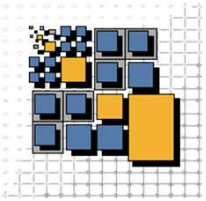
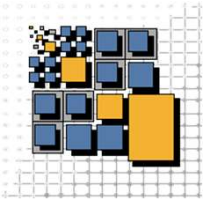


DSG-SOA-M 2024:
- Primer -
- Conceptual Foundations -

Dr. Andreas Schönberger

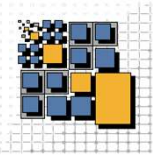
Distributed Systems Group
Faculty Information Systems and Applied Computer Science
University of Bamberg



- Primer -

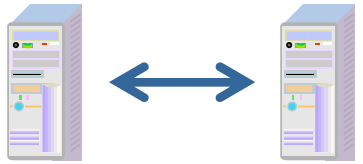
Distributed Systems Group
Faculty Information Systems and Applied Computer Science
University of Bamberg

This course is about composing complex systems from independent subsystems



Which means this course is about

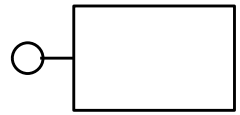
- distributed systems (fundamentals)



- architecting complex systems

- aligning systems with business goals (service-oriented architecture styles)

- integration based on interface technologies



- technologies for running services in a convenient way, container + K8S

About Andreas Schönberger

- Born 1979, lives in Bamberg with his family (two children)
- Information systems studies at University of Bamberg
- PhD in Computer Science at University of Bamberg
- Previously, trainer and software architect for Java EE Solutions, and architecture consultant at Siemens Corporate Technology
- Founder and CEO of Lion5 GmbH in Bamberg
  - Engages in industrial software platform projects
  - Develops cloud-based services and apps
- Supervisor for more than 20 diploma / master theses
- Author and co-author of more than 30 scientific papers
- Open to network
  - https://www.facebook.com/andreas.schonberger.7982/
  - https://www.linkedin.com/in/andreasschoenberger/

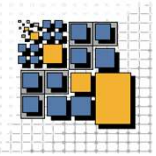
About Stefan Winzinger

- Born 1988, lives in Fürth with his family
- M.Sc. in Computer Science at University of Erlangen-Nuremberg
- Currently pursuing a PhD in Computer Science
- Previously, research staff assistant in Erlangen and software developer at GfK SE
- Research interest: Integration testing of serverless systems
- Supervisor for Bachelor/Master theses
  - Do not hesitate to contact me for thesis topics

Organization

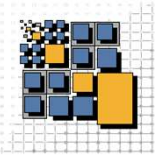
- Who?
  - Lectures 
  - Labs, Assignments (#=2), Tool Introductions 
  - Oral Examination  + 
- Where and When?
  - Lectures: In room WE5/00.019
  - Labs and Tool Introductions: ➔ Watch the course calendar!
  - Assignments: self-organized or in lab
- Support
  - Consulting hours:
    - Andreas Schönberger: get in touch with me via andreas.schoenberger@uni-bamberg.de
    - Stefan Winzinger: https://www.uni-bamberg.de/pi/team/winzinger-stefan/
  - Online: VC course forums, {stefan.winzinger | andreas.schoenberger}@uni-bamberg.de
- More information
  - DSG Homepage: http://www.uni-bamberg.de/pi/
  - VC course: https://vc.uni-bamberg.de/course/view.php?id=67962

Aims of the Course I



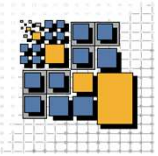
- Aims
  - Understand the characteristics of Microservices and Service-Oriented Architectures (SOA)
  - Know relevant technologies and standards in the field and being able to use those to develop basic Web Services
  - Being able to compare WSDL Web Services to REST Web Services
  - Being able to use container technology for integrating software
  - Being able to judge IT architectures from a Microservices perspective
- How will we do that?
  - Domain overview and knowledge through *lectures*
  - Understanding through *hands-on examples* and *discussions*
  - Implementation capabilities through *assignments*

Aims of the Course II



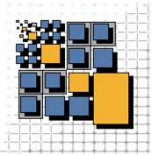
http://www.youtube.com/watch?v=BKorP55Aqvg

Who Asks for that?



