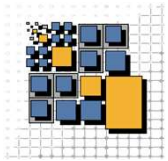
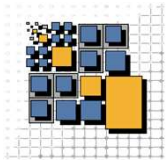


DSG-SOA-M 2024:
- Thinking in Services -

Dr. Andreas Schönberger

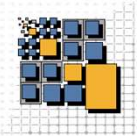
Lehrstuhl für Praktische Informatik
Fakultät WIAI
Otto-Friedrich-Universität Bamberg



- What is a Service? -

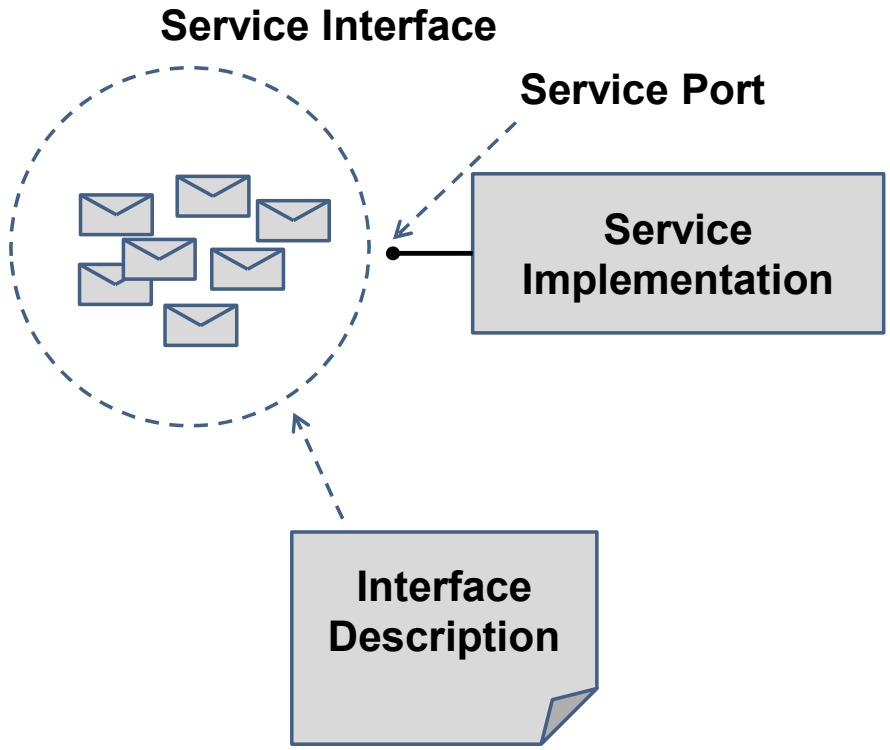
Lehrstuhl für Praktische Informatik
Fakultät WIAI
Otto-Friedrich-Universität Bamberg

Service Definitions I



- **Contemporary** view on (web) services:
  "…services are out-of-process components who communicate with a mechanism such as a web service request, or remote procedure call."
  (https://martinfowler.com/articles/microservices.html)
- **Classic** Web Service definition:
  "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."
  (http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/)

Service Definitions - II



Important "parts" of a service:

Service implementation :=
The actual software component(s) providing the business logic and the remote communication

Service Port :=
The (network) address for accessing the service

Service Interface :=
The set of messages that can be exchanged with a service and the corresponding constraints

Interface Description :=
The potentially machine-processible description of the service interface (optional!)

My personal point of view

Service Definitions - II

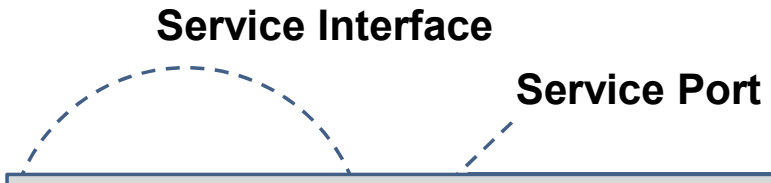
Important “parts” of a service:

Service implementation :=
The actual software component(s)

description of the service interface
(optional!)

My personal point of view

Service Definitions - II



Important “parts” of a service:

