

Softwaretechnik

Softwarearchitektur



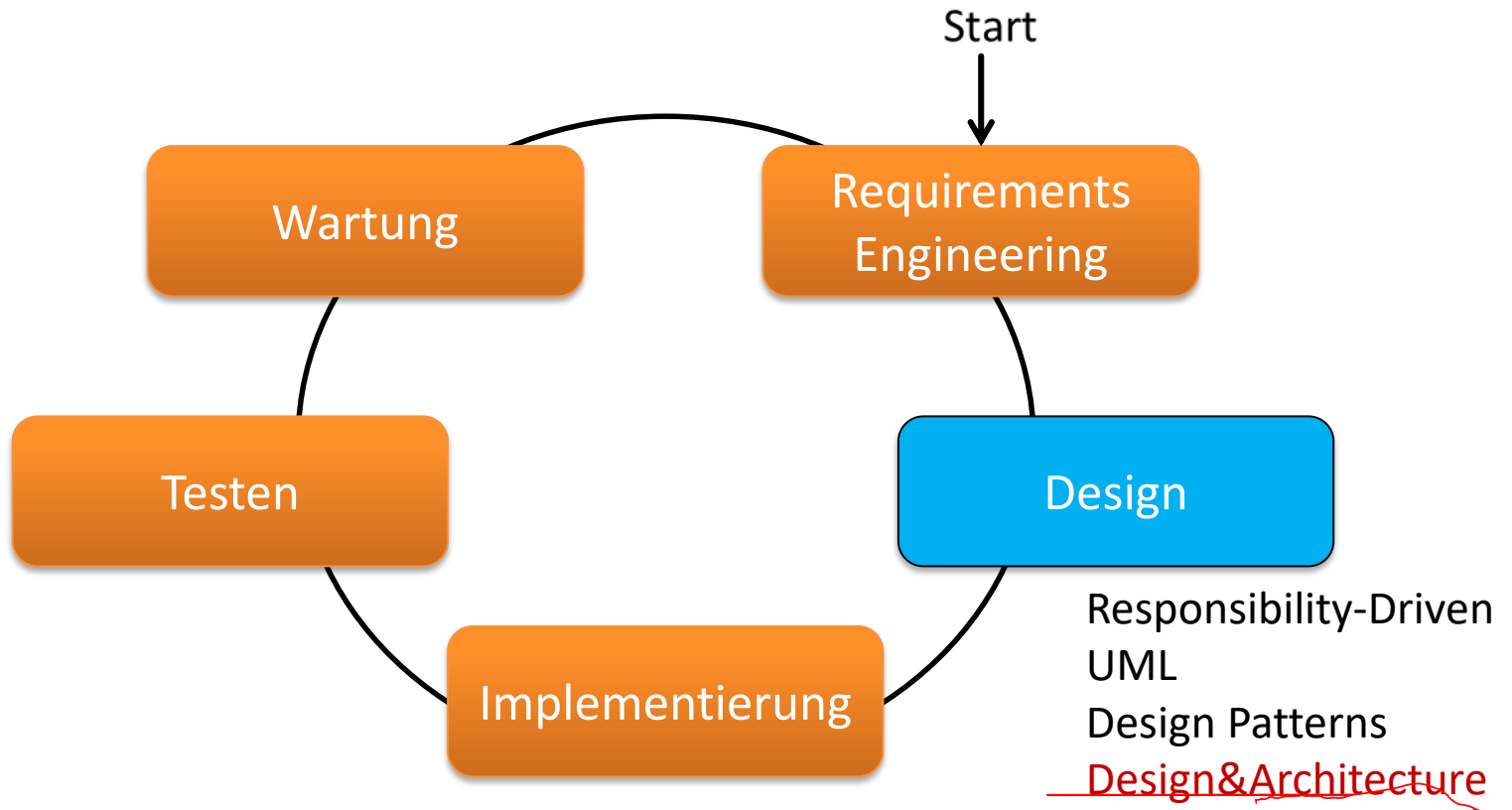
SOFTWARE
SYSTEME



UNIVERSITÄT
LEIPZIG

Prof. Dr.-Ing. Norbert Siegmund
Software Systems

Einordnung



Lernziele

- Arten von Software-Architekturen kennen
- Zusammenhang zwischen Design und Architektur einordnen
- Konkrete Architekturmuster wissen

Was ist Software Architektur?



Was ist Software Architektur?

A neat-looking drawing of some boxes, circles, and lines, laid out nicely in Powerpoint or Word, does not constitute an architecture.

— D'Souza & Wills

Was ist (wirklich) Software Architektur?

Die Architektur eines Systems besteht aus:

- der *Struktur(en) ihrer Teile*
 - einschließlich Design-, Test und Laufzeit Hardware und Software Teile
- den *extern sichtbaren Eigenschaften* dieser Teile
 - Module mit Interfaces, Hardware-Einheiten und Objekte
- den *Beziehungen und Bedingungen* zwischen ihnen

In anderen Worten:

The set of *design decisions* about any system (or subsystem) that keeps its implementors and maintainers from exercising “*needless creativity*”.

Design vs. Architektur

- Design beschreibt Aufbau von Subsystemen und Komponenten (fein granular)
 - Welche Klassen gibt es (in Modul X) und wie interagieren sie?
- Architektur beschreibt den groben Aufbau eines Systems (welche Komponenten gibt es?)
 - Welche Komponenten / Module gibt es und wie interagieren sie?

Sub-systeme, Module und Komponenten

- Ein Sub-system ist selbst ein System, dessen Operation *unabhängig* von den Leistungen und Funktionen anderer Sub-systeme ist.
- Ein Modul ist eine Systemkomponente, die *Dienstleitungen / Funktionen anbietet*, welche andere Komponenten benötigen, aber welche nicht als komplett separates System angesehen werden.
- Eine Komponente ist eine *unabhängig auslieferbare Einheit* von Software, die ihr Design und Implementierung eingeschlossen hat (hiding) und ihr Interface zur Außenwelt anbietet, so dass sie mit anderen Komponenten zusammengefasst werden kann, um ein größeres System zu bilden.



Arten von SW Architektur



Parallelen zur (echten) Architektur

- Architekten sind die *Schnittstelle* zwischen den Kunden und den Auftragnehmern, die das System/Gebäude bauen
- Eine schlechte Architektur für ein Gebäude *kann nicht mehr durch gute Konstruktion gerettet werden*— gleiches gilt für Software
- Es gibt *spezielle Typen* von Gebäuden und Software-Architekten.
- Es gibt *Schulen oder Styles* des Bauens und der Software – Architektur.

Architectural Styles

*An architectural style defines a **family of systems** in terms of a pattern of structural organization. More specifically, an architectural style defines a vocabulary of **components** and **connector** types, and a set of **constraints** on how they can be combined.*

— Shaw and Garlan

SW-Architekturen für große Softwaresysteme

Monolith

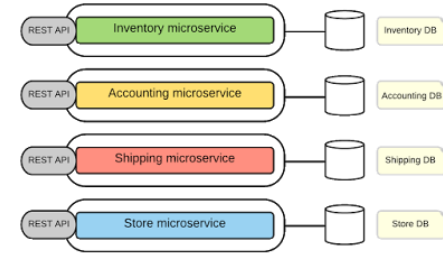
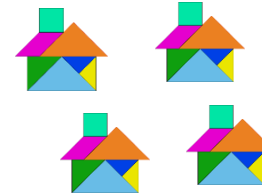
System of Systems

Schichtenarchitektur

Komponenten
orientiert

Service
orientierte

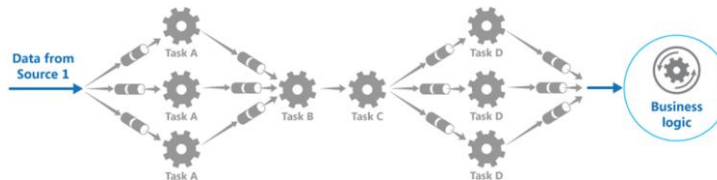
MicroServices




Spezialarchitekturen:

Pipes and Filters


Peer-to-Peer



Welche Architektur?



Koffer, Rucksäcke & Taschen ▾



Prime music Über 2 Millionen Songs. Ohne Werbung. ▸

Alle Kategorien ▾


Jonas' Amazon

Angebote

Gutscheine

Verkaufen


Hilfe

DE 

Hallo, Jonas
Mein Konto ▾

Mein Prime ▾

Meine Listen ▾

 Einkaufswagen

Amazon Fashion

DAMEN ▾

HERREN ▾

KINDER & BABY ▾

GEPÄCK ▾

MARKEN ▾

SALE ▾

KOSTENLOSER RÜCKVERSAND
Innerhalb von 30 Tagen ▸ Mehr Informationen

Koffer, Rucksäcke & Taschen ▸ Taschen ▸ Umhängetaschen





AmazonBasics Tasche für Laptop / Tablet mit Bildschirmdiagonale 15,6 Zoll / 39,6 cm

von AmazonBasics

★★★★★ ▾ 616 Kundenrezensionen | 64 beantwortete Fragen

Bestseller Nr. 1 in Notebook-Aktentaschen

Preis: **EUR 19,49**  **Prime**

Alle Preisangaben inkl. USt

Auf Lager.

Verkauf und Versand durch Amazon. Geschenkverpackung verfügbar.