- Industry
  - Integrating complex systems is one of the most common and most challenging software engineering tasks
  - Candidates with theoretical background, analytical skills AND hands-on experience wanted
  - Doers wanted, not windbags
  - A lot of DSG graduates have great jobs today at great companies such as Lion5, InnoQ, Senacor, but also Siemens, Bosch, Allianz …
- Academia
  - Research methods frequently call for prototypic implementations
  - Container and Service Architectures are popular research topics
  - Several DSG graduates do their PhD studies now
- Yourself?
  - Be an explorer of the Microservices, Web Services, Container space!
  - Focus on theory and practice possible
  - Good starting point for theses
  - Check out scientific work…find out about interest in PhD studies?
    ➔ Apply for a research oriented thesis

Who Asks for that?



Architecting the new [redacted] Group

Dear Andreas,

I hope this email finds you well.

We are looking for a passionate Technical Architect who can build up our PLM system (product life cycle management) from scratch.

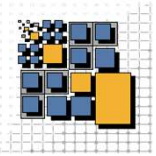
This role is for a hands on architect who could make his hands dirty with code reviews, system design, working closely with the developers and building the entire system from scratch. Your LinkedIn profile makes me strongly believe that you are the kind of professional whom we are looking for.

Interested in learning more? lets catch-up on a quick call and discuss in detail, let me know your telephone number and time of convenience.

Last but not least the job description for your review: https://jobs.[redacted] group.com/s/rci35X I am looking forward to hearing from you.

Best regards,

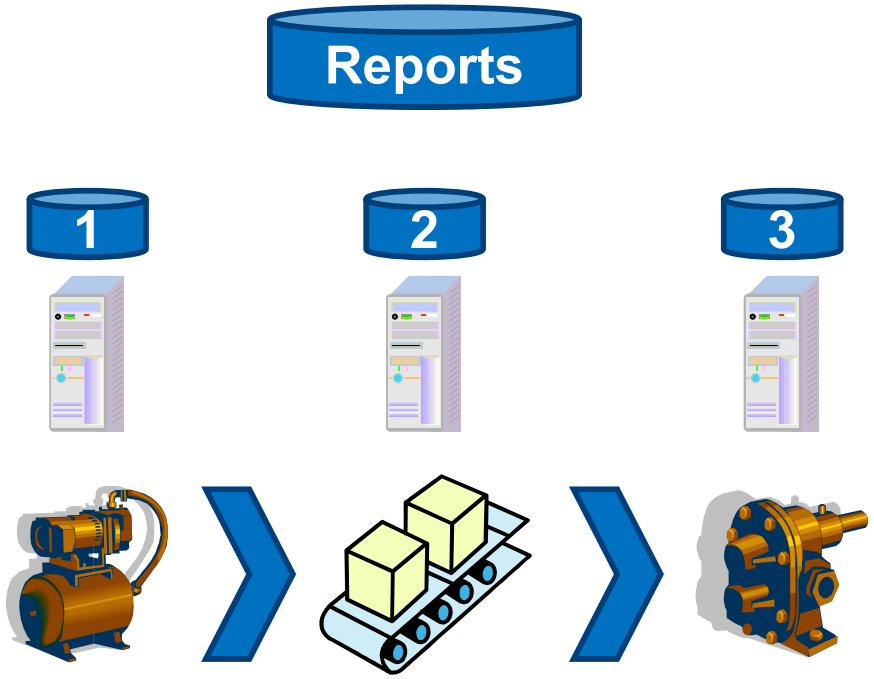
Who Asks for that? – The Production Site Case



The problem:

For a large production site (simplified view) local data storages (1, 2 and 3) had to be updated in a consistent manner and a reports database had to be concurrently written.

Yet, the development team did not implement distributed transactions and mutual exclusion!



The effect:

An expert consultant had to review the control mgt. system, parts of the system had to be reimplemented and the original delivery date was delayed by half a year.

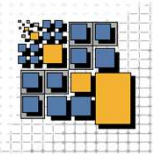
Travel and consulting cost	60	k$
Reimplementation cost	140	k$
Delay penalty	6.5	M$

Your potential role in the game

- System/Software architect
- Expert consultant
- Project manager

This course is at the heart of your future professional life!