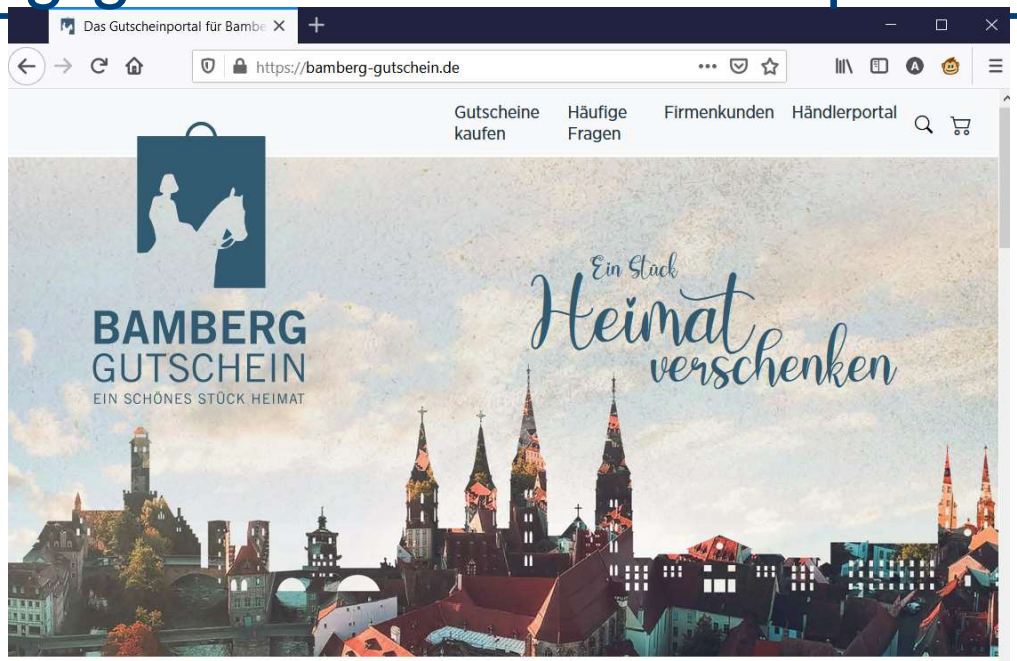
Service implementation :=
The actual software component(s)

Is
Web Services Programming
=
Java Programming?

description of the service interface
(optional!)

My personal point of view

bamberg-gutschein.de as Example

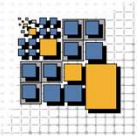


When the Corona Crisis started, Lion5 has set up a cost-free online service for local stores to sell digital coupons.

We will use this real-world example as a reference for discussing important service computing aspects.

I will use #bamberggutschein to mark corresponding content.

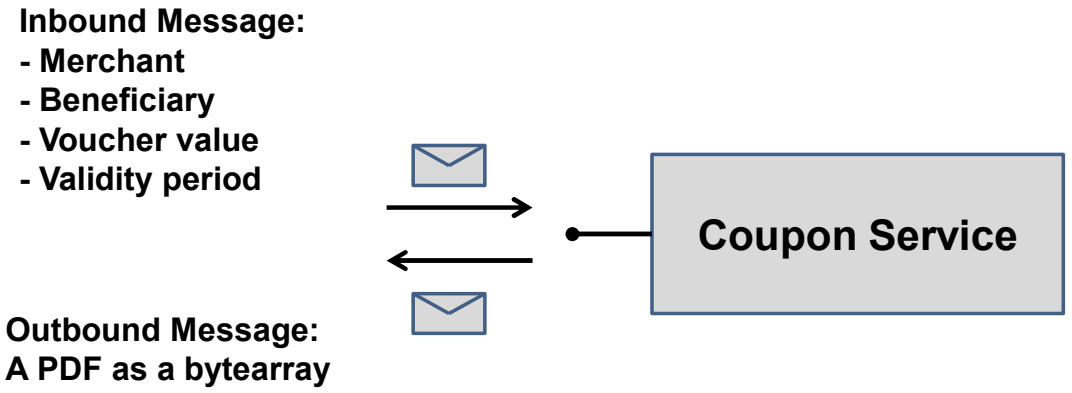
A Coupon Generation Service



“As a supporter of local stores, I want to buy a coupon that entitles me to consume goods and services of a specific merchant worth a defined amount of money.”

#bamberggutschein

Let’s define for now that generating a coupon will be an essential service for realizing this user story