Größe: **15,6 Zoll / 39,6 cm**

11,6 Zoll / 29,5 cm EUR 13,99  Prime	14,1 Zoll / 35,8 cm EUR 14,99  Prime	15,6 Zoll / 39,6 cm EUR 19,49  Prime	17,3 Zoll / 44 cm EUR 19,99  Prime
--	--	--	--

- Schlanke und kompakte Tasche, perfekt für Laptops bis 29,5 cm (15,6 Zoll), ohne unnötigen Ballast
- Zubehörtaschen für Maus, iPod, Handy und Stifte
- Einschließlich gepolstertem Schulterriemen

▸ Weitere Produktdetails

[Falsche Produktinformationen melden](#)

Möchten Sie Ihr Elektro- und Elektronik-Gerät kostenlos recyceln? ([Erfahren Sie mehr.](#))

Teilen    

Menge: 1 ▾

 In den Einkaufswagen

oder

 Jetzt mit 1-Click® kaufen

Innerhalb von **3h 9min** bestellen und Lieferung erfolgt am:
Samstag, 21 Jan
(GRATIS Premiumversand)

Lieferort:
Jonas Hecht- Weimar - 99423 ▾

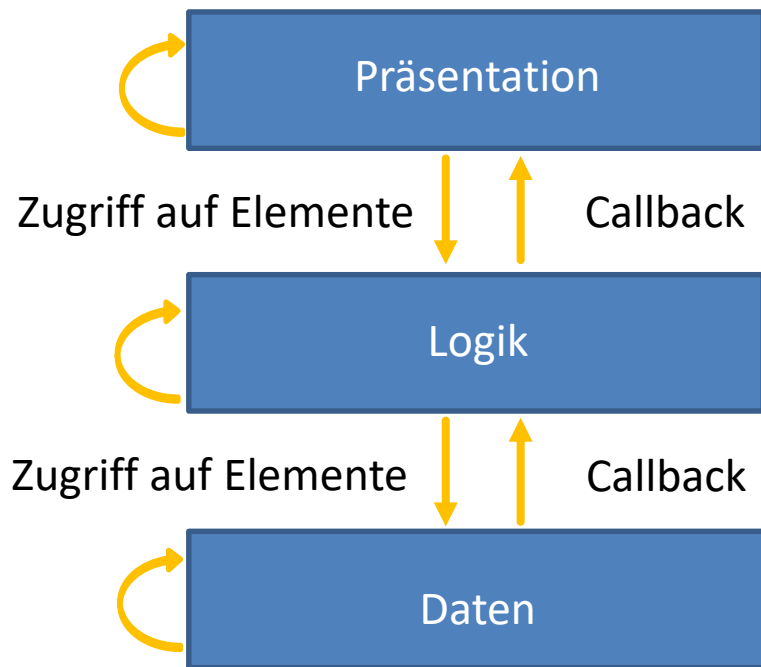
☐ Dies ist ein Geschenk

Auf die Liste ▾

Möchten Sie verkaufen?
[Bei Amazon verkaufen](#)

Schichtenarchitekturen

- Eine Schichtenarchitektur organisiert ein System in eine Menge von Schichten, wobei jede Schicht eine Menge von Leistungen / Funktionen für die Schicht “darüber” anbietet.



Vorteile:

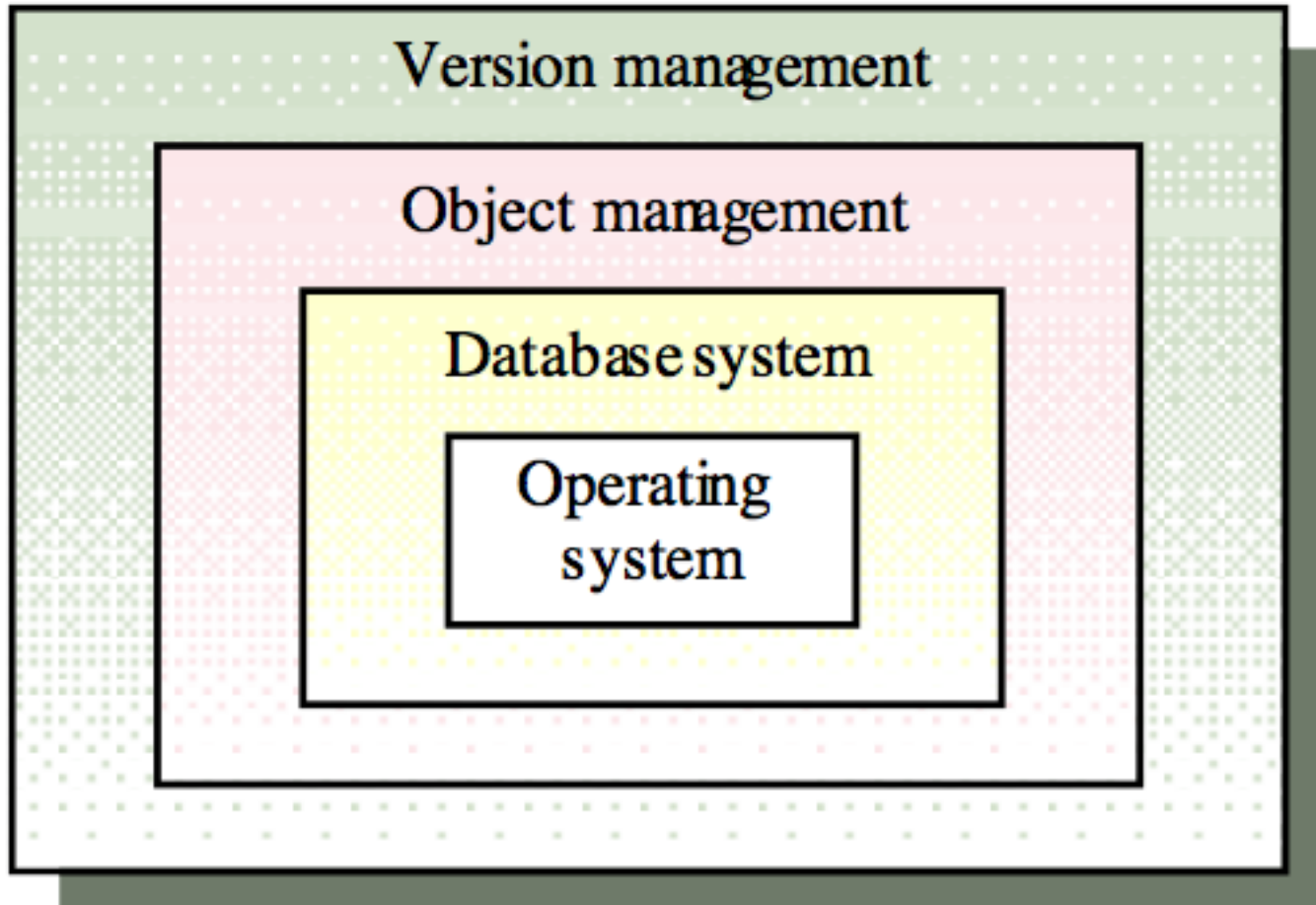
- Inkrementelle Entwicklung von Subsystemen
- Wenn ein Interface einer Schicht sich ändert, *sind nur benachbarte Schichten betroffen*
- Modularität, Kohäsion

Layered Architectures

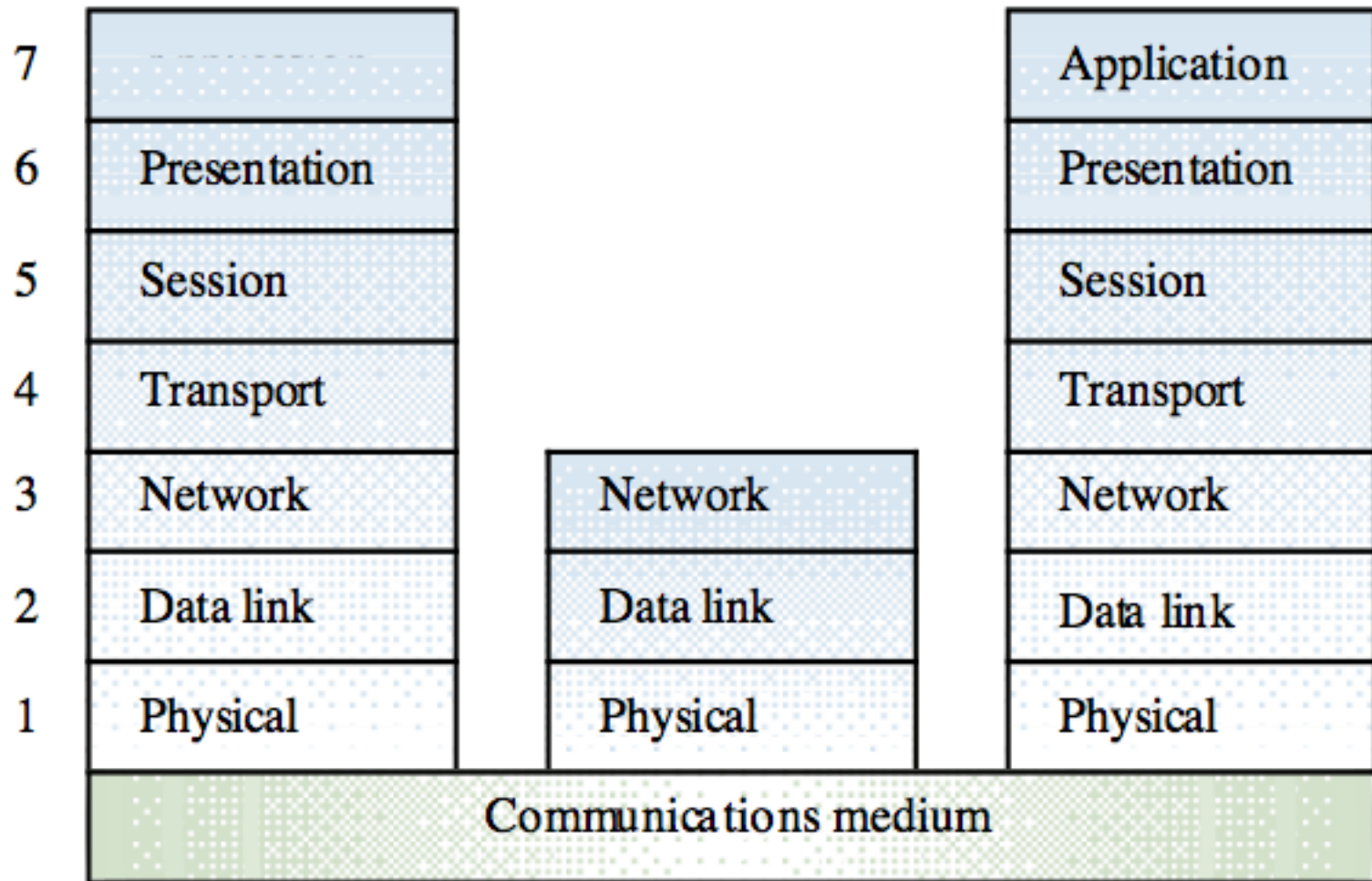
Eine Schichtenarchitektur organisiert ein System in eine Menge von Schichten, wobei jede Schicht eine Menge von Leistungen / Funktionen für die Schicht “darüber” anbietet.

- Schichten sind normalerweise *beschränkt*, so dass Elemente nur
 - Andere Element in der gleichen Schicht oder
 - Elemente von der Schicht darunter sehen können
- *Callbacks* können verwendet werden, um mit höheren Schichten zu kommunizieren
- Unterstützt die *inkrementelle Entwicklung* von Sub-systemen in unterschiedlichen Schichten
 - Wenn ein Interface einer Schicht sich ändert, *sind nur benachbarte Schichten betroffen*.

