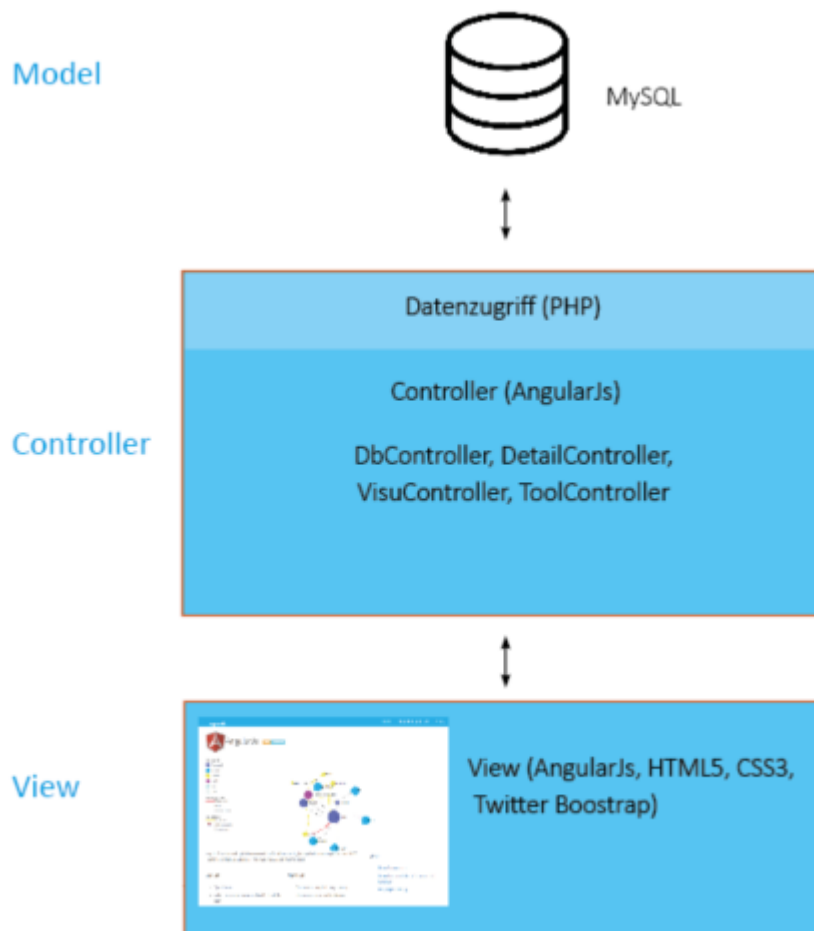


Abbildung 19: Architektur des Prototyps

A.12 Lokale Installation

Das Werkzeug kann online unter: <http://depends-thesis.de> verwendet werden.

Wenn eine lokale Installation auf dem Rechner erwünscht ist, kann das Projekt bei GitHub unter: <https://github.com/Jennyfa/masterprojekt> heruntergeladen werden, oder der Ordner ‚Masterarbeit‘ auf dem beigefügten Datenträger verwendet werden.

Zunächst muss auf dem lokalen System XAMPP installiert werden, dies kann unter: <https://www.apachefriends.org/de/index.html> heruntergeladen werden. Während der Installation muss ein Username und ein Passwort eingegeben werden, diese Daten werden später noch benötigt.

Um die Datenbank für den Prototyp zu verwenden, muss die Datenbank noch importiert werden. Dazu <localhost/phpmyadmin> aufrufen und mit den zuvor eingegeben Daten einloggen. Unter dem Tab ‚Importieren‘ kann die Datenbank für den Prototyp importiert werden. Dazu die Datei ‚db_master.sql‘ (von dem beigefügten Datenträger oder aus dem ZIP von GitHub) auswählen und auf OK klicken. Teilweise meldet phpmyadmin einen Fehler beim Import der db_master.sql. Bei nochmaliger Ausführung hat es bei der Verfasserin jedoch funktioniert.

Nun muss der Ordner ‚Masterarbeit‘ in den htdocs Ordner von xamp gelegt werden und die Datei ‚database_connections.php‘ in dem Ordner ‚databaseFiles‘ aufgerufen werden. Dort müssen die selbstgewählten Daten für User und Passwort der Datenbank eingefügt werden.

Nun kann die Seite localhost im Browser aufgerufen werden, dann sollte die Startseite wie in Abbildung 8 zu sehen sein.

A.13 Fragebogen zur Bewertung

Sie arbeiten an einem Projekt, welches schon vor längerer Zeit begonnen wurde. Dazu benötigen Sie eine Frontend-Technologie, welche mittels OData-Services mit Ihrer Datenbank und dem Backend kommunizieren kann, um den Aufwand möglichst gering zu halten. Benutzen Sie das Tool für eine Empfehlung.

Entwickler 1:

- Wie hat Ihnen das Tool geholfen?

Das Tool hat mir geholfen. Vor allem die Darstellungen der verschiedenen Alternativen Möglichkeiten sind gut.

- **Wie finden Sie die bereitgestellten Informationen?**

Die Visualisierung der Abhängigkeiten ist für mich etwa unübersichtlich. Die Sektion für Kombinationen, Alternativen, Empfehlungen und Ermöglichkeiten gefällt mir sehr gut. Auch die Vor- und Nachteile finde ich sehr gut

- **Würden Sie zusätzliche Aspekte interessieren?**

Abgrenzungen würde ich noch nützlich finden. Also eine Abfrage, welche Technologien ich auf keinen Fall nutzen möchte.

- **Was sind die wichtigsten Kriterien für Sie?**

Zur Auswahl einer Technologie ist für mich wichtig:

- die Community,
- ob es Open-Source ist
- ob Verpflichtungen wie Lizenzkosten dahinterstehen
- schnell erlernbar ist

- **Wie viel Zeit denken Sie, hätten Sie benötigt um dieselben Informationen selbst herauszufinden?**

Eine Menge Zeit, gewisse Technologien darin kenne ich gar nicht.