Classic Context Diagram

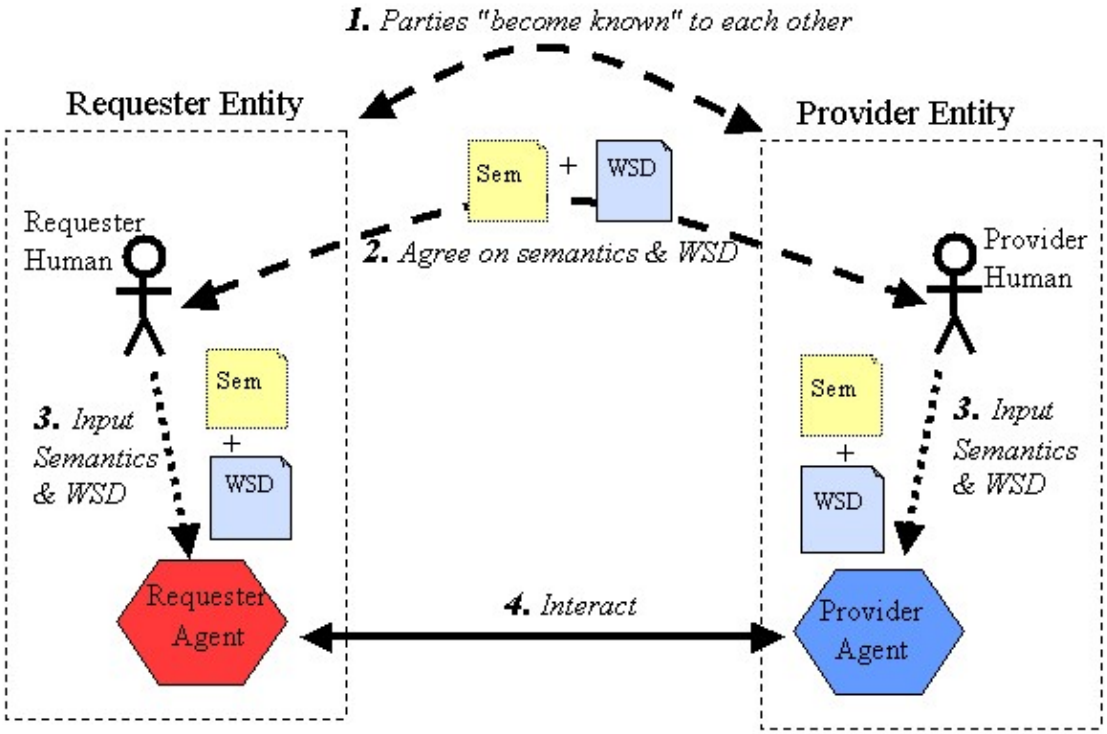
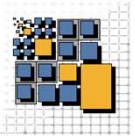


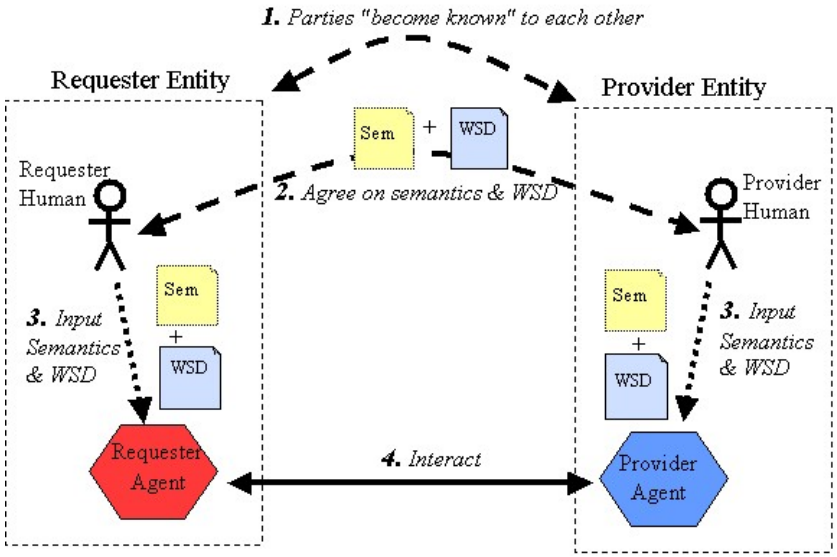
Figure 1-1. The General Process of Engaging a Web Service

http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

Interaction Scenarios



Well supported



http://www.w3.org/TR/ws-arch/

No agreement upon obligatory interaction scenarios

→ Proposal:

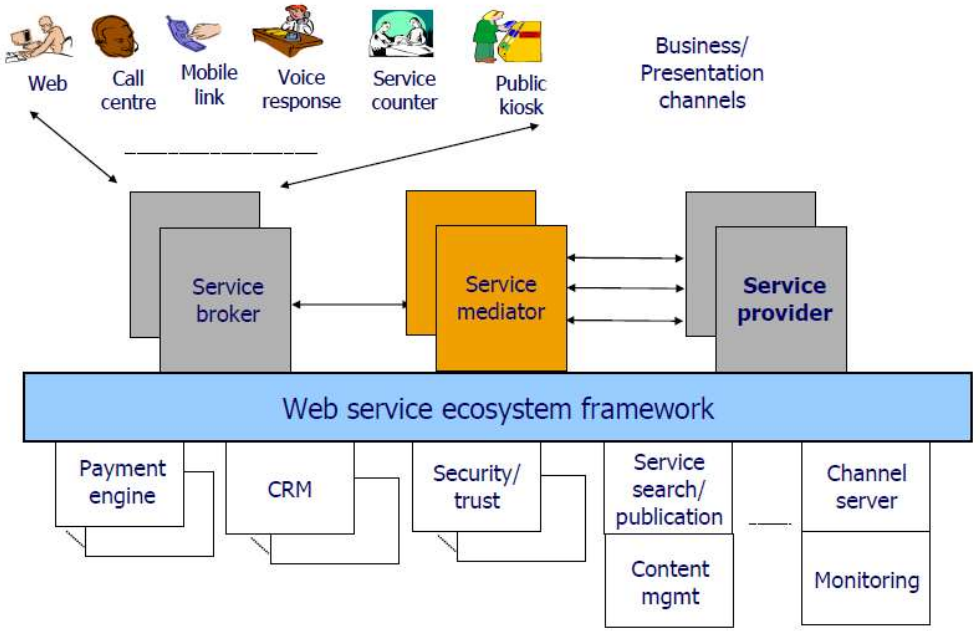
Barros, Dumas, ter Hofstede, *Service interaction patterns*, BPM 2005, Nancy, France