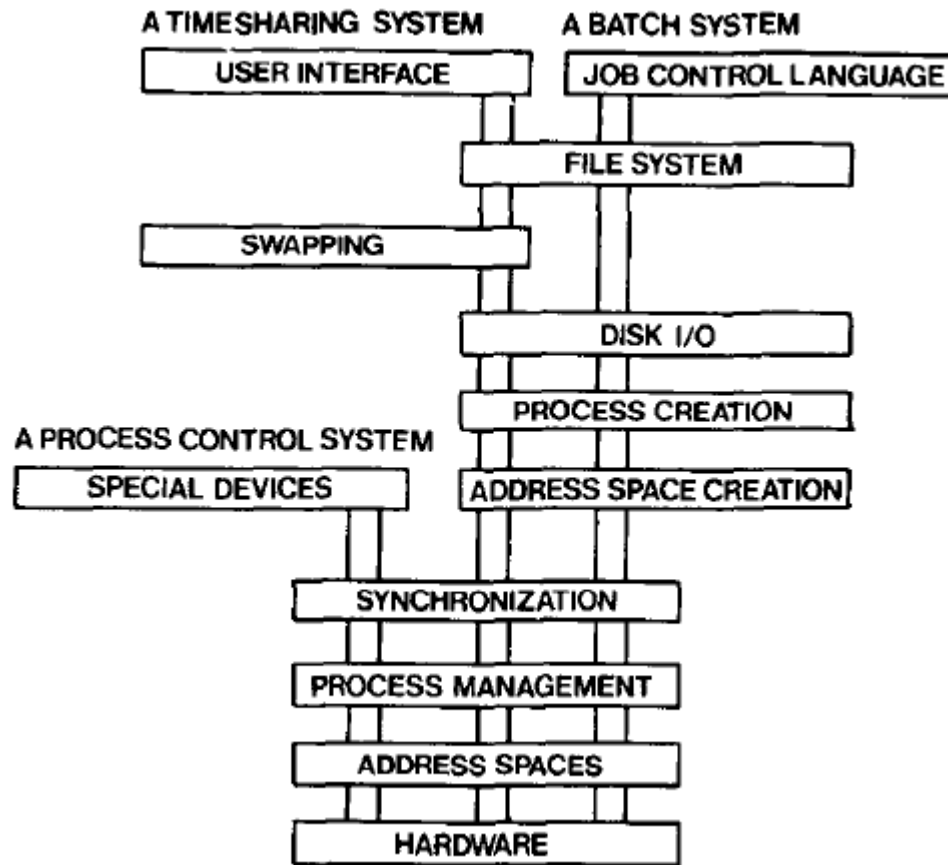
Version Management System



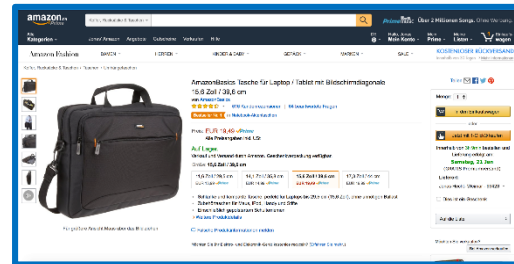
OSI Reference Model



Operating System Family



Beispiel: 3-Layer Architektur



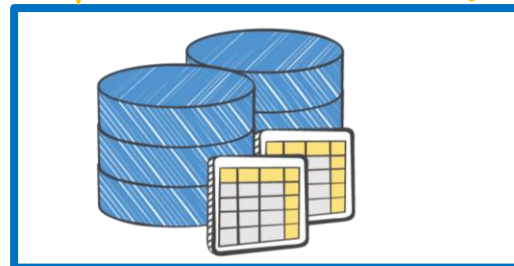
JavaScript Implementierung

Apache, Tomcat



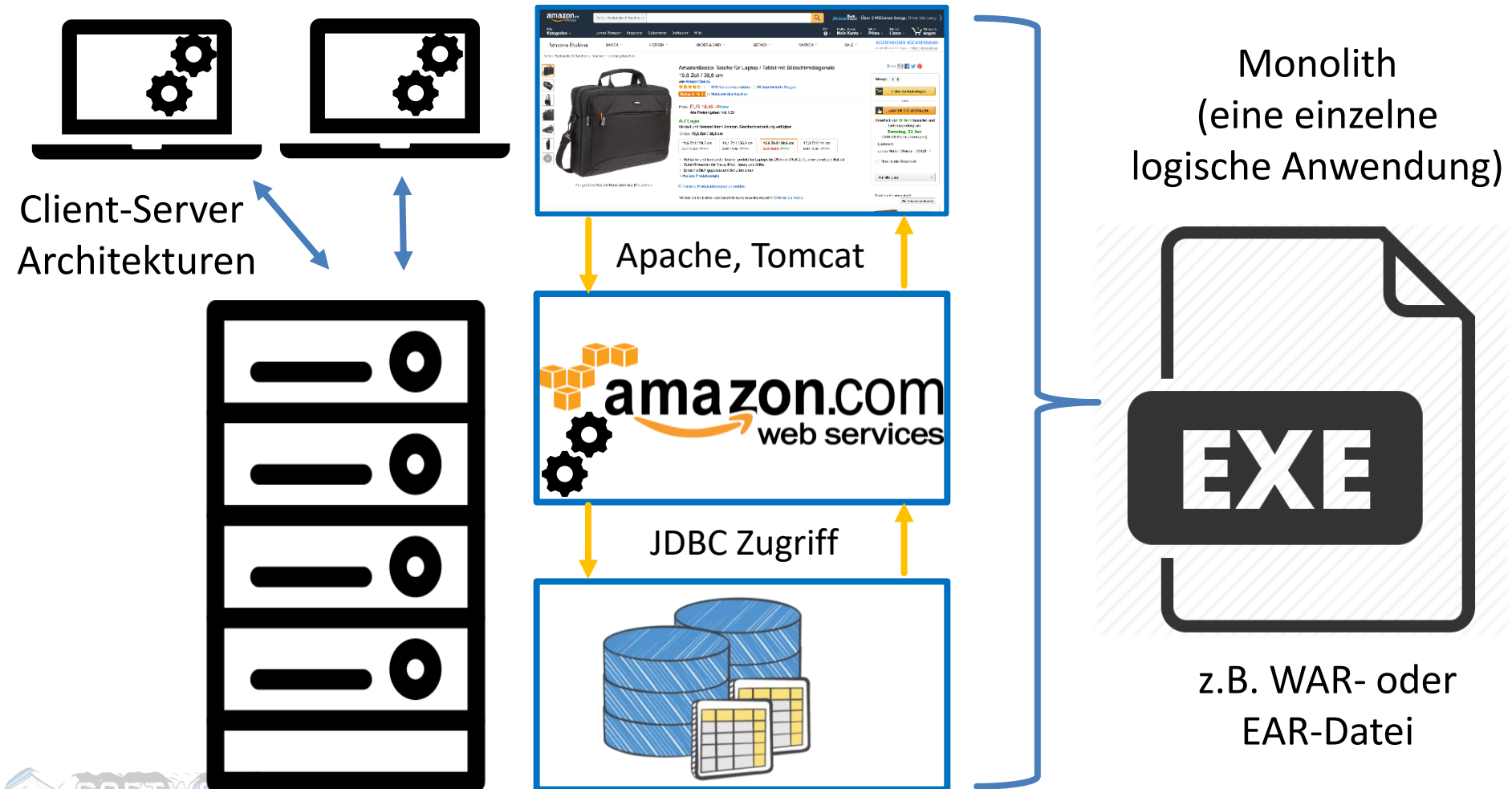
Java Implementierung

JDBC Zugriff



Oracle, etc.

Client-Server & Monolith



Client-Server Architektur

Eine Client-Server Architektur *verteilt Applikationslogik und Funktionalität* zu einer Anzahl von Klienten (clients) und Server-Subsystemen, wobei jede potentiell auf einer unterschiedlichen Maschine läuft und über das Netzwerk kommuniziert.

Vorteile:

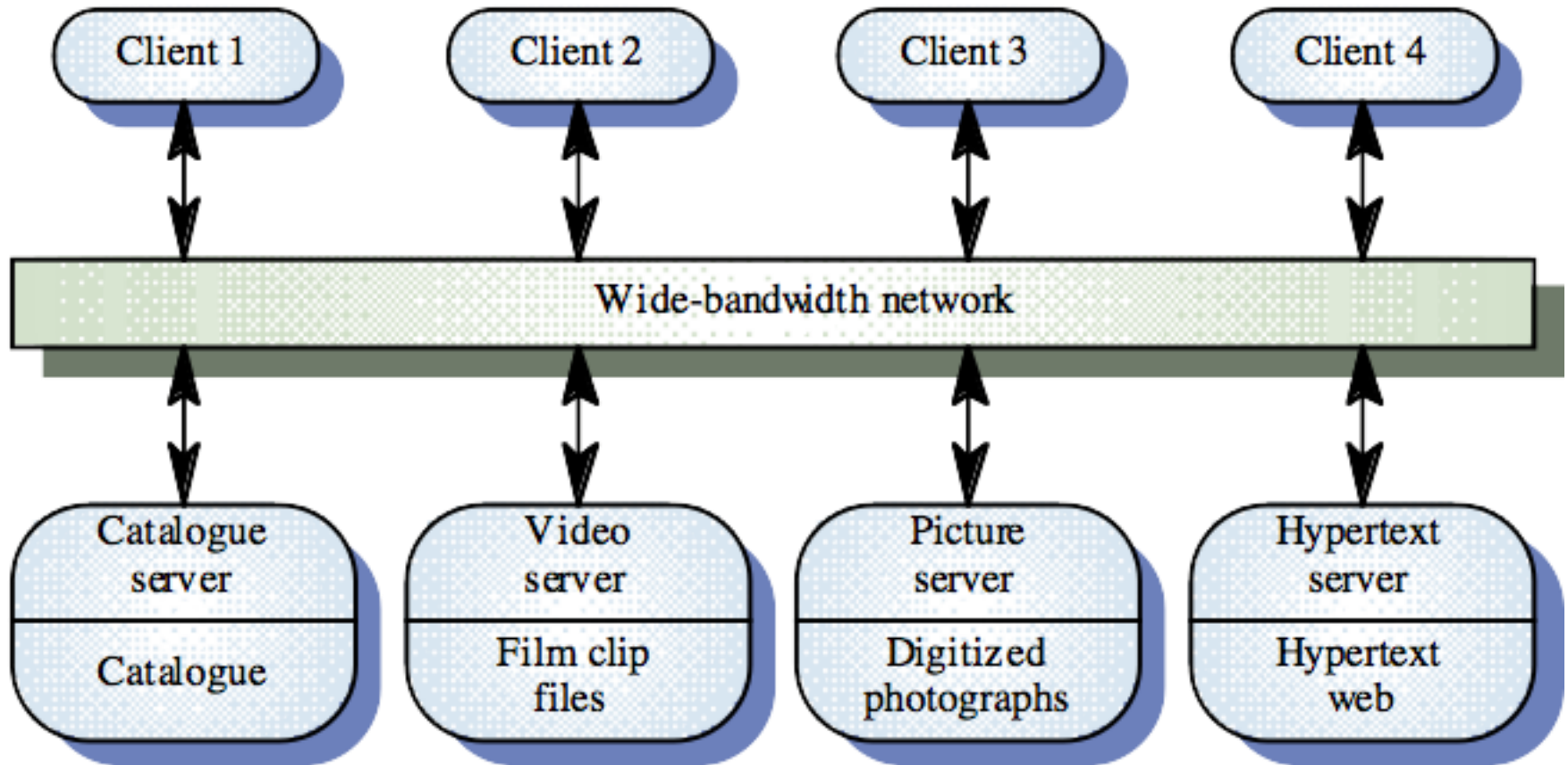
- Einfache *Datenverteilung*
- Effektive *Hardwareauslastung*
- Einfaches *Hinzufügen* neuer Server

Nachteile:

- Kein *geteiltes Datenmodell*
- *Redundante* Verwaltung
- Evtl. *zentrale Registrierung* erforderlich
(welcher Server stellt welche Dienstleistung zur Verfügung?)