- **Wäre so ein Werkzeug für Ihre Arbeit hilfreich? Was müsste es können um hilfreich zu sein?**

Ja, wäre es. Es könnte noch hilfreich sein, auswählen zu können, welche Technologien auf jeden Fall enthalten sein müssen. Und auf welchen Geräten, das Produkt genutzt werden soll, und ob es auch offline auf einem Smartphone genutzt werden soll.

- **Wenn ja, wie häufig würden Sie das Tool benutzen? Tägliche Benutzung oder nur selten?**

Ich es immer bei neuen Produkten, oder wenn neue Technologien auszuwählen sind nutzen. Also derzeit wahrscheinlich so 3x im Jahr. Dabei würde es aber viel Zeit sparen.

- **Wie untersuchen Sie derzeit Abhängigkeiten?**

Derzeit google ich nach den Technologien, schreibe die Einzelheiten aus und diskutiere das Wissen mit Kollegen. Das wiederhole ich, bis eine Entscheidung gefallen ist.

- **Wie viel Zeit benötigen Sie dafür?**

Sehr lange. Sehr, sehr lange. Etwa 10-12 Stunden für eine Technologie.

- **Wie nützlich ist die Identifikation der Abhängigkeiten?**

Sehr nützlich, jedoch die Darstellung wie gesagt, noch etwas unübersichtlich.

- **Welche Auswahlkriterien für das Werkzeug würden sie noch interessieren?**
 - Abgrenzungen von Technologien
 - Hardware
 - Will man einen extra Webserver haben oder nicht
 - Nur Lizenzfreie Software auswählbar
- **War das Werkzeug leicht zu bedienen?**

Ja, es war leicht bedienbar.
- **Was könnte man besser machen?**

Die Übersichtlichkeit der Visualisierung.
- **Was hat Ihnen gut gefallen?**

Es ist einfach auswählbar. Die allgemeine Tabelle für die Technologien gefällt mir auch gut, so kann man sich auch einfach zu Technologien informieren.
Das ist auch sehr übersichtlich.
- **Was hat Ihnen nicht gut gefallen?**
 - Der Graph könnte noch übersichtlicher gestaltet werden
 - Das mit STRG die bisherige Auswahl entfernen kann und Mehrfachauswahl möglich ist.
 - Alphabetische Sortierung wäre gut
 - Suchfeld in Multiselects

Entwickler 2:

- **Wie hat Ihnen das Tool geholfen?**

Es hat mir geholfen einen Überblick zu erhalten.
- **Wie finden Sie die bereitgestellten Informationen?**

Finde ist super. Vorallem die Visualisierung finde ich sehr gut. Hammer. Farblich schön abgestimmt. Gelb ist allerdings vielleicht etwas zu hell.
- **Würden Sie zusätzliche Aspekte interessieren?**

Der Schwierigkeitsgrad wäre für mich noch wichtig.
- **Was sind die wichtigsten Kriterien für Sie?**

-
- **Wie viel Zeit denken Sie, hätten Sie benötigt um dieselben Informationen selbst herauszufinden?**

Bietet schon viele Infos. Also länger als jetzt. Wahrscheinlich mehrere Tage, wenn man die verschiedenen Technologien vergleichen will. Eventuelle hätte man eine Analyse gestartet.
- **Wäre so ein Werkzeug für Ihre Arbeit hilfreich? Was müsste es können um hilfreich zu sein?**

Ja, wäre es. Aufgrund der Menge an Technologien, die es im Webbereich gibt, würde es einen guten Überblick geben.

- **Wenn ja, wie häufig würden Sie das Tool benutzen? Tägliche Benutzung oder nur selten?**

Immer zu Anfang eines Projektes bzw. wenn Technologieentscheidungen innerhalb der Projekts nötig sind.

- **Wie untersuchen Sie derzeit Abhängigkeiten?**

Sehr schwierig, die meiste Zeit google ich.

- **Wie viel Zeit benötigen Sie dafür?**

Sehr viel. Erst einmal die Artikel zu den Technologien lesen. Viel Zeit, deswegen würde das Werkzeug viel eingrenzen.

- **Wie nützlich ist die Identifikation der Abhängigkeiten?**

Finde ich sehr gut.

- **Welche Auswahlkriterien für das Werkzeug würden sie noch interessieren?**

-

- **War das Werkzeug leicht zu bedienen?**

Ja, war es.

- **Was könnte man besser machen?**

Hilfetexte, wie was bedeuten die jeweiligen Features etc. und eine Anmerkung, das Mehrfachauswahl möglich ist.

- **Was hat Ihnen gut gefallen?**

Das Design hat mir sehr gut gefallen. Und für einen Prototypen wirklich gut.

- **Was hat Ihnen nicht gut gefallen?**

- Die gelbe Farbe in der Visualisierung könnte noch etwas dunkler.
- Zurück Button bei Tool Empfehlung, um die Auswahlkriterien zu verändern.

Entwickler 3:

- **Wie hat Ihnen das Tool geholfen?**

Grundsätzlich nettes Ding.

- **Wie finden Sie die bereitgestellten Informationen?**

Sehr beeindruckend, sehr modern.

- **Würden Sie zusätzliche Aspekte interessieren?**

Der Preis für die Technologien

- **Was sind die wichtigsten Kriterien für Sie?**

-

- **Wie viel Zeit denken Sie, hätten Sie benötigt um dieselben Informationen selbst herauszufinden?**

Jahre, also wirklich lange.

- **Wäre so ein Werkzeug für Ihre Arbeit hilfreich? Was müsste es können um hilfreich zu sein?**

Ja, wäre es. Setzt allerdings eine Breite an Wissen und einen Standard des Wissens voraus.