There are many ways for "becoming known" and interacting → just think…

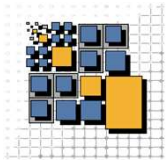
- *Fixed*: Known WSD, Known Semantics, Known Partner, Known URL
- *Relocatable*: Known WSD, Known Semantics, Known Partner, Dynamic URL
- *Pooled*: Known WSD, Known Semantics, Pool of Partners, Dynamic URL
- *Traded*: Known WSD, Known Semantics, Traded Partner (by enhanced registry/broker), Dynamic URL
- *Dynamic-By-Semantics*: Known Semantics, Unknown WSD, Unknown/Traded/Known Partner, Dynamic URL (semantic technologies needed)
- *Dynamic-By-Type*: Known WSD, Unknown Semantics, Unknown/Traded/Known Partner, Dynamic URL (semantic technologies needed)

Advanced Interaction Scenarios

- **Provider**
  implements the actual business functionality;
  offers its service with varying SLAs
- **Broker**
  selects and implements distribution channels
- **Mediator**
  adapts services to different usage contexts such as payment methods, interface adaptation etc.
- **Consumer**
  selects and uses services



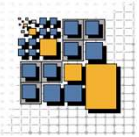
Barros, Dumas, Bruza,
***The Move to Web Service Ecosystems*,**
BPTrends 3(3), 2005



- What is a Services Architecture? -

Lehrstuhl für Praktische Informatik
Fakultät WIAI
Otto-Friedrich-Universität Bamberg

Service-Oriented Architecture (SOA)



- Core idea:

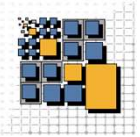
  Think in services, not applications!

- Thinking in applications means:
  - Aligning business tasks with functionalities of software.
  - Trusting in consistent integration of functionalities within software products.
  - Buying a large set of application software because everybody does
    → mail server, file server, web server, content management server…
  - Just rely on a web shop product for selling digital coupons?

#bamberggutschein

- Thinking in services means:
  - Business Alignment (“Business Drives Technology”)
    → Align computing facilities with business tasks. (Make-Or-Buy)
  - Service Value Assessment
    → NO resources without clear value attribution
  - Agile IT Management
    → Replace/Compose/Extend/Modify services as needed.

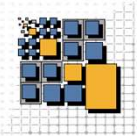
SOA Definitions



- *"Service Oriented Architecture (SOA) is a paradigm for **organizing** and **utilizing distributed capabilities** that may be under the control of different **ownership domains**."*
  (Reference Model for Service Oriented Architecture 1.0, OASIS Standard, 12 October 2006, http://www.oasis-open.org/specs/index.php#soa-rm)

- *„SOA ist ein Architekturmuster, das den Aufbau einer Anwendungslandschaft aus einzelnen fachlichen, das heißt **geschäftsbezogenen Komponenten** beschreibt. Diese sind lose miteinander gekoppelt, indem sie einander ihre **Funktionalität in Form von Services** anbieten. Die Definition eines Service hat den Charakter einer **vertraglichen Übereinkunft** zwischen Serviceanbieter und Servicenutzer. Services werden über einen **einheitlichen** Mechanismus aufgerufen, der die Komponenten **plattformunabhängig** miteinander verbindet und alle technischen Details der Kommunikation verbirgt. Dies erlaubt es, Geschäftsprozesse auf der Basis von Services möglichst einfach zu implementieren und zu ändern."*
  (Regeln für serviceorientierte Architekturen hoher Qualität, Informatik-Spektrum, sd&m Research, 2006)

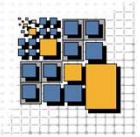
SOA Definitions



*"SOA is a logical way of designing a software system to provide services to either end-user applications or to other services distributed in a network, via **published** and **discoverable interfaces**. A well-constructed, standards-based Service Oriented Architecture can empower a business environment with a flexible infrastructure and processing environment. SOA achieves this by provisioning independent, reusable automated business process and systems functions as services and providing a robust and secure foundation for leveraging these services. Efficiencies in the design, implementation, and operation of SOA-based systems can allow organizations to adapt far more readily to a changing environment."*