Film and Picture Library



Und ohne Web?

Wie würde Sie die Architektur entwerfen, wenn wir eine Desktop-Anwendung schreiben würden?

Kommunikation / Controller

Apache, Tomcat, JavaScript

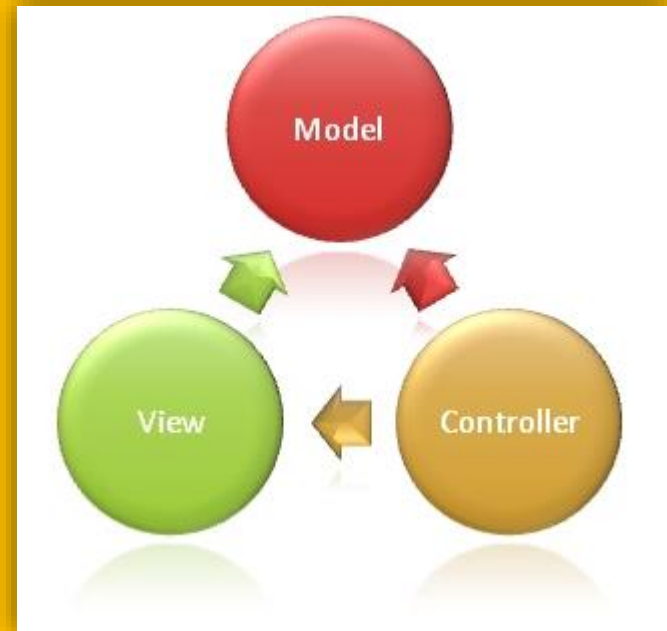
Sicht / View

JavaScript Frontend

Java Implementierung, DB, ...

Model / Business-Logik / Daten

Model-View-Controller



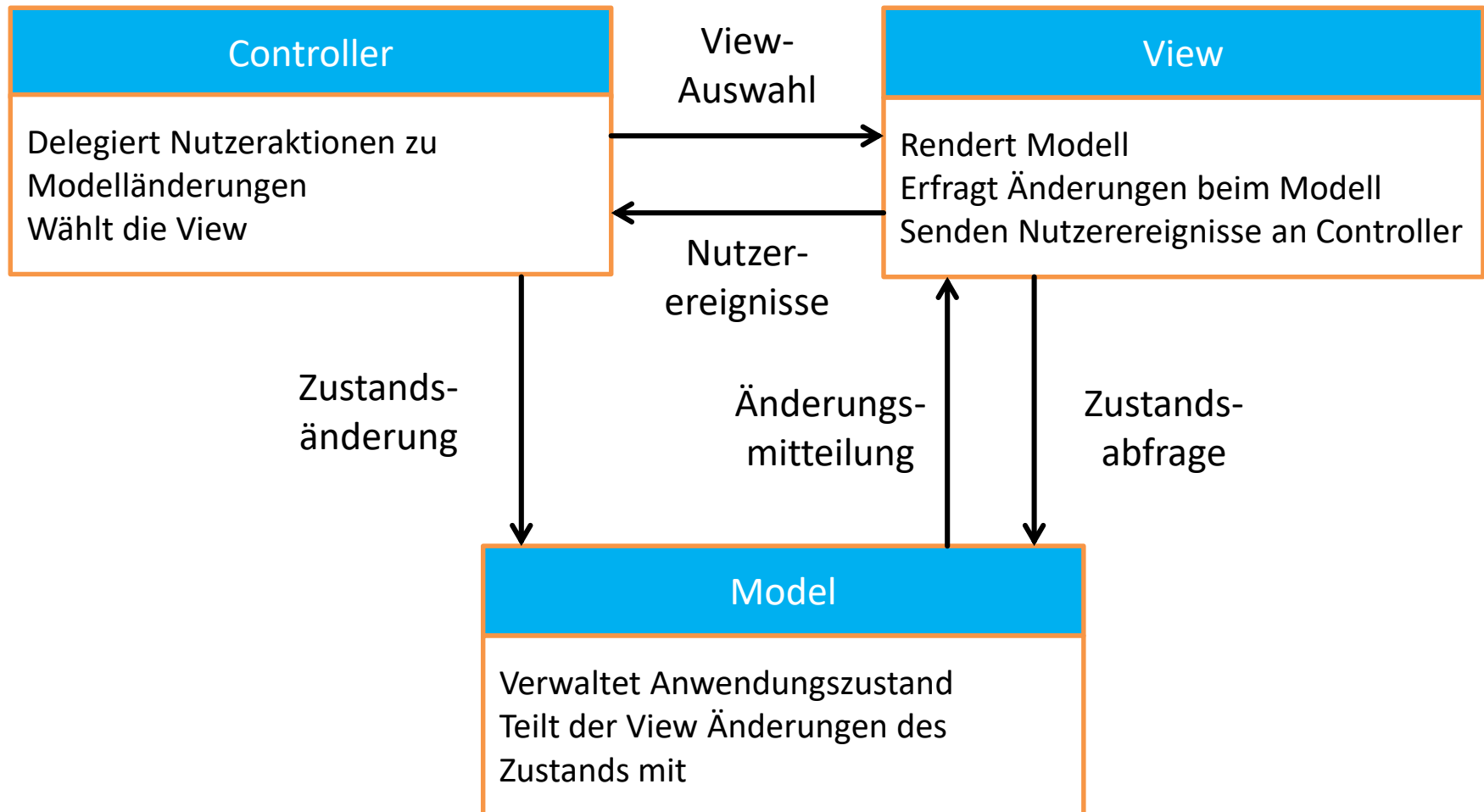
Model-View-Controller (MVC) Architektur

Idee: Separiere *Präsentation* und *Interaktion* von den *Daten* des Systems

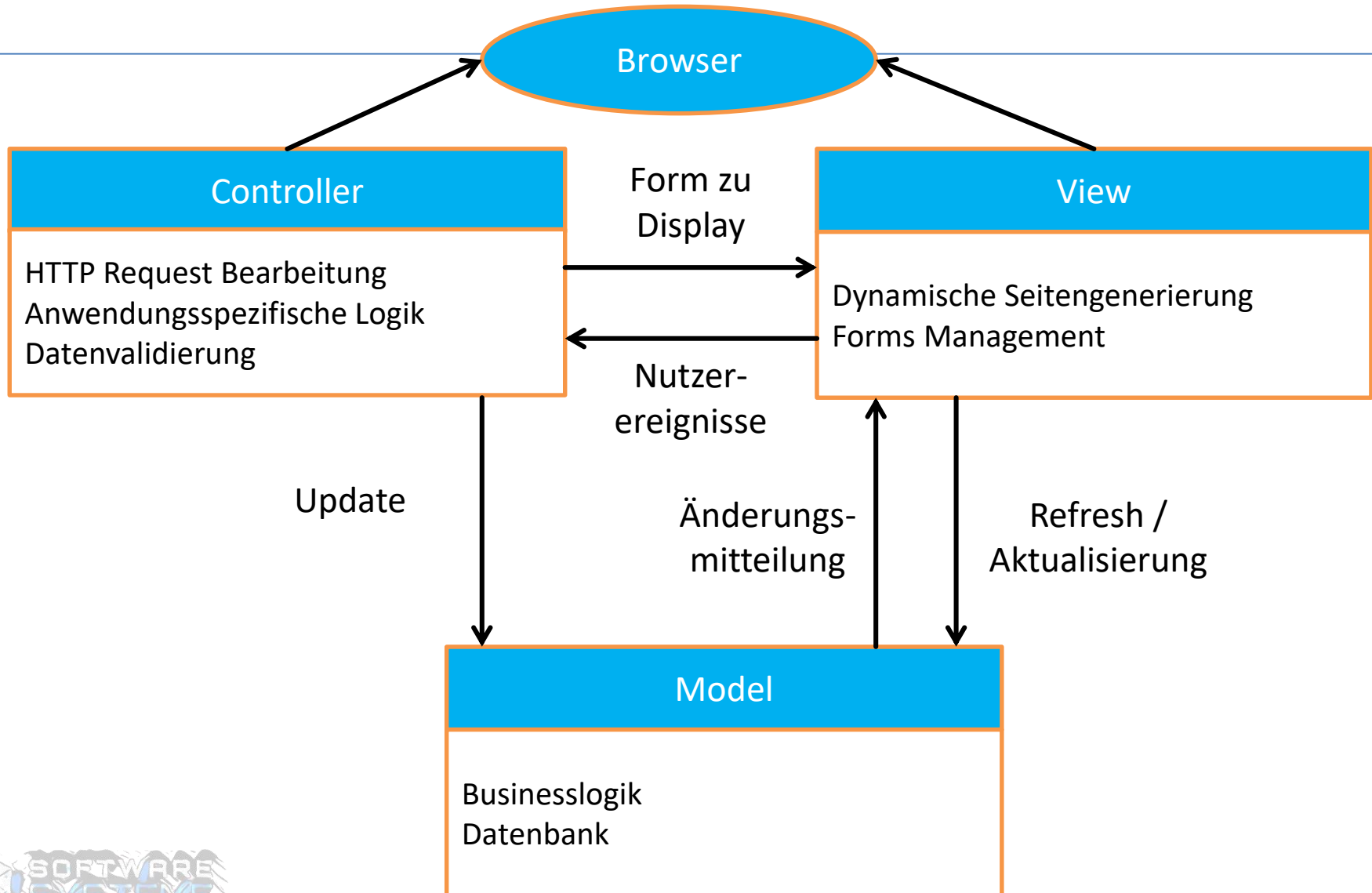
- Das System ist strukturiert in drei Komponenten:
 - *Model*: verwaltet Systemdaten und Operationen auf den Daten
 - *View*: Präsentiert die Daten zum Nutzer
 - *Controller*: händelt Nutzerinteraktion; schickt Informationen zur View und zum Model
- Nützlich, wenn es mehrere Wege gibt auf die Daten zuzugreifen
- Ermöglicht das Ändern der Daten unabhängig von deren Repräsentation
- Unterstützt unterschiedliche Präsentationen der gleichen Daten