- **Wenn ja, wie häufig würden Sie das Tool benutzen? Tägliche Benutzung oder nur selten?**

Immer, wenn man vor Entscheidungen steht. Würde es auch an Kollegen empfehlen.

- **Wie untersuchen Sie derzeit Abhängigkeiten?**

Derzeit google ich.

- **Wie viel Zeit benötigen Sie dafür?**

Gefüllt Jahre.

- **Wie nützlich ist die Identifikation der Abhängigkeiten?**

Die genutzten Abhängigkeiten finde ich sehr nützlich.

- **Welche Auswahlkriterien für das Werkzeug würden sie noch interessieren?**

-

- **War das Werkzeug leicht zu bedienen?**

Ja, sehr intuitiv.

- **Was könnte man besser machen?**

Man könnte die untern Bereiche wie Vor- und Nachteile etc. noch anders darstellen, irgendwie moderner. So, dass das einfliegt oder so. Derzeit sind es für mich irgendwie zwei Welten. Erklärungen für die Abhängigkeiten als Pop-up würde ich auch gut finden. Und wie man die Mehrfachauswahl macht.

- **Was hat Ihnen gut gefallen?**

Die Visualisierung ist cool. Wie die rein kommt und auch die Bewegung.

Entwickler 4:

- **Wie hat Ihnen das Tool geholfen?**

Das ist auf jeden Fall eine gute Idee, die mir sehr helfen würde. Ist eine sinnvolle Sache, allerdings muss es dann immer aktuell gehalten werden.

- **Wie finden Sie die bereitgestellten Informationen?**

Die Darstellung ist total fancy, sehr gut. I like.

- **Würden Sie zusätzliche Aspekte interessieren?**

Eine Blogging Funktion würde ich gut finden. Bei der Entwickler und Architekten, noch etwas zu den Technologien hinzufügen können.

- **Was sind die wichtigsten Kriterien für Sie?**

Eine dynamische Gewichtung der einzelnen Kriterien würde ich als sinnvoll erachten, also dass der Nutzer diese selber vorgeben kann.

- **Wie viel Zeit denken Sie, hätten Sie benötigt um dieselben Informationen selbst herauszufinden?**

-

- **Wäre so ein Werkzeug für Ihre Arbeit hilfreich? Was müsste es können um hilfreich zu sein?**

Ich würde das Werkzeug weniger nutzen aber ich kann mir sehr gut vorstellen, dass es für Kollegen die Technologien auswählen müssen sehr sinnvoll ist.

- **Wenn ja, wie häufig würden Sie das Tool benutzen? Tägliche Benutzung oder nur selten?**

Ein IT-Leister vielleicht 2-3 x im Jahr. Als Berater könnte man das Tool auch zur Validierung benutzen. Und Webentwickler wohl auch bei jedem neuen Projekt.

- **Wie untersuchen Sie derzeit Abhängigkeiten?**

Durch das Knowhow von Kollegen, (Experten Meinungen). Und wenn niemand was dazu weiß google ich. Sehr aufwendig diese Informationen suchen zu müssen.

- **Wie viel Zeit benötigen Sie dafür?**

Wohl Tage.

- **Wie nützlich ist die Identifikation der Abhängigkeiten?**

Finde ich sehr umfangreich und auch relevant.

- **Welche Auswahlkriterien für das Werkzeug würden sie noch interessieren?**

Die Erlernbarkeit der Technologie (schwer, leicht etc.)

- **War das Werkzeug leicht zu bedienen?**

Ja, ist es.

- **Was könnte man besser machen?**

Wäre toll, wenn man bei den Empfehlungen zurückspringen könnte und das Werkzeug der Community bereitgestellt werden würde. Dadurch könnte es feingranularer werden. Zudem sollte eine Anmerkung gemacht werden wie man mehrere Technologien auswählen kann. Und eine Hilfetext, warum sich das Auswahlverhalten bei Neuimplementierung und Erweiterung ändert.

- **Was hat Ihnen gut gefallen?**

- Informationen sind sehr übersichtlich
- Einfache Bedienung

- Design gefällt mir gut
- Visualisierung auch

Entwickler 5:

- **Wie hat Ihnen das Tool geholfen?**

Ja es ist sehr hilfreich, Hinweise und Empfehlungen zu bekommen, welche Technologien einsatzbar wären und welche eingesetzt werden müssten. Auch die Zusammenhänge finde ich interessant.

- **Wie finden Sie die bereitgestellten Informationen?**

Finde ich gut.

- **Würden Sie zusätzliche Aspekte interessieren?**

Wie funktioniert die Kombination, dass das einmal beschrieben wird. Und noch detaillierte generelle Beschreibung würde ich gut finden.

- **Was sind die wichtigsten Kriterien für Sie?**

-

- **Wie viel Zeit denken Sie, hätten Sie benötigt um dieselben Informationen selbst herauszufinden?**

Für eine Technologie denke ich so 1 bis 1 ½ Stunden.

- **Wäre so ein Werkzeug für Ihre Arbeit hilfreich? Was müsste es können um hilfreich zu sein?**

Rezensionen aus der Community würde ich noch hilfreich für die jeweiligen Technologien finden.

- **Wenn ja, wie häufig würden Sie das Tool benutzen? Tägliche Benutzung oder nur selten?**

4-5 Mal im Jahre denke ich, weil man sich nicht so häufig neue Technologien raussucht. Meist arbeitet man mit denselben. Würde mir dann aber viel Arbeit abnehmen, wenn ich es müsste.

- **Wie untersuchen Sie derzeit Abhängigkeiten?**

Mit den offiziellen Seiten der Technologie, Forumsbeiträge, Arbeitskollegen und Freunde fragen.

- **Wie viel Zeit benötigen Sie dafür?**

Für eine Technologie wohl auch so 1 bis 1 ½ Stunden denke ich.

- **Wie nützlich ist die Identifikation der Abhängigkeiten?**

Sehr nützlich.