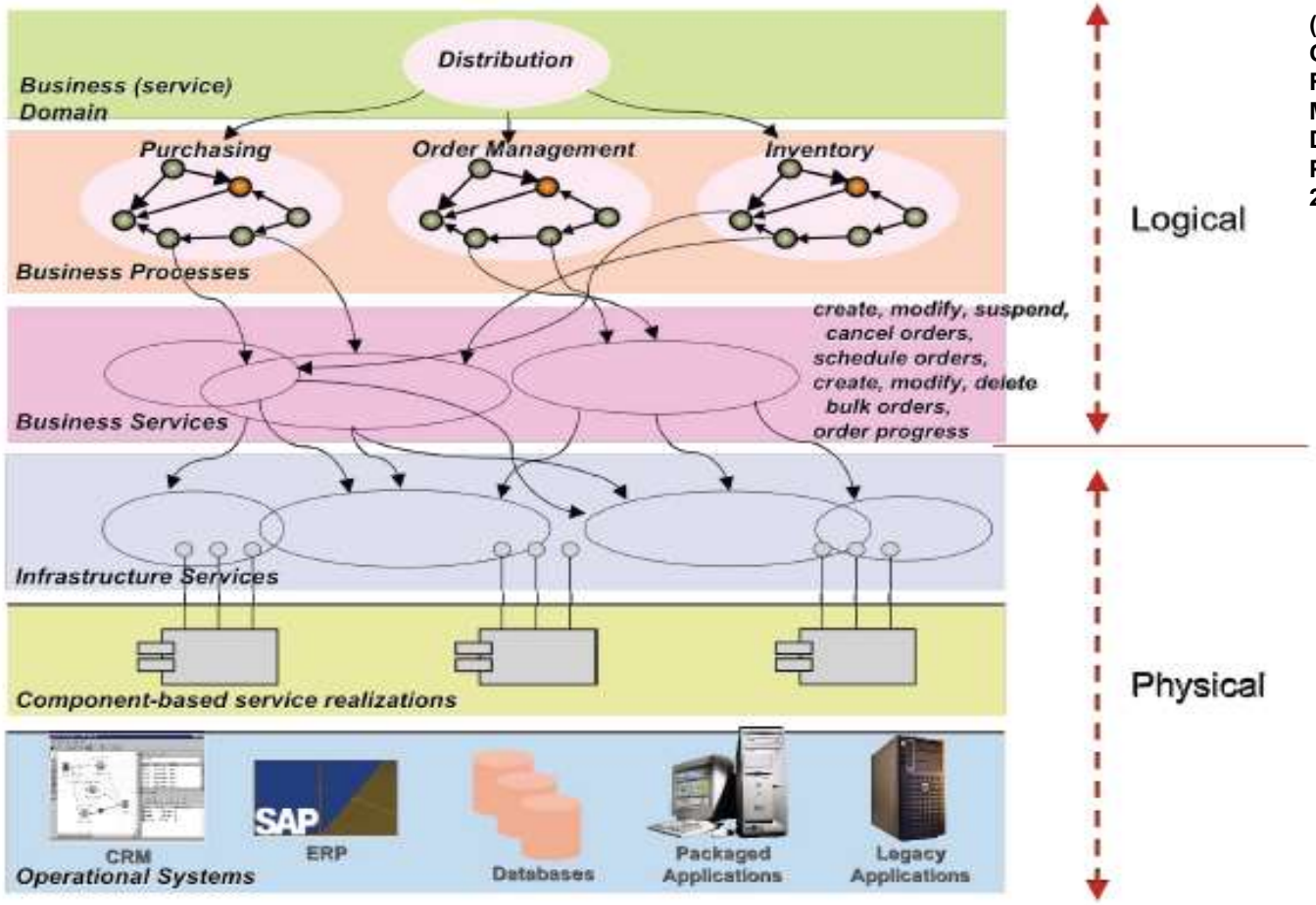
(Service-Oriented Computing Research Roadmap, M. Papazoglou et al., Dagstuhl Seminar Proceedings 05462, 2006)

SOA Implications



- Contract Definition
  → How do you specify the what, how and when of service delivery?
- Service Granularity
  → How can you make sure that your service is useful to more than one consumer?
- Dependency Management
  → If services can be used throughout the enterprise who keeps track of who uses which service in which configuration?
- Risk Management
  → How do you monitor and enforce service contracts?
- Implementation Challenges
  → How do you create a multi-purpose, flexible, robust, efficient … service?
- Common Delivery Platform
  → A common service delivery platform for, say, Siemens AG?
- SOA Governance
  → Who enforces organizational policies for the use of services?