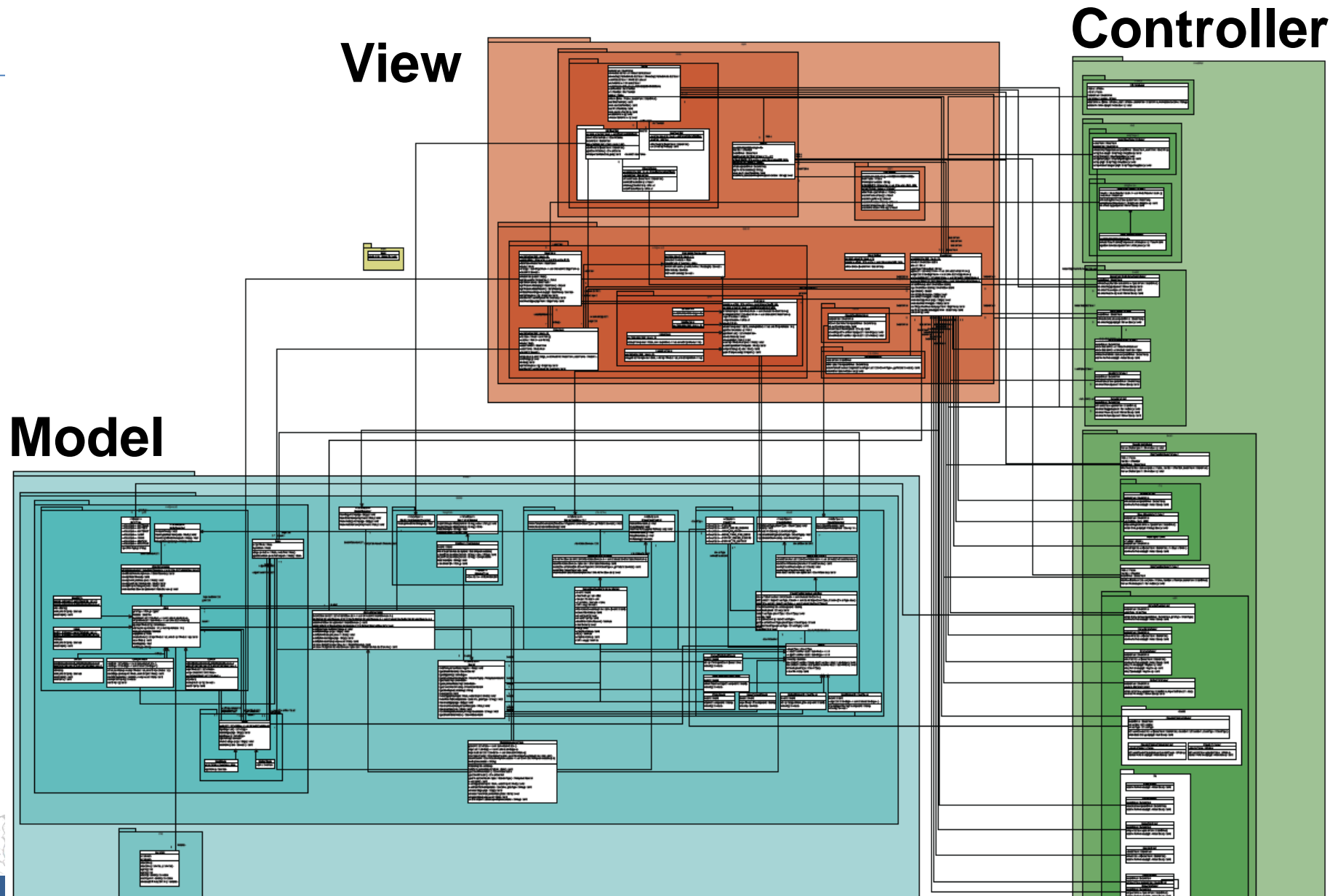
MVC Übersicht



MVC Beispiel

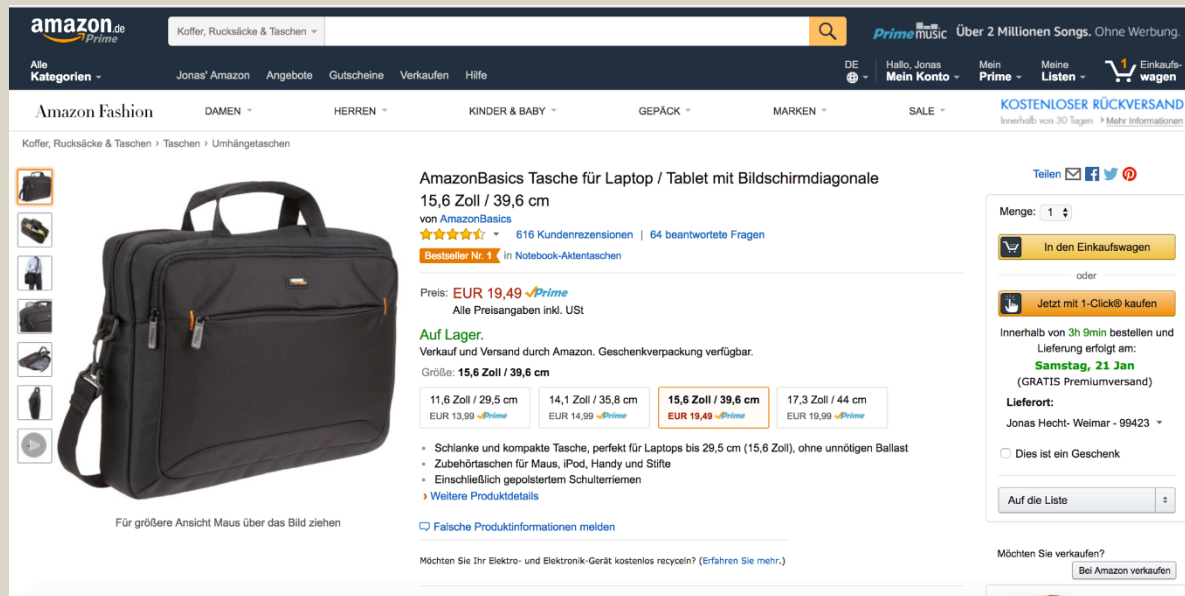


MVC in Action (Circuit Simulation, SEP'11)



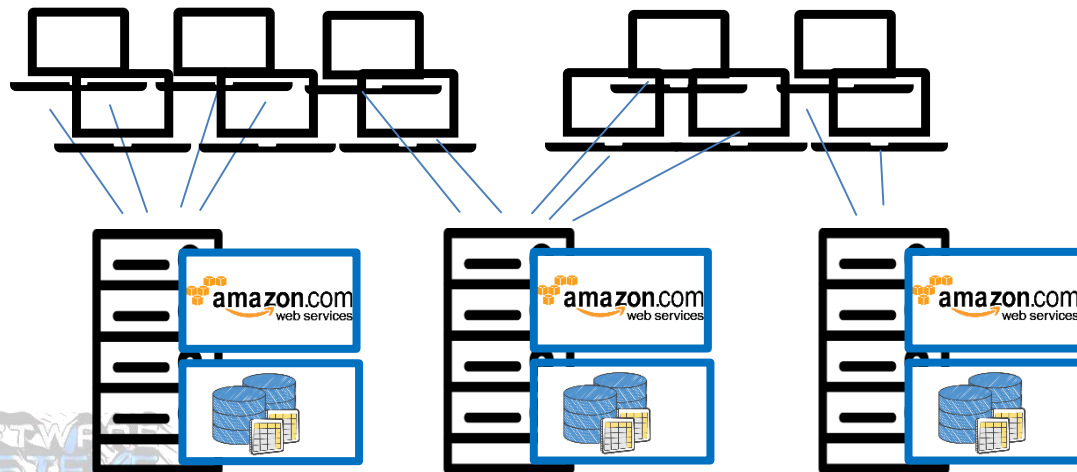
Probleme der Architekturen

- Welche Probleme erwarten Sie, wenn wir diese Architekturen in der Praxis für unser Beispiel einsetzen?
- Hinweise:
 - Großes Softwaresystem
 - Hohes Nutzeraufkommen



Probleme traditioneller Architekturmuster für große Softwaresysteme

- Häufige Änderungen
 - Monolithisches System muss komplett neu gebaut werden
 - Abhängigkeiten zwischen Subsystemen erschweren und verzögern Änderungen
- Skalierbarkeit der Hardware
 - Alle Architekturen sind schwer skalierbar, selbst Client-Server



Neue Probleme:

- Verteilte, *replizierte* Daten
- *Konsistenz* und Synchronität?
- *Gesamtes* System repliziert, obwohl nur Subsysteme ausgelastet sind

Probleme traditioneller Architekturmuster für große Softwaresysteme

- Was passiert bei Ausfällen und Fehlern von Subsystemen?
 - Oft zu starke Kopplung der Subsysteme und kaum Möglichkeit des „Fail-overs“
 - Alternative Views bei MVC möglich, aber alternative Modelle?
- Wie wird das laufende System geupdatet?
 - Herunterfahren der Server ist keine Option
- Wie kann die Entwicklung von Subsystemen parallelisiert werden?
 - Schichtenarchitektur und MVC oft zu grob-granular
 - Komponenten und Services nicht gut bei querschneidenden Belangen und erfordern zu viel Glue-Code / Kommunikation
- Wie vereinfache ich Testen? Usw.

Gewünschte Eigenschaften

- Schneller Austausch von Subsystemen ohne, dass das gesamte System betroffen ist (lose Kopplung + hohe Modularität)
- Skalierbarkeit bei großen Lasten von einzelnen Subsystemen (Beispiel: Black Friday)
- Weniger Abstimmungsaufwand innerhalb der Unternehmensorganisation (bei Entwicklern und Sachverständigen)
- Parallelisierung in der Entwicklung sowie Anwendbarkeit von agilen Methoden der Softwareentwicklung
- Einfache Testbarkeit von Subsystemen

Welche Architektur?

amazon.de Prime

Koffer, Rucksäcke & Taschen

Alle Kategorien

Jonas' Amazon Angebote Gutscheine Verkaufen Hilfe

DE

Hallo, Jonas Mein Konto

Mein Prime

Meine Listen

Einkaufswagen

Prime music Über 2 Millionen Songs. Ohne Werbung.