- **Welche Auswahlkriterien für das Werkzeug würden sie noch interessieren?**

Das man Technologien ausschließen kann.

- **War das Werkzeug leicht zu bedienen?**

Ja, ist es.

- **Was könnte man besser machen?**

Eine Erklärung, wie die Mehrfachauswahl möglich ist, für mich jette kein Problem, aber vielleicht für andere. Auswahlfenster der Kriterien größer. Der Name in der Tabelle schon als Link zu den Details (intuitiv will man daraufklicken). Bei der Visualisierung einen Button aus der „Details ansehen“ machen. Damit man weiß, dass man daraufklicken kann.

- **Was hat Ihnen gut gefallen?**

Die Grafik generell finde ich sehr gut. Die Visualisierung und die Legende dazu. Auch das einem schon Vor und Nachteile angezeigt werden.

- **Was hat Ihnen nicht gut gefallen?**

Die Visualisierung könnte etwas übersichtlicher gestaltet werden.

Entwickler 6:

- **Wie hat Ihnen das Tool geholfen?**
- **Wie finden Sie die bereitgestellten Informationen?**

Finde ich gut. Allerdings konnte ich mir auch noch Beispiel Quellcode vorstellen, etwa wie eine Datenbank-Anbindung in der jeweiligen Technologie funktioniert. Wenn man viele Sprachen beherrscht ist das als Auffrischung sehr nützlich.

- **Würden Sie zusätzliche Aspekte interessieren?**

Webservices würden mich noch interessieren.

- **Was sind die wichtigsten Kriterien für Sie?**

Die Features sind für mich sehr wichtig und welche Technologien bereits eingesetzt wurden.

- **Wie viel Zeit denken Sie, hätten Sie benötigt um dieselben Informationen selbst herauszufinden?**

1-3 Tage für eine Technologie, je nach Umfang des Projektes und das benötigte Wissen dazu.

- **Wäre so ein Werkzeug für Ihre Arbeit hilfreich? Was müsste es können um hilfreich zu sein?**

Wenn man dieses Werkzeug für sehr viele Technologien entwickeln würde, wäre es in jedem Fall sehr hilfreich.

- **Wenn ja, wie häufig würden Sie das Tool benutzen? Tägliche Benutzung oder nur selten?**

Für jedes Projekt. Bei Updates und Erweiterungen-

- **Wie untersuchen Sie derzeit Abhängigkeiten?**
Durch das Lesen über die Technologien, Googlen. Wobei Handbücher selbst meist zu lang sind.
- **Wie viel Zeit benötigen Sie dafür?**
1-5 Tage bei unbekannten Technologien, 1-2 Stunden bei bekannten.
- **Wie nützlich ist die Identifikation der Abhängigkeiten?**
Ich finde sie nützlich, jedoch kommt es auch immer auf das Projekt an.
- **Welche Auswahlkriterien für das Werkzeug würden sie noch interessieren?**
-
- **War das Werkzeug leicht zu bedienen?**
Es ist sehr leicht und einfach zu bedienen. Sehr übersichtlich
- **Was könnte man besser machen?**
Das nach oben scrollen. Einen Zurück Button bei den Empfehlungen.
- **Was hat Ihnen gut gefallen?**
Das es einfach zu bedienen ist, die Abhängigkeiten und deren Visualisierung.

Literaturverzeichnis

- [Adlon, 2015] J. Adlon: „SAP HANA – Architektur und Betrieb“ unter: <https://assets.cdn.sap.com/sapcom/docs/2015/11/22cacd45-4c7c-0010-82c7-eda71af511fa.pdf>, 2015, (abgerufen am 27.08.2017).
- [Advani, 2013] Advani, R.: „How to solve cyclic dependency between different modules in a project in eclipse?“, unter: <https://stackoverflow.com/questions/17208339/how-to-solve-cyclic-dependency-between-different-modules-in-a-project-in-eclipse> (abgerufen am 27.08.2017).
- [Akudev, 2015] Akudev: „SAPUI5 Basics - How SAPUI5 works“, unter: <https://stackoverflow.com/questions/27637685/sapui5-basics-how-sapui5-works> (abgerufen am 27.08.2017).
- [Almis, 2015] Almis.: „Collapsing nav doesnt work nor dropdowns!? bootstrap“, unter: <https://stackoverflow.com/questions/28591114/collapsing-nav-doesnt-work-nor-dropdowns-bootstrap/28591179#28591179> (abgerufen am 27.08.2017).
- [AngularJs, 2017] AngularJs.: „Alternatives“, unter: <https://angularjs.org/> (abgerufen am 29.08.2017).
- [AngularJsDocs, 2017] AngularJs Docs.: „Alternatives“, unter: <https://docs.angularjs.org/guide/bootstrap> (abgerufen am 29.08.2017).
- [Antolovic, 2015]] M. Antolovic: „Einführung in SAPUI5“, 2. Auflage, Rheinwerk Verlag., Bonn, 2015.
- [Atlassian, 2017] Atlassian : „Supported platforms“, unter: <https://confluence.atlassian.com/adminjiraserver073/supported-platforms-861253018.html#Supportedplatforms-postgresnote> (abgerufen am 27.08.2017).
- [Bachmann, 2009] K. U. Bachmann: „Maven 2. Eine Einführung, aktuell zur Version 2.0.9“, Addison-Wesley, München, 2009.
- [Balasubramanee et al., 2013] V. Balasubramanee, C. Wimalasena, R. Singh und M. Pierce, „Twitter Bootstrap and AngularJS Frontend Frameworks to expedite Science Gateway development“ in IEEE International Conference on Cluster Computing (CLUSTER), S. 4799, 2013.
- [Bass et al., 2012] L. Bass, P. Clements, and R. Kazman: „Software Architecture in Practice“, 3. Auflage. Addison Wesley, Bosten, 2012.
- [Bootstrap, 2017] Bootstrap : „Getting Started“, unter: <https://getbootstrap.com/docs/3.3/getting-started/> (abgerufen am 27.08.2017).