A SOA Proposal …



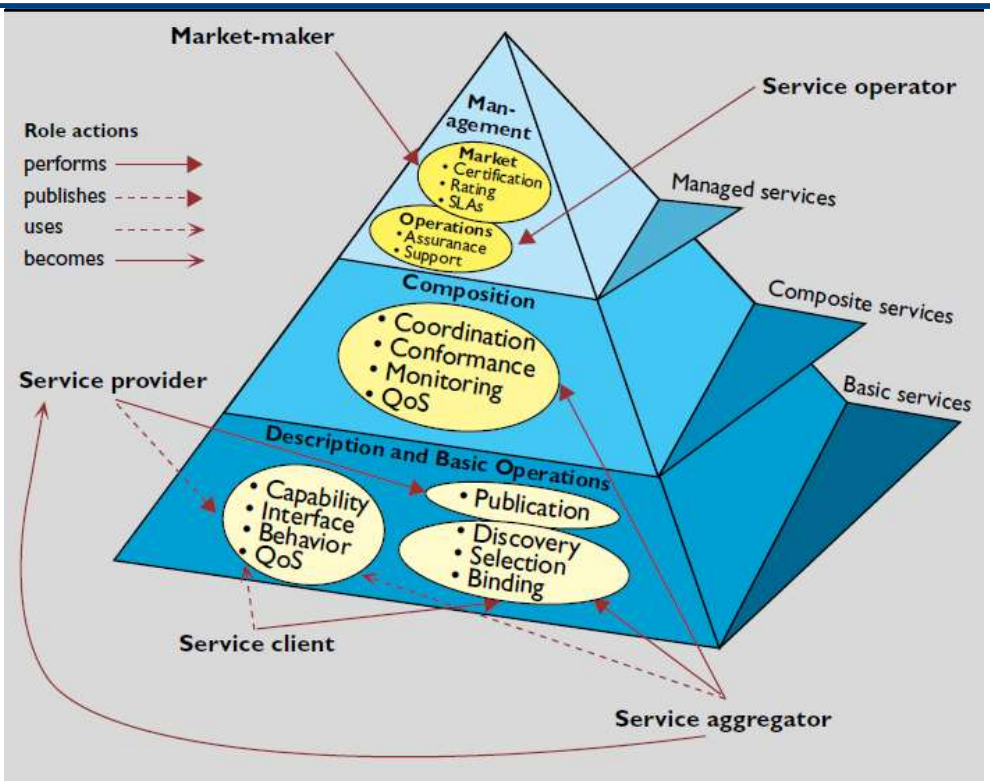
(Service-Oriented Computing Research Roadmap, M. Papazoglou et al., Dagstuhl Seminar Proceedings 05462, 2006)

Figure 4: The Web Services Development Life Cycle hierarchy

… and Another SOA Proposal?

Common Components/ Tasks in SOAs

- Description
- SLA Definition
- Monitoring and adaptation
- Mediation
- Brokerage
- Composition
- Distribution

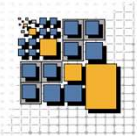


Papazoglou, Georgakopoulos, "Service Oriented Computing", Commun. ACM 46, 10 (October 2003)

Discuss:

- **Relationship between Interaction Scenarios and Architectures**
- **Fiction and Reality**

… and SOA Creation Rules

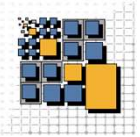


- Regeln für den Komponentenschnitt (Rules on component cut)
  - R1: Fachliche Komponenten (Domain-driven components)
  - R2: Komponenten nach Servicekategorien (One service category per component)
  - R3: Abhängigkeiten gemäß Servicekategorien (Dependencies follow service categories)
  - R4: Keine zyklischen Abhängigkeiten (No cyclic dependencies)
  - R5: Enger Zusammenhalt, geringe Kopplung (High Cohesion, Loose coupling)
  - R6: Datenhoheit (Data encapsulation)
- Regeln für das Design von Services (Rules on service design)
  - R7: Technikneutral (Technology neutral)
  - R8: Referenzfrei (No references)
  - R9: Normal, vollständig und redundanzfrei (Normal, i.e., complete and no redundancies)
  - R10: Grobgranular (Coarse granularity)
  - R11: Idempotent (Idempotent)
  - R12: Kontextfrei (Context free)
- Regeln zur Kopplung (Rules on coupling)
  - R13: Koppelungsmechanismen (Coupling mechanisms follow domain coupling)
  - R14: Transaktionssteuerung (Transaction control follows domain coupling)
  - R15: Datentypen (Data types)

Hess, Humm, Voss
***Regeln für serviceorientierte Architekturen hoher Qualität*,**
Informatik-Spektrum,
sd&m Research, 2006

Rules depend on distinction between 'elementary', 'aggregated', 'orchestratable' services.