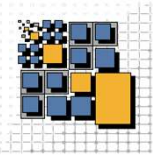
Complementary DSG Offers



- IDistrSys: Introduction to Distributed Systems (summer, 4 hours, 6CP)
- **DSAM-M: Distributed Systems Architecture and Middleware (winter, 4 hours, 6CP)**
- **SRDS-M: Selected Readings in Distributed Systems (summer or winter, 2 hours, 3CP)**
- **Project-M: Master Project in Distributed Systems (summer or winter, 6 hours, 9CP)**
- **Bachelor and Master Theses (any time)**

- Reflects DSG research efforts
  - Distributed programming and middleware frameworks
  - Design, validation and implementation of complex (interacting) systems
  - Workflow and business process modeling languages
  - External validation by means of conference/journal submissions
  - Collaboration with other research groups
  - Participation in international standardization efforts such as *RosettaNet Message Choreography and Control*
  - External PhD students (BASF-IT, RedHat, Siemens Corporate Technology, Siemens HealthCare)
  - External Bachelor and Master Theses, including Siemens Technology

How to Succeed in this Course

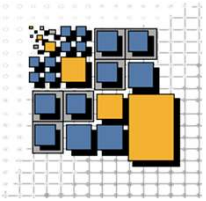


- Computer Science means actively trying out/applying theories, algorithms, modeling and programming languages.
- Practical computer science puts the emphasis on application scenarios, frameworks and IDEs

➔This course is not about learning item lists by heart

- So,…
  - Check presented material against sample scenarios
  - Try out, modify, recompile, test sample code
  - Ask and discuss

  … on a weekly basis!
- You don't have to be an expert in server programming, REST, MQTT …, Linux, but you must be willing to spend some time on it



- Distributed Systems and Middleware -

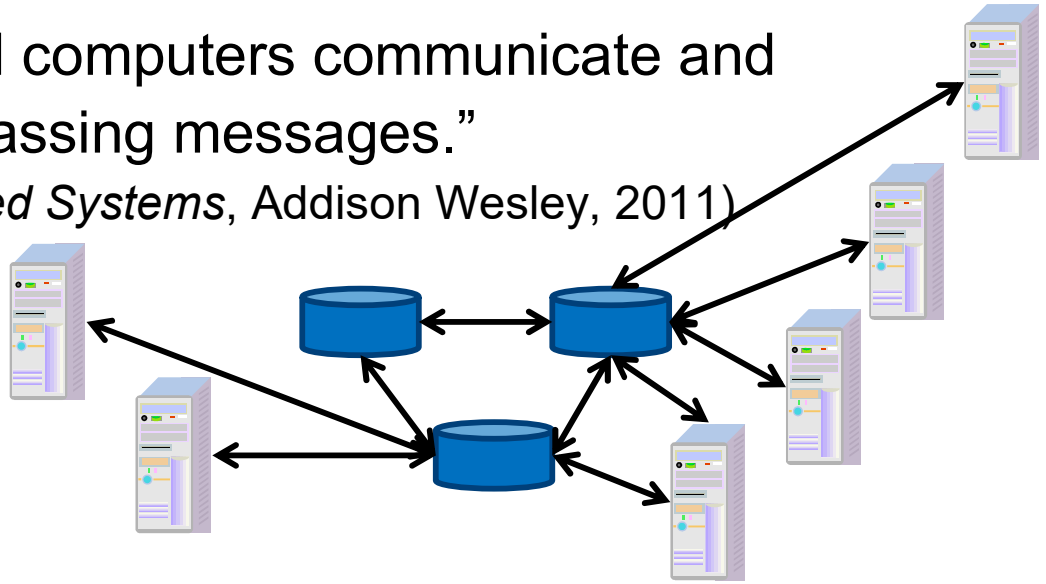
Distributed Systems Group
Faculty Information Systems and Applied Computer Science
University of Bamberg

Distributed Systems

- Definition:

  “A distributed system is a collection of independent computers that appears to its users as a single coherent system.”
  (Tanenbaum, *Distributed Systems*, Prentice Hall, 2017)

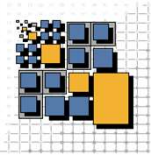
  “We define a distributed system as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.”
  (Coulouris, Dollimore, Kindberg, *Distributed Systems*, Addison Wesley, 2011)