- [Broy et al., 2006] M. Broy et al.: „*Manifest: Strategische Bedeutung des Software Engineering in Deutschland*,“ Informatik-Spektrum, vol. 29, no. 3, S. 210–221, 2006.
- [C.N., 2015] N., C.: „Does Apache Olingo support Oracle database“, unter: <https://stackoverflow.com/questions/30544713/does-apache-olingo-support-oracle-database> (abgerufen am 27.08.2017).
- [Caijiahao, 2016] Caijiahao : „springMvcPlusMongo“, unter: <https://github.com/caijiahao/springMvcPlusMongo> (abgerufen am 27.08.2017).
- [Chen u. Xing, 2016] C. Chen and Z. Xing, „Mining Technology Landscape from Stack Overflow,“ 2016.
- [Chen u. Xing 2, 2016] C. Chen und Z. Xing, „Towards Correlating Search on Google and Asking on Stack Overflow“, in 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), S. 83–92, 2016.
- [CubeServ , 2015]. CubeServ: „SAPUI5 revolutioniert die Anwendungsentwicklung“, unter: <https://www.referenzportal.ch/allgemein/sapui5-revolutioniert-die-anwendungsentwicklung/> (abgerufen am 27.08.2017).
- [Dascalescu, 2008] Dascalescu , D.: „What are alternatives to ExtJS?“, unter: <https://stackoverflow.com/questions/200284/what-are-alternatives-to-extjs?answertab=oldest#tab-top> (abgerufen am 27.08.2017).
- [Dira, 2013] K. Dira, Software Architecture, Bd. 7957, Nr. July. 2013.
- [Engelbrecht u. Wegelin, 2016] M. Engelbrecht, M. Wegelin: „SAP Fiori: Implementierung und Entwicklung“, Rheinwerk Verlag, Bonn, 2016.
- [Fäber, 2004] F. Fäber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, J. Dees: „*The SAP HANA Database – An Architecture Overview*“, in „Data Engineering“, Society, Bd. 26, Nr. 1, 2004.
- [Fechner, 2016] T. Fechner: „Identifikation von Technologie-Features in existierenden Softwaresystemen“, 2016.
- [Foundation, 2017] Foundation: „Welcome to Foundation 6“, unter: <http://foundation.zurb.com/sites/docs/> (abgerufen am 27.08.2017).
- [Gerdes et al., 2014] S. Gerdes, S. Lehnert, und M. Riebisch, „Combining architectural design decisions and legacy system evolution“, Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Bd. 8627 LNCS, S. 50–57, 2014.
- [Gołębiowski, 2014] Gołębiowski: „Check Dependencies“, unter: <https://github.com/mgol/check-dependencies> (abgerufen am 27.08.2017).

-
- [Gorton et al., 2015] I. Gorton, J. Klein, und A. Nurgaliev, „Architecture Knowledge for Evaluating Scalable Databases“, Proc. - 12th Work. IEEE/IFIP Conf. Softw. Archit. WICSA 2015, S. 95–104, 2015.
- [Grail, 2014] C. Grail: „SapUI5 for Maven“, unter: <https://github.com/cgrail/SapUI5-for-Maven> (abgerufen am 04.09.2017).
- [Green u. Seshadri, 2013] B. Green u. S. Seshadri: „AngularJS“, O'Reilly Media, Sebastopol, 2013.
- [Große Bley et al., 2012] O. Große Bley, M. Nee, L. Jungclaus: „Beschreibung und Einsatz von In-Memory-Computing“, unter: http://winfwiki.wi-fom.de/index.php/Beschreibung_und_Einsatz_von_In-Memory-Computing (abgerufen am 27.08.2017).
- [H., 2015] Eric H.: „Is there any AngularJS + ASP.NET-WebApi + OData + Breeze.js + Typescript examples or did someone try to combine those“, unter: <https://stackoverflow.com/questions/29058810/is-there-any-angularjs-asp-net-webapi-odata-breeze-js-typescript-example> (abgerufen am 27.08.2017).
- [Hery-Moßmann, 2016] N. Hery-Moßmann: „Backend und Frontend - was ist das? Einfach erklärt“, unter: http://praxistipps.chip.de/backend-und-frontend-was-ist-das-einfach-erklart_41384 (abgerufen am 27.08.2017).
- [Hurtado, 2017]. A. Hurtado: „Play-Bootstrap“, unter: <https://adrianhurt.github.io/play-bootstrap/> (abgerufen am 06.09.2017).
- [IEEE, 2007] ISO/IEC Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems, in *ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15*, vol., no., pp. c1-24, July 15 2007.
- [Jackson, 2015] J. Jackson: „GitHub launches desktop client to lure more developers“, unter: <https://www.playframework.com/documentation/2.5.x/Installing> (abgerufen am 27.08.2017).
- [Jain et al., 2014] N. Jain, P. Mangal und D. Mehta, „AngularJS : A Modern MVC Framework in JavaScript“, in *Journal of Global Research in Computer Science*, Bd. 5, Nr. 12, 2014.
- [Jakob, 2016] Jakob: „Bootstrap 3.3.6 and JQuery 3.1.0 not compatible?“, unter: <https://stackoverflow.com/questions/38434141/bootstrap-3-3-6-and-jquery-3-1-0-not-compatible> (abgerufen am 27.08.2017).
- [Jansen u. Bosch, 2004] A. Jansen, J. Bosch, „Evaluation of tool support for architectural evolution,“ S. 123–142, 2004.
- [Jansen u. Bosch, 2005] A. Jansen and J. Bosch: „Software Architecture as a Set of Architectural Design Decisions“, *5th Work. IEEE/IFIP Conf. Softw. Archit.*, pp. 109–120, 2005.
-