… and a Real-World Example



#bamberggutschein

Try to map the concepts of

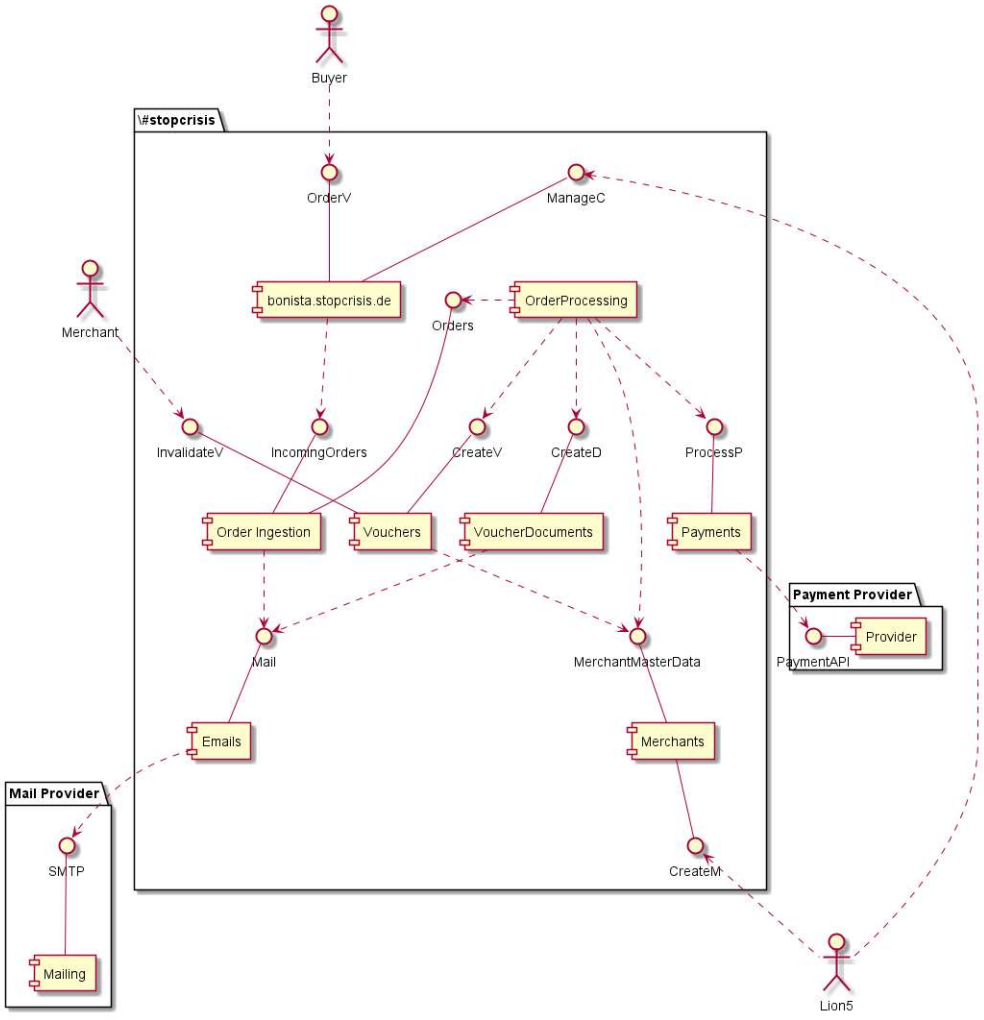
Business Domain,
Business Processes,
Business Services

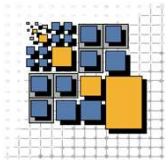
and

Infrastructure Services

to this example!

(cf. slide 17)

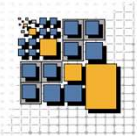




- Service Creation -

Lehrstuhl für Praktische Informatik
Fakultät WIAI
Otto-Friedrich-Universität Bamberg

Service Value First!



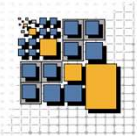
Let's consider the task of registering merchants for a digital coupon service

#bamberggutschein

- Option 1:
  Create a Social Network Style Onboarding Service, i.e., with profile images editor, settings editor etc.
- Option 2:
  Create a Google Forms-based Registration

What would you do?

Service Value First!



#bamberggutschein

Let's consider the task of registering merchants for a digital coupon service

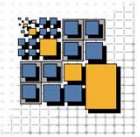
- Option 1:
  Create a Social Network Style Onboarding Service, i.e., with profile images editor, settings editor etc.
- Option 2:
  Create a Google Forms-based Registration

What would you do?

There is nothing more powerful than a running system!

At #bamberggutschein, we sold the first coupon after 4 days.
Guess which option we have chosen!

Service Evolution I



#bamberggutschein

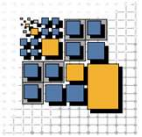
Let's reconsider the coupon service.