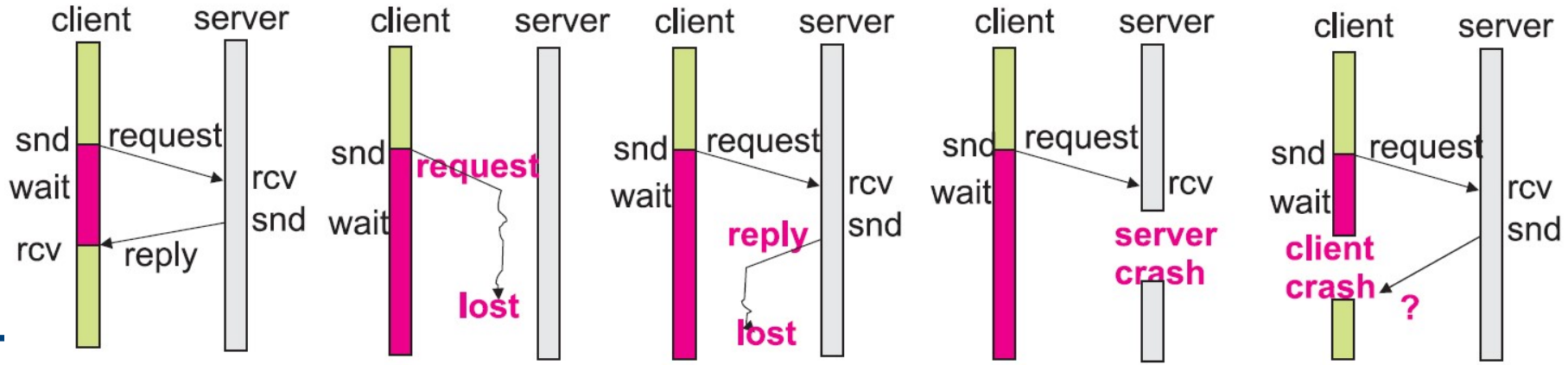
➔ Distributed Computing =

“The task of engineering, developing and running programs on distributed systems”

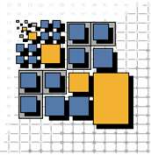
Characteristics of Distributed Systems



- Core characteristics of distributed systems determine the programming paradigm
  - Autonomous Entities / Partial Failures
  - No Global Time
  - No Global Memory
  - Communication Errors
  - Heterogeneity (technical, semantical)
  - Complex Associations (dynamic bindings, multi-party)
- How do those characteristics influence the scenario below?



Excursus: Coordinated Attack Problem



see also IDistrSys and Prof. Mendler's lecture on Communication and Concurrency

Consider this:

- Two allied generals occupy a fortified city
- Both have agreed to attack the city, but they have not agreed upon time
- The attack will only be successful if both generals attack at exactly the same time
- The ONLY way of communication is sending a messenger (NO mobile phones, NO fireworks, NO smoke signal!). However, messengers may be intercepted (and killed).

Can the two generals agree upon a time for attack such that each of them can be sure about the other general's participation? (The generals and messengers do not lie!)

Excursus: Leader Election

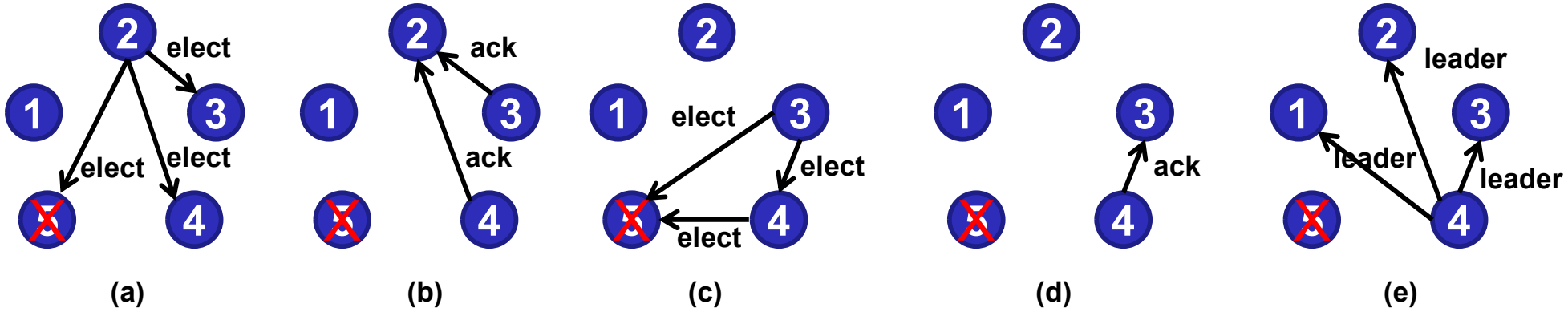
Problem:
How to elect a “leader” in a set of processes?

see also IDistrSys and Prof. Mendler’s lecture on Communication and Concurrency

Classic solution (“Bully algorithm”)

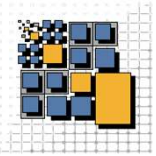
- Impose an order on the set of all processes
- If leader is not responding, ask higher level processes
  If there are no replies you are the leader, then announce it
- If asked by lower level processes, acknowledge election

more advanced algorithms for different system contexts (e.g. more failures) and large-scale systems available



But wait, what is a leader good for?