Amazon Fashion DAMEN HERREN KINDER & BABY GEPÄCK MARKEN SALE

KOSTENLOSER RÜCKVERSAND Innerhalb von 30 Tagen > Mehr Informationen

Koffer, Rucksäcke & Taschen > Taschen > Umhängetaschen

AmazonBasics Tasche für Laptop / Tablet mit Bildschirmdiagonale 15,6 Zoll / 39,6 cm
von AmazonBasics
★★★★★ 616 Kundenrezensionen | 64 beantwortete Fragen
Bestseller Nr. 1 in Notebook-Aktentaschen

Preis: **EUR 19,49** Prime
Alle Preisangaben inkl. USt

Auf Lager.
Verkauf und Versand durch Amazon. Geschenkverpackung verfügbar.

Größe: **15,6 Zoll / 39,6 cm**

11,6 Zoll / 29,5 cm EUR 13,99 Prime	14,1 Zoll / 35,8 cm EUR 14,99 Prime	15,6 Zoll / 39,6 cm EUR 19,49 Prime	17,3 Zoll / 44 cm EUR 19,99 Prime
--	--	--	--------------------------------------

- Schlanke und kompakte Tasche, perfekt für Laptops bis 29,5 cm (11,6 Zoll), ohne unnötigen Ballast
- Zubehörtaschen für Maus, iPod, Handy und Stifte
- Einschließlich gepolstertem Schulterriemen

[Weitere Produktdetails](#)

[Falsche Produktinformationen melden](#)

Möchten Sie Ihr Elektro- und Elektronik-Gerät kostenlos recyceln? ([Erfahren Sie mehr.](#))

Teilen

Menge: 1

In den Einkaufswagen

oder

Jetzt mit 1-Click® kaufen

Innerhalb von **3h 9min** bestellen und Lieferung erfolgt am:
Samstag, 21 Jan
(GRATIS Premiumversand)

Lieferort:
Jonas Hecht- Weimar - 99423

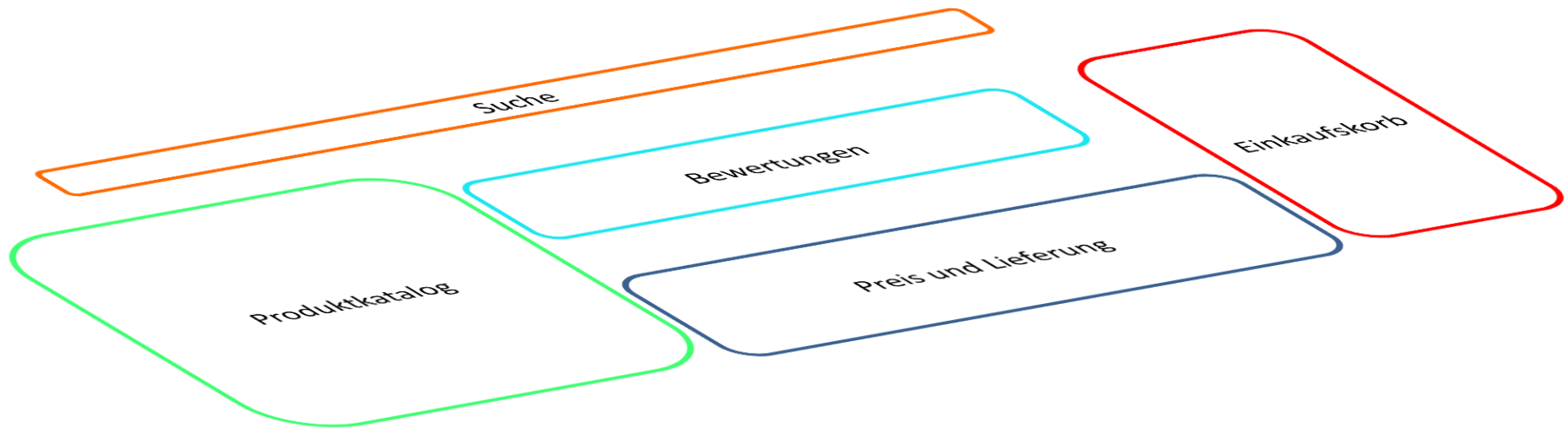
☐ Dies ist ein Geschenk

Auf die Liste

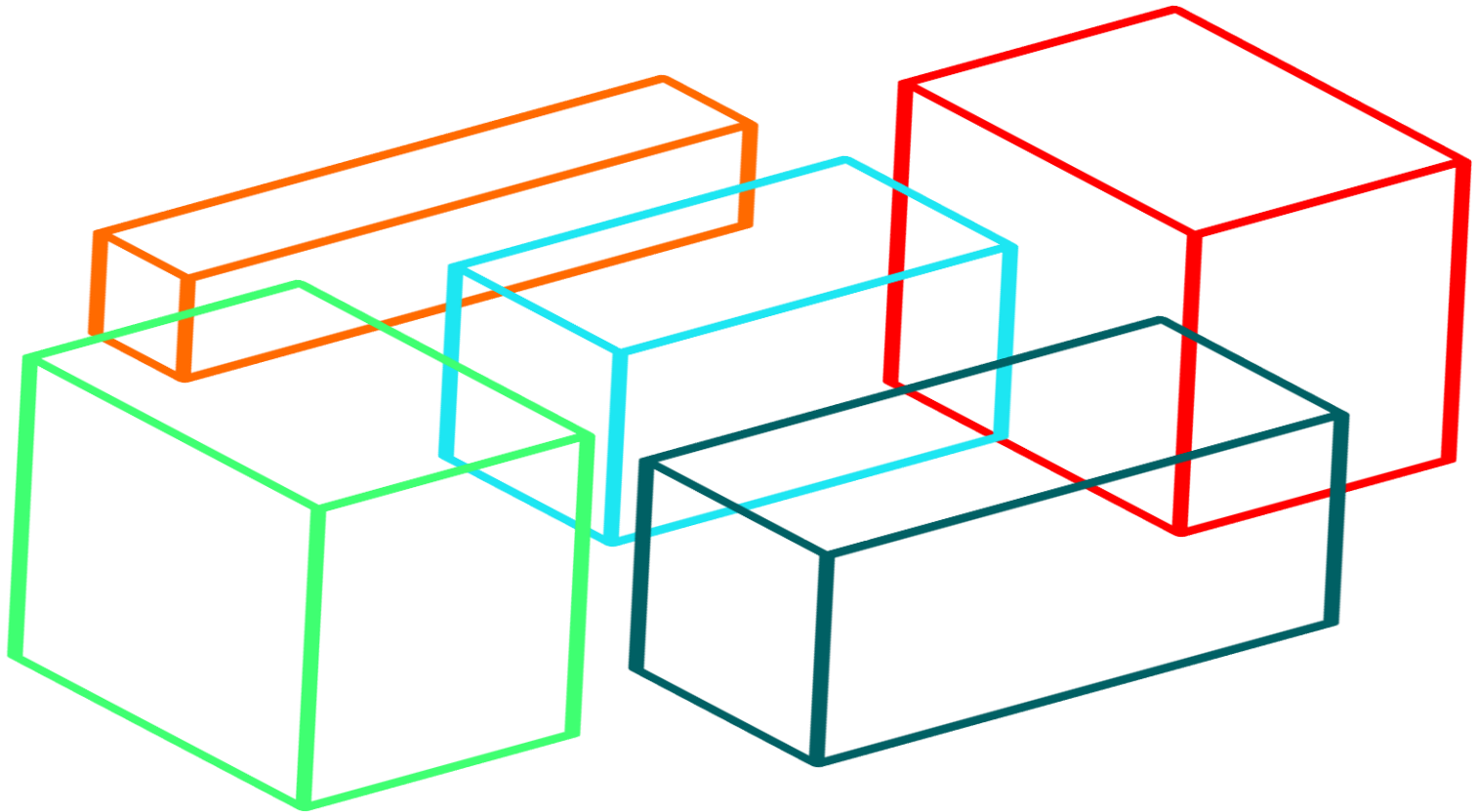
Möchten Sie verkaufen?
Bei Amazon verkaufen

Für größere Ansicht Maus über das Bild ziehen

Zerlegung des Systems...



Zerlegung des Systems...



... nach fachlichen Funktionen

Produkt-
katalog

Suche

Preis und
Lieferung

Bewertungen

Einkaufskorb

nach fachlichen Funktionen

Preis: **EUR 19,49** ✓ Prime
Alle Preisangaben inkl. USt

Auf Lager.

Verkauf und Versand durch Amazon. Geschenkverpackung verfügbar.

Größe: 15,6 Zoll / 39,6 cm

11,6 Zoll / 29,5 cm
EUR 13,99 ✓ Prime

14,1 Zoll / 35,8 cm
EUR 14,99 ✓ Prime

15,6 Zoll / 39,6 cm
EUR 19,49 ✓ Prime

17,3 Zoll / 43,9 cm
EUR 19,99 ✓ Prime

AmazonBasics Tasche für Laptop / Tablet mit Bildschirmdiagonale
15,6 Zoll / 39,6 cm

von AmazonBasics

★★★★★

616 Kundenrezensionen | 64 beantwortete Fragen

Bestseller Nr. 1 in Notebook-Aktentaschen

Menge: 1

In den Einkaufswagen

oder

Jetzt mit 1-Click® kaufen

Innerhalb von 3h 9min bestellen und

Lieferung erfolgt am:

Samstag, 21 Jan

(GRATIS Premiumversand)

Lieferort:

Jonas Hecht - Weimar - 99423

☐ Dies ist ein Geschenk

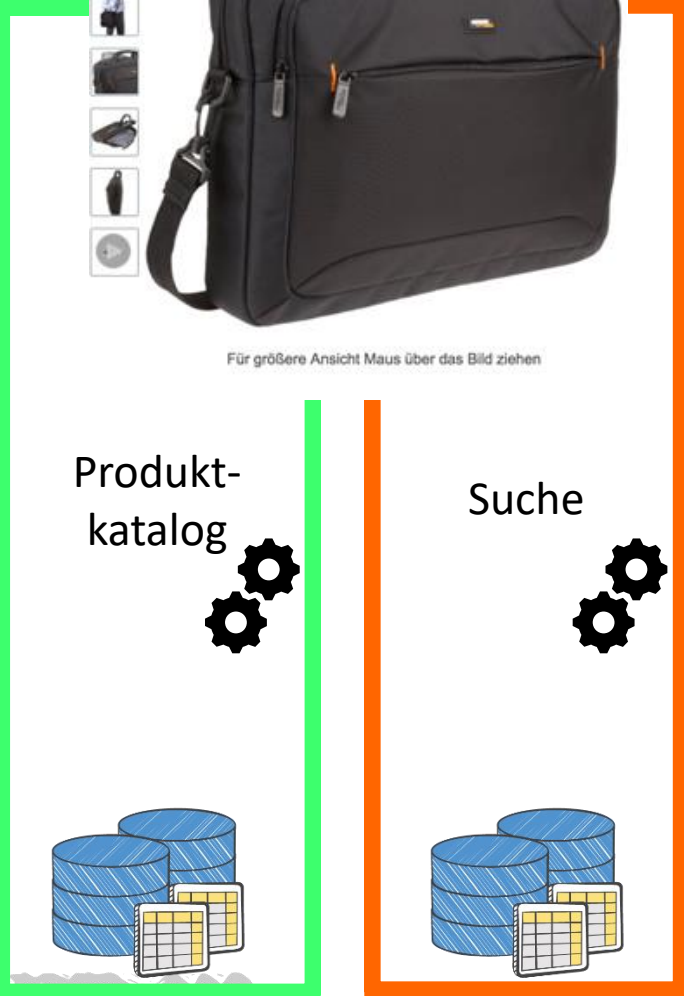
MicroService

MicroService

MicroService

MicroService

MicroService



Frontend

Business-Logik

Daten

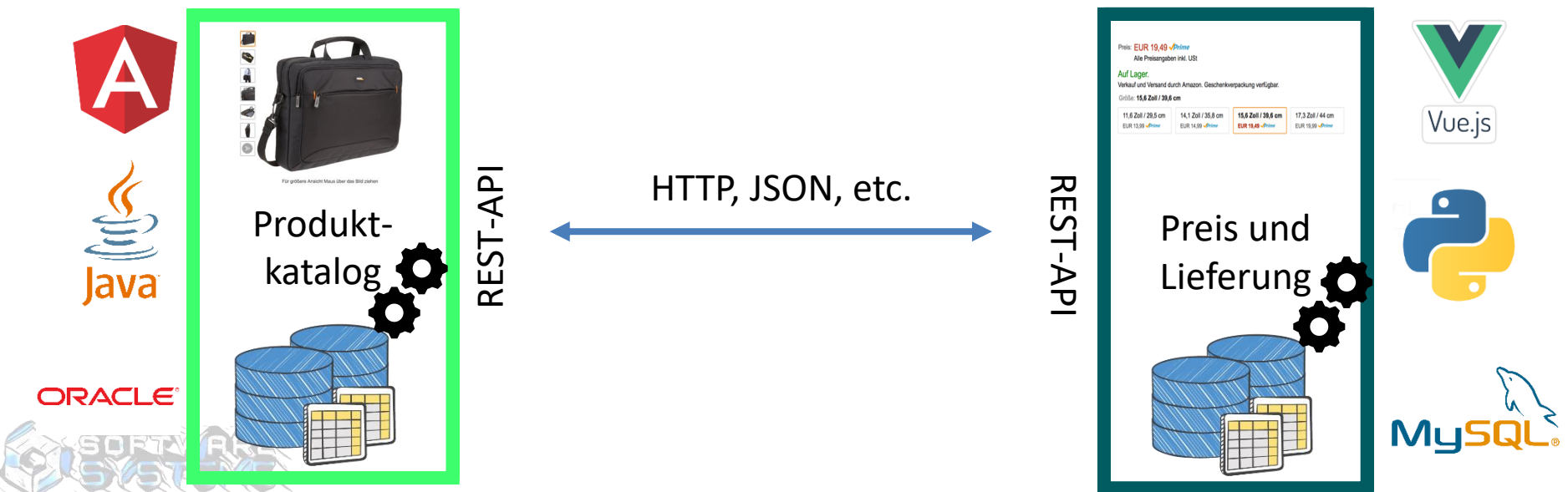
MicroServices: Idee

- Jede *fachliche* Funktion in einen *autonomen, unabhängigen Subsystem* modularisieren
- Jeder Service bietet die *vollständige* Funktionalität für die jeweilige fachliche Aufgabe an
 - Alle *Daten*, die hierfür notwendig sind
 - Gesamte *Business-Logik* für die Aufgabe
 - Alle *Sichten* und *Interaktionsmöglichkeiten*
- Teamzusammensetzung ideal für agile Entwicklung
 - Fachexperte, Entwickler, Tester, DevOps

Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure. -- Melvyn Conway, 1967

Von MicroService zum Softwaresystem

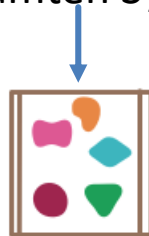
- Anwendung besteht aus einer Menge an MicroServices
 - Jeder hat eigene Prozesse und Daten
 - Leichtgewichtige Kommunikation (meist REST-API)
 - Unabhängig voneinander auslieferbar, testbar, entwickelbar
- Maximale Unabhängigkeit der MicroServices



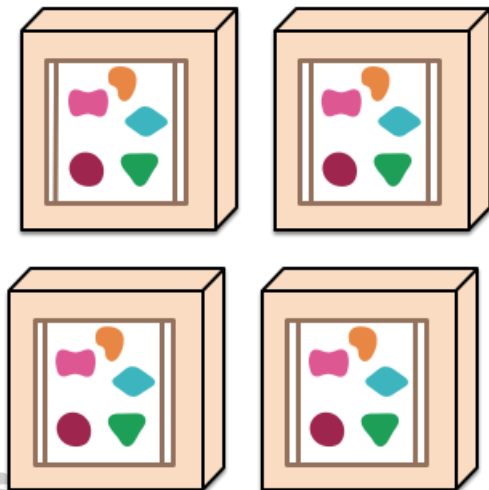
Monolith vs. MicroServices

Monolith

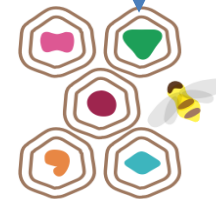
Änderung an einem Teil
bedarf Bauen und Ausliefern
des gesamten Systems



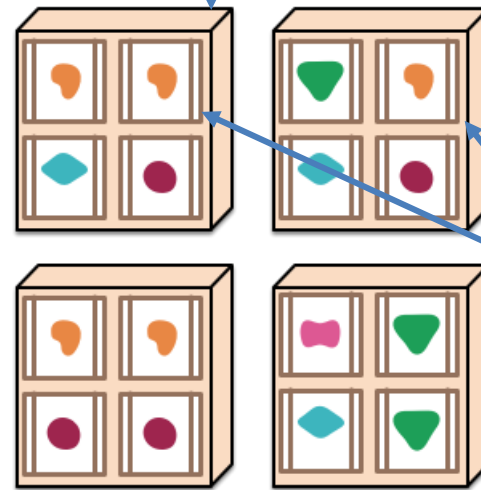
Skalierung betrifft alle
Systemteile



Unabhängiges entwickeln,
testen, bauen, ausliefern
einzelner Teile



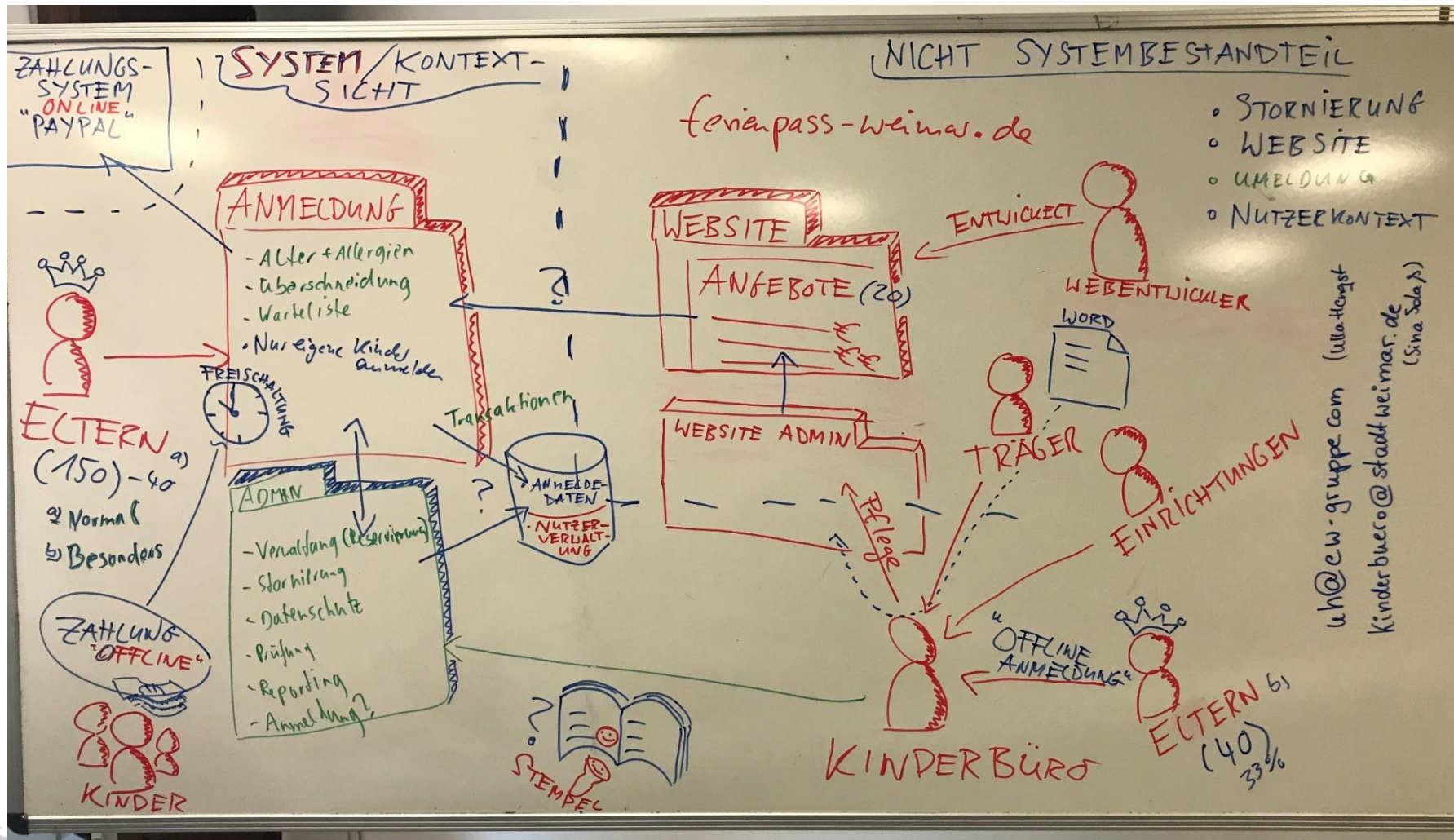
Skalierung von nur
tatsächlich überlasteten Funktionen