- [Jeong, 2015] H. Jeong: „SpringOfToby“, unter: <https://github.com/heestory/SpringOfToby> (abgerufen am 04.09.2017).
- [Klein u. Gorton, 2015] [Klein u. Gorton, 2015] J. Klein und I. Gorton, „Design Assistant for NoSQL Technology Selection“, Proc. 1st Int. Work. Futur. Softw. Archit. Des. Assist. - FoSADA '15, S. 7–12, 2015.
- [Kruchten, 2004] P. Kruchten, „An Ontology of Architectural Design Decisions in Software-Intensive Systems,“ 2nd Groningen Work. Softw. Var., pp. 1–8, 2004.
- [Kruchten et al., 2004] P. Kruchten, P. Lago, and H. Van Vliet, „Building up and reasoning about architectural knowledge,“ *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4214 LNCS, pp. 43–58, 2006.
- [Laravel, 2017] Laravel: „Installation“, unter: <https://laravel.com/docs/5.0> (abgerufen am 27.08.2017).
- [Long, 2017] J. Long: „Dependency Check“, unter: <https://github.com/jeremylong/DependencyCheck> (abgerufen am 27.08.2017).
- [Maven, 2017] Maven: „Pom Reference“, unter: <https://maven.apache.org/pom.html> (abgerufen am 27.08.2017).
- [Makoto, 2014] Makoto: „Resolving or Compiling Circular Dependency in Maven“, unter: <https://stackoverflow.com/questions/27071994/resolving-or-compiling-circular-dependency-in-maven> (abgerufen am 27.08.2017).
- [Maxogden, 2017] Maxogden: „Dependency Check“, unter: <https://github.com/maxogden/dependency-check> (abgerufen am 27.08.2017).
- [Miesbauer u. Weinreich, 2013] C. Miesbauer and R. Weinreich, „Classification of design decisions: An expert survey in practice,“ in Proceedings of the 7th European Conference on Software Architecture, ser. ECSA'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 130–145.
- [Mszalbach, 2013] Mszalbach.: „how to resolve maven cyclic dependency“, unter: <http://stackoverflow.com/questions/16468525/how-to-resolve-maven-cyclic-dependency> (abgerufen am 27.08.2017).
- [Niehues, 2015] Niehues, T.: „How to build, deploy and test a SAPUI5/Fiori application on HANA XS in less than 10 minutes“, unter: <https://stackoverflow.com/questions/31829906/how-to-build-deploy-and-test-a-sapui5-fiori-application-on-hana-xs-in-less-than/31829907#31829907> (abgerufen am 27.08.2017).

-
- [Neumann u. Schicker, 2014] O. Neumann, E. Schicker: „*SAP HANA als In-Memory-Datenbank-Technologie für ein Enterprise Data Warehouse*“, Angew. Forsch. der Wirtschaftsinformatik Prozesse, Technol. Anwendungen, Syst. und Manag., S. 197–210, 2014.
- [Node.js, 2017] Node.js: „Peer Dependencies“, unter: <https://nodejs.org/en/blog/npm/peer-dependencies/> (abgerufen am 04.09.2017).
- [openSAP, 2017] openSAP: „Developing Java-Based Apps on SAP HANA Cloud Platform“ unter: <https://open.sap.com/courses/hcp2?locale=de> (abgerufen am 04.09.2017).
- [OpenUI5, 2017] OpenUI5: „Key Features“, unter: <http://openui5.org/features.html> (abgerufen am 27.08.2017).
- [Play, 2017] Play: „Installing Play“, unter: <https://www.playframework.com/documentation/2.5.x/Installing> (abgerufen am 27.08.2017).
- [Play_modules, 2017] Play: „Play modules“, unter: <https://www.playframework.com/documentation/2.5.x/ModuleDirectory> (abgerufen am 27.08.2017).
- [Popp, 2013] G. Popp: „Konfigurationsmanagement mit Subversion, Maven und redmine: Grundlagen für Softwarearchitekten und Entwickler“, dpunkt.verlag, Heidelberg, 2013.
- [Rauhe, 2011] H. Rauhe, „HANA In-Memory Computing Engine Concepts and Architecture Overview Hannes Rauhe Project & Team Research“, Techniques, 2011.
- [Roca, 2015] Roca, A.: „Angular with Play (Scala): ng-include references“ unter: <https://stackoverflow.com/questions/31699552/angular-with-play-scala-ng-include-references> (abgerufen am 27.08.2017).
- [Sapstudent, 2015] Sapstudent: „SAP HANA – Architecture (Part 1)“ unter: <http://www.sapstudent.com/hana/sap-hana-architecture-part-1> (abgerufen am 27.08.2017).
- [SAPUI5, 2017] SAPUI5: „Supported Library Combinations“ unter: <https://sapui5.hana.ondemand.com/#docs/guide/363cd16eba1f45babe3f661f321a7820.html> (abgerufen am 27.08.2017).
- [Schmitz, 2015] A. Schmitz: „Was ist eigentlich SAP HANA?“ vom 07.09.2015, unter: <http://news.sap.com/germany/ist-eigentlich-sap-hana/> (abgerufen am 27.12.2016).
- [Semantic UI, 2017] Semantic UI: „Neu in 2.2“ unter: <https://semantic-ui.com/introduction/new.html> (abgerufen am 27.08.2017).
-