So, we started out with:

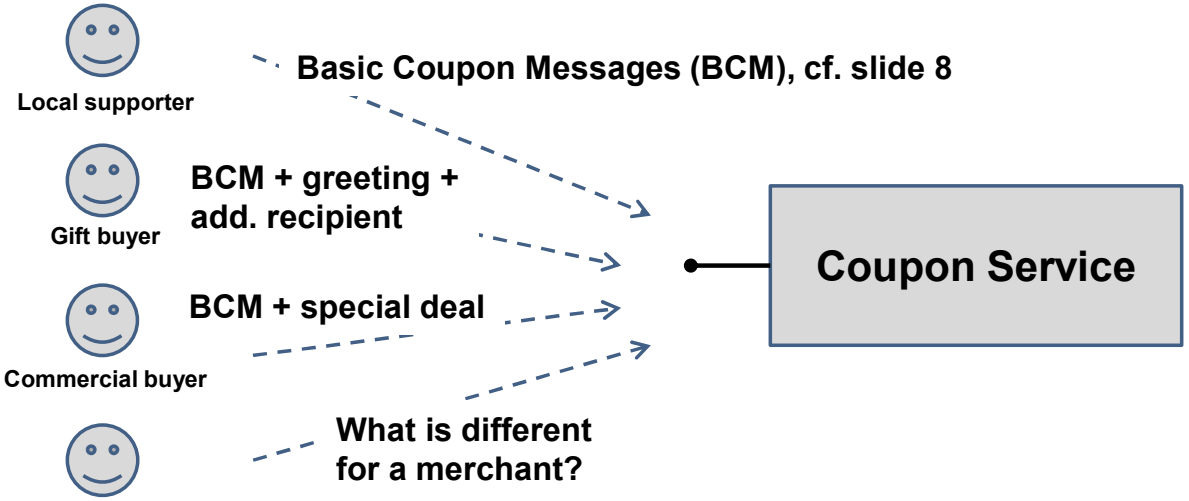
"As a supporter of local stores, I want to buy a coupon that entitles me to consume goods and services of a specific merchant worth a defined amount of money."

- Now, what if
  - the supporter does not want to buy a coupon for himself but rather a gift for somebody else?
  - the merchant directly sells a coupon?
  - the supporter is a company who buys tens of coupons?

Service Evolution II



#bamberggutschein

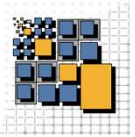


What can you do to evolve your service?

- Option 1: Rewrite the service and tell your existing consumers
- Option 2: Support two different versions of the service
- Option 3: Use Consumer-Driven Contracts
- Something else?

What other kinds of service evolution problems can you think of?

Consumer-Driven Contracts

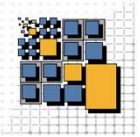


Coupon Service

```
//
if (contains(msg.request,
"greeting")){
   createCouponWithGreeting(...);
} else {
   createCoupon(...);
}
//
```

https://martinfowler.com/articles/consumerDrivenContracts.html

Service Monitoring



Service monitoring is key to the reliability of the system!

#bamberggutschein

Solving an issue requires knowning the issue,
so the most important steps in service monitoring are:

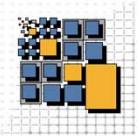
- Define what it means that your system is up and running
  (btw.: this is the hardest part)
  Example:

```
WHEN a local supporter pays the invoice
THEN her coupon shall be generated immediately
```

Is this criterion good enough?

- Add control and observation points
  - control points: interfaces to test-drive functionalites
  - observation points: interfaces to query the state of the system
- Define and implement monitoring cycles, reports and notifications!

Data and …



#bamberggutschein

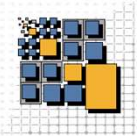
Let’s look at one of those typical project questions

Shall I offer new Payment Methods,
i.e. Apple Pay, Klarna etc.?

Go an discuss with your fellow students and you will be surprised how many opinions you get ;-)

- Option 1
  Look at how many transactions are canceled
- Option 2
  Research market studies on the influence of payment methods on business
- Option 3
  Do an experiment, i.e. check the reaction of users to additional payment options.
- What else can you think of?

Speed



#bamberggutschein

Let's look at the classical Cost vs. Speed discussion:

Online payment providers vary in percentual and fixed transaction costs.
For another project we chose a provider with a low pricing model, but:

- It took 8 weeks from signing the contract to service delivery!
- Two senior developers did not get the basic form-to-invoice generation workflow set up in a complete working day

- So, we evaluated a payment provider that has specialized in online transactions with a slightly higher price
  - On the first day, at 16:30 h, I started the registration form and on the same day basic payment options were ready for production
  - On the second day, later in the evening, form-to-invoice and payment-to-coupon workflows were up and running (done by one developer)
  - I leave the business case calculation as an exercise

Speed is king!