Austausch von
einzelnen Services
möglich

MicroServices

Erfahrungen Ihrer VorgängerInnen



Und in der Praxis?

amazon NETFLIX ► zalando

UBER

 TheGuardian


REWE digital

ebay

 28. SEPTEMBER 2018
JUG SAXONYDAY

Kritische Diskussion

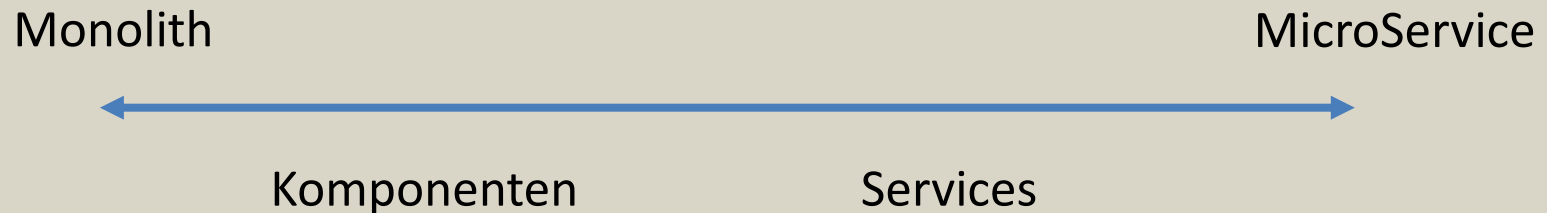
- Was sind mögliche Nachteile einer MicroService Architektur?
- Welche Besonderheiten und Voraussetzungen sollten beim Einsatz dieser Architektur betrachtet werden?
 - MicroServices für Word oder Virens Scanner?

Nachteile von MicroServices

- Benötigt richtigen „Mindset“ der Entwickler und klare Definition von eines MicroServices
- Immer noch hoher Kommunikationsaufwand zwischen Teams
- Moderne DevOps Pipeline erforderlich, Stichwort: Contineous Integration and Delivery
- Technologien entwickeln sich schnell weiter
- Architektur resultiert in ein verteiltes System mit all seinen Vor- und Nachteilen

Services vs. MicroServices

- Bisher:



- Die extreme bzgl. Kopplung von Subsystemen sind nun bekannt. Jetzt:
 - Welche Abstufungen sind dazwischen möglich?
 - Welche Kommunikationsformen und Granularitäten?
 - Welche Einsatzszenarien für welche Architektur?

Kriterien zur Auswahl von Architekturmustern

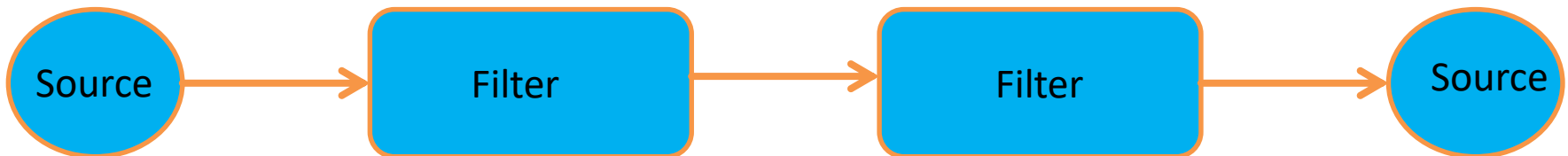
- Welche Struktur des Softwaresystems?
 - Komponenten-orientiert, monolithisch, Schichten, Pipes and Filters
- Wie kommunizieren die Sub-Systeme?
 - Ereignis-basiert, Publish-Subscribe , asynchrone Nachrichten
- Wie sind die Sub-Systeme verteilt?
 - Client-Server, shared nothing, Peer2Peer, service-orientiert, Cloud

Pipes and Filters



Aufbau

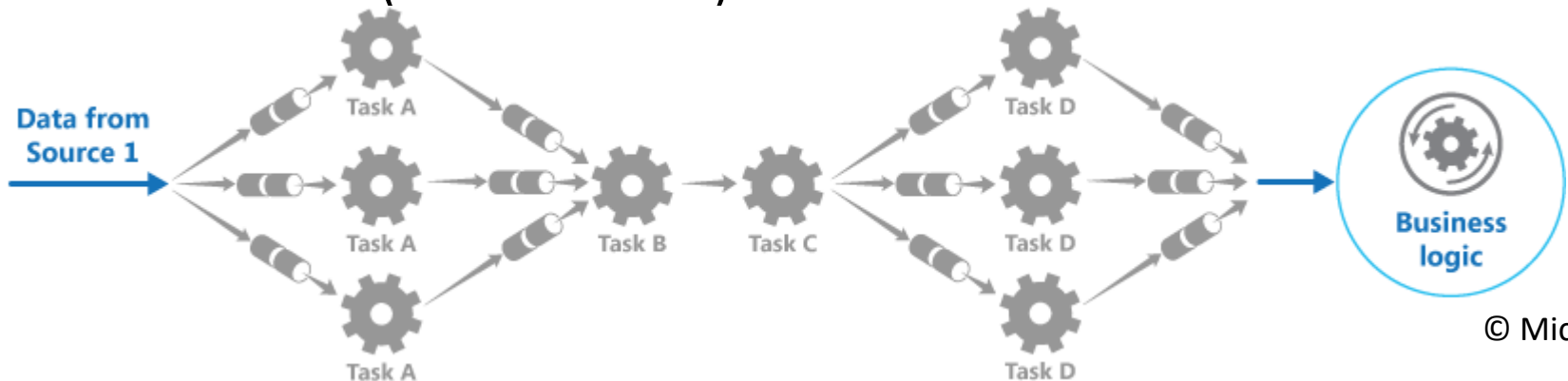
- Pipe: Verbindungsglied, welches Daten von einem Filter zu einem anderen weiterleitet
- Filter: Transformiert Daten, die es durch eine Pipe bekommen hat



- Vorteile:
 - Filter können einfach hinzugefügt und herausgenommen werden
 - Robust, performant, skalierbar, gute Wartbarkeit!

Vor- und Nachteile

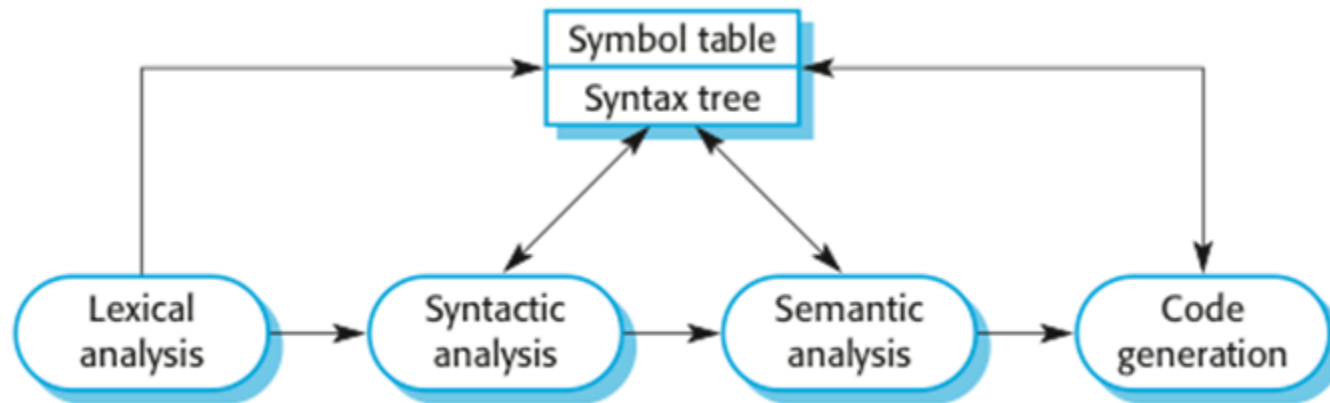
- Parallelisierbar (Lastbalanciert)



- Probleme:
 - Komplexität steigt
 - Bufferoverflow möglich
- Wann anwendbar?
 - Anwendung kann in mehrere unabhängige Teile zerlegt werden
 - Wenn viele Transformationen auf Daten nötig sind
 - Wenn Flexibilität notwendig ist (z.B. in der Cloud)

Beispiele

<i>Domain</i>	<i>Data source</i>	<i>Filter</i>	<i>Data sink</i>
<i>Unix</i>	<code>tar cf - .</code>	<code>gzip -9</code>	<code>rsh picasso dd</code>
<i>CGI</i>	HTML Form	CGI Script	generated HTML page



Was Sie mitgenommen haben sollten:

- Inwiefern wirkt sich die Architektur auf das Softwaresystem aus?
- Wie kann die Auswahl einer geeigneten Architektur den Entwurf / Implementierung vereinfachen?
- Was bedeutet Kopplung und Kohäsion auf Architekturebene?
- Was ist ein Architektur-Stil?
- Für welche Szenarien sind MVC oder Pipes and Filters sinnvoll?
- Was sind Limitierungen von monolithischen Systemen?
- Was sollten Schichten nicht auf Elemente in Schichten darüber Zugriff haben?
- Wann macht der Einsatz von MicroServices Sinn und wann nicht?
- Welche Kriterien für den Einsatz bestimmter Architekturmuster gilt es zu beachten?

Was Sie mitgenommen haben sollten:

- How does software architecture constrain a system?
- How does choosing an architecture simplify design?
- What are coupling and cohesion?
- What is an architectural style?
- For which application scenarios is MVC beneficial? For which is pipes and filters?
- Why shouldn't elements in a software layer "see" the layer above?
- What is the role of programming in model-driven architecture?

Literatur

- Bertrand Meyer, Object-Oriented Software Construction, Prentice Hall, 1997 [Chapter 3, 4]
- *Software Engineering*, I. Sommerville, 7th Edn., 2004.
- *Objects, Components and Frameworks with UML*, D. D'Souza, A. Wills, Addison-Wesley, 1999
- *Pattern-Oriented Software Architecture — A System of Patterns*, F. Buschmann, et al., John Wiley, 1996
- *Software Architecture: Perspectives on an Emerging Discipline*, M. Shaw, D. Garlan, Prentice-Hall, 1996

Literatur

- Service-Oriented Architecture (SOA) vs. Component Based Architecture. Helmut Petritsch (http://petritsch.co.at/download/SOA_vs_component_based.pdf)
- Microservices. James Lewis und Martin Fowler.
<https://martinfowler.com/articles/microservices.html>