- [Shaw u. Garlan, 1996] M. Shaw u. D. Garlan: Software Architecture. Perspectives on an emerging discipline“, Prentice Hall, Upper Saddle River, 1996.
- [Simon, 2002] H. Simon: „What We Know About Learning“ unter: <http://158.132.155.107/posh97/private/learning/learning-simon.htm> (abgerufen am 27.08.2017).
- [Soliman u. Riebisch, 2014] M. Soliman u. M. Riebisch, „Modeling the Interactions between Decisions within Software Architecture Knowledge“, Softw. Archit., S. 33–40, 2014.
- [Soliman et al., 2015] M. Soliman, M. Riebisch, und U. Zdun, „Enriching Architecture Knowledge with Technology Design Decisions“, Proc. - 12th Work. IEEE/IFIP Conf. Softw. Archit. WICSA 2015, S. 135–144, 2015.
- [Soliman et al., 2016] M. Soliman, M. Galster, A. R. Salama, and M. Riebisch: „Architectural knowledge for technology decisions in developer communities: An exploratory study with StackOverflow,“ Proc. - 2016 13th Work. IEEE/IFIP Conf. Softw. Archit. WICSA 2016, pp. 128–133, 2016.
- [Stack Overflow, 2016] Stack Overflow: „Developer Survey Results “ unter: <http://stackoverflow.com/insights/survey/2016#work> (abgerufen am 27.08.2017).
- [Statista, 2017] IDATE: : „Umsatz im Markt für Software und IT-Services weltweit von 2005 bis 2019 (in Milliarden Euro) von Juni 2016“. In Statista - Das Statistik-Portal unter: <https://de.statista.com/statistik/daten/studie/159325/umfrage/weltweiter-umsatz-mit-software-und-it-services-seit-2005/> (abgerufen am 03.03.2017).
- [Bahdanovich, 2013] A. Bahdanovich: „Add Dependencies to Maven pom.xml File“ unter: <http://www.tech-recipes.com/rx/39256/add-dependencies-to-maven-pom-xml-file/> (abgerufen am 27.08.2017).
- [Uthaman, 2016] S.Uthaman : „SAP HANA In-Memory Computing Engine (IMCE)– Index Server Architecture“ unter: <http://teachmehana.com/in-memory-computing-engine-architecture-sap-hana/> (abgerufen am 27.08.2017).
- [Tutorialspoint, 2017] Tutorialspoint: „Maven - Manage Dependencies“ unter: https://www.tutorialspoint.com/maven/maven_manage_dependencies.htm (abgerufen am 27.08.2017).
- [Tutorialspoint_hana, 2017] Tutorialspoint: „SAP HANA Tutorial“ unter: https://www.tutorialspoint.com/sap_hana/ (abgerufen am 27.08.2017).

-
- [Vogel et al., 2009] O. Vogel, I. Arnold, A. Chughtai, E. Ihler, T. Kehrer, U. Mehlig, U. Zdun: „*Software-Architektur: Grundlagen – Konzepte – Praxis*“, 2. Auflage, Spektrum Akademischer Verlag, Heidelberg, 2009.
- [Ward u. Leroux, 2014] J. Ward und N. Leroux: “Play for Java. Shelter Island”, Manning, Shelter Island, 2014.
- [Williams u. Dabirsiaghi, 2014] J. Williams und A. Dabirsiaghi: “The Unfortunate Reality of Insecure Libraries”, unter: <https://www.contrastsecurity.com/the-unfortunate-reality-of-insecure-libraries,2014>, abgerufen am 04.09.2017.
- [ysokol, 2017] ysokol : „Mavenproject“ unter: <https://github.com/ysokol/mavenproject> (abgerufen am 27.08.2017).
- [Zimmermann et al., 2007] O. Zimmermann, T. Gschwind, und J. Küster, „Reusable Architectural Decision Models for Enterprise“, S. 15–32, 2007.
- [Zimmermann et al., 2009] O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster, “Managing architectural decision models with dependency relations, integrity constraints, and production rules,” *J. Syst. Softw.*, vol. 82, no. 8, pp. 1249–1267, 2009.

Abbildungsverzeichnis

Abbildung 1: SAP-HANA-Datenbank-Architektur [Uthaman, 2016].....	12
Abbildung 2: Suche nach „depends“ auf Stack Overflow	32
Abbildung 3: Beispiel einer Abfrage über Stack Exchange	33
Abbildung 4: Grundlegendes Metamodell	44
Abbildung 5: Erweitertes Metamodell	45
Abbildung 6: Verwendung von Data.js in SAPUI5-Projekten	49
Abbildung 7: SAPUI5-Bibliotheken [SAPUI5, 2017]	50
Abbildung 8: Startseite	57
Abbildung 9: Werkzeug zur Entscheidungsunterstützung	58
Abbildung 10: Ergebnis der Abfrage	59
Abbildung 11: Auszug aus Datenbank-Tabelle Beziehung.....	59
Abbildung 12: Visualisierung der Abhängigkeiten	60
Abbildung 13: Alle Technologien, die in der Datenbank vorhanden sind	61
Abbildung 14: Umsatz im Markt für Software und IT-Services [Statista, 2017]	70
Abbildung 15: „Peer Dependencies“ [Node.js, 2017]	72
Abbildung 16: Ergebnis der Abfrage	74
Abbildung 17: Metamodell [Soliman, 2015]	75
Abbildung 18: Erweitertes Metamodell [Fechner, 2016]	76
Abbildung 19: Architektur des Prototyps	82

Tabellenverzeichnis

Tabelle 1: Abhängigkeiten von Kruchten und Zimmermann zusammengefasst.....	22
Tabelle 2: Abhängigkeiten in Build-Dateien	24
Tabelle 3: Abhängigkeiten Dokumentationen.....	29
Tabelle 4: Abhängigkeiten Stack Overflow	33
Tabelle 5: Gesamtüberblick über vorkommende Abhängigkeiten	35
Tabelle 6: Relevanz der Abhängigkeiten.....	36
Tabelle 7: Geräteunterstützung.....	78

Glossar

AngularJs AngularJs ist ein JavaScript basiertes Framework, welches vorwiegend für Single Page Applikationen verwendet wird und von Google Inc. entwickelt wurde.

Alternative Mit Alternative ist in dieser Arbeit eine weitere Technologie B gemeint, welche statt eine Technologie A verwendet werden kann, da beide die benötigten Features aufweisen.

Austauschbare Abhängigkeit Eine Technologie A benötigt entweder Technologie B oder C. Eine von beiden wird benötigt damit Technologie A funktioniert.

Backend Das Backend ist der Teil eines Systems, welches näher an der Verarbeitung ist, im Gegensatz dazu steht das Frontend.

Big Data Eine große Menge an Daten

Bower Ist ein kostenloses Paketverwaltungstool, welches den Umgang mit Abhängigkeiten vereinfacht, da Bibliotheken automatisch heruntergeladen und aktualisiert werden.

Constraint Eine Einschränkung der möglichen Lösungen, häufig durch verwendete Technologien, bereits getroffenen Entwurfsentscheidungen oder auch Anforderungen.

CSS3 Eine Stylesheet-Sprache, die viel im Bereich der Webentwicklung verwendet wird, um HTML5 Seiten optisch zu verändern.

Direktionale Abhängigkeit Eine Technologie A benötigt um zu funktionieren eine Technologie B. Dann ist A direktional abhängig von B.

Entwurfsentscheidung auch, Architekturentscheidung oder Designentscheidung. Eine Entscheidung die während des Architekturentwurfs getätigt wird.

Feature Beschreibt eine Funktionalität einer Software

Feature Direktionale Abhängigkeit Eine Technologie A besitzt ein Feature, welche bei Verwendung die Technologie B benötigt.

Framework Eine Software, welche durch Anpassungen an die eigenen Bedürfnisse angepasst werden kann

Frontend Ist der Teil des Systems, welcher näher am Benutzer ist. Es beschreibt den Teil mit dem der Benutzer interagiert. Im Gegensatz dazu steht das Backend.

GitHub Eine Internetplattform, auf der Nutzer Projekte hochladen können zur Versionierung und um sie mit anderen zu Teilen.

HTML5 Eine Sprache, welche zur Erstellung von einfachen Webseiten verwendet wird.

- JAVA** Ist sowohl eine Programmiersprache, als auch eine Technologie, welche einen Übersetzer und Bibliotheken enthält zur Übersetzung von Programmen die beispielsweise in Java geschrieben sind.
- JavaScript** Ist eine Skriptsprache, welche Webseiten die mit HTML und CSS erstellt wurden, erweitern sollte durch dynamische Inhalten und Interaktionen.
- JSON** steht für *JavaScript Object Notation* und ist ein lesbares Dateiformat zum Austausch von Daten.
- INNODB** Ein Speichersubsystem die für MySQL verwendet wird.
- Maven** Ist ein Projektmanagement Framework, welches häufig nur als Build-Management-Werkzeug gesehen/verwendet wird. Ähnlich zu sbt.
- MDX Multidimensional Expressions**, Datenbanksprache für OLAP-Datenbanken
- Metamodell** Ein Modell, welches verwendet wird um ein Modell zu modellieren.
- Model-View-Controller (MVC)** ein Architekturmuster, welches aus einem Model (dies ist meist eine Datenbank), einer View (die Anzeige, als das was der User sieht) und einem Controller besteht, der alles steuert.
- MySQL** Ist ein Datenbankverwaltungssystem
- Play** Play ist ein Web Framework, welches es Entwicklern erleichtert mit Hilfe von Java oder Scala Webanwendungen zu stellen.
- OData** Open Data Protocol, wurde von Microsoft entwickelt und stellt einen Datenzugriff zwischen zwei kompatiblen Systemen
- OLAP** Online Analytical Processing, Methode der analytischen Informationssysteme
- OpenUI5** OpenUI5 ist die Open Source Version von SAPUI5 und verfügt weniger Features und Bibliotheken als SAPUI5.
- SAP HANA** Eine In-Memory Datenbanklösung von SAP
- SAPUI5** Ist ein Framework, welches auf JavaScript basiert und von SAP entwickelt wurde. Die Open Source Variante ist OpenUI5.
- Sbt** Ist ein Build-Werkzeug für Scala und Java Projekte, ähnlich zu Maven.
- Softwarearchitekt** Entwirft Softwaresysteme, durch Entwurfsentscheidungen, Vorgaben und die Verwendung von Architekturwissen.
- Softwarearchitektur** Keine allgemeingültige Definition vorhanden, nach Bass [2012] ist es die Struktur bzw. die Menge an Strukturen, die Softwareelemente sowie die sichtbaren Eigenschaften und Beziehungen untereinander.
- Softwareevolution** Nach der initialen Entwicklung und Auslieferung verändern sich die Anforderungen an die Software und diese muss weiter angepasst werden, dieser Prozess der sich nennt sich Softwareevolution.
- Softwareerosion** Bezeichnet den Verfall eines Systems, häufig die Verschlechterung der Wartbarkeit durch Änderungen während der Softwareevolution.

Stack Exchange Ein Netzwerk von Frage- und Antwort-Seiten, zudem auch Stack Overflow gehört

Stack Overflow Eine Plattform auf der Nutzer Fragen zum Thema Softwareentwicklung stellen können und andere Nutzer sie beantworten.

Stakeholder Stakeholder sind alle Personen, die an einem Softwareprojekt beteiligt sind, wie beispielsweise die Entwickler und Auftraggeber, aber auch die späteren Nutzer.

Technologiewissen Ist das Wissen über eine bestimmte Technologie.

Third-Party Technologien Technologien von Drittanbietern, welche verwendet werden um deren Funktionalitäten einer Anwendung hinzuzufügen

Transitiv Eine Technologie A ist von einer Technologie B abhängig, und eine die Technologie B von einer Technologie C. Dann ist die Technologie transitiv abhängig von C.

Repository Verzeichnisse, häufig verwendeter Begriff im Zusammenhang mit GitHub

XML steht für *Extensible Markup Language* und ist eine Auszeichnungssprache

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ich bin mit einer Einstellung meiner Abschlussarbeit in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den 27.09.2017

Jennyfa Jurisch