Architectural styles of distributed systems

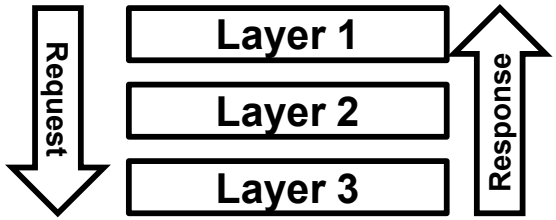


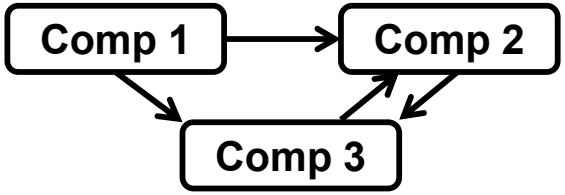
Architectural styles give you guidance on how to structure your distributed system

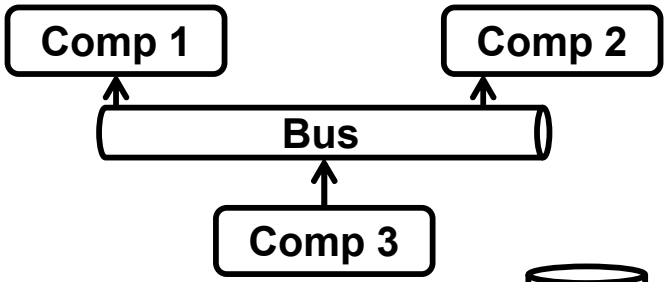
- Layered architectures
  Order your components in layers where calls go from higher-level layers to lower-level layers.
- Distributed objects
  Expose your components via an endpoint to remote calls
- Event-driven
  Decouple your components via events that are distributed via an event bus (publish / subscribe)
- Data-centric
  Use data junks or files as means for coordination, e.g., web-based systems

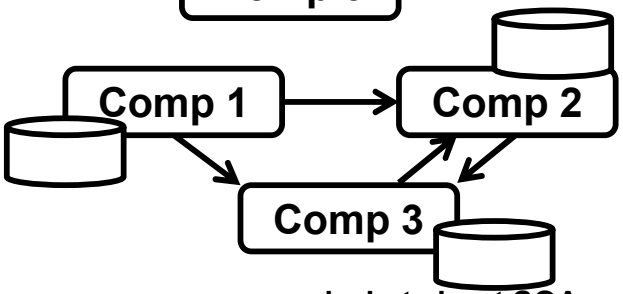
→ also think about hybrid styles: YOU choose

cf. Tanenbaum and van Steen: Distributed Systems, Prentice Hall!



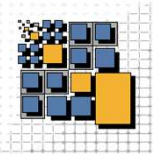






and what about SOA and Microservices?

Use Distributed Systems in spite of Complexity?



- You have to, the potential is too big!
  - → Flexibility, Robustness, Availability, Cost Savings…
  - Connectivity for actors of all sizes, data centers and mobile devices
  - Integrate existing infrastructure and applications (EAI)
  - Implement business processes across different company locations (EAI)
  - Implement business processes across company boundaries (B2Bi)
  - Only option for really dependable systems

➔Almost every system is a distributed system!

BEWARE: There's no such thing like a free lunch!

Really … no free lunch!

heise newsticker, 04/15/2013:

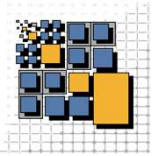
Vaillant-Heizungen mit Sicherheits-Leck

“Die Vaillant-Heizungsanlagen des Typs ecoPower 1.0 enthalten ein hochkritisches Sicherheitsloch, durch das **ein Angreifer die Anlage über das Internet ausschalten und potenziell beschädigen kann**. In einem Informationsschreiben rät der Hersteller seinen **Kunden** daher zu einem drastischen Schritt: Sie **sollen den Netzwerkstecker ziehen und auf den** Besuch eines **Servicetechnikers warten**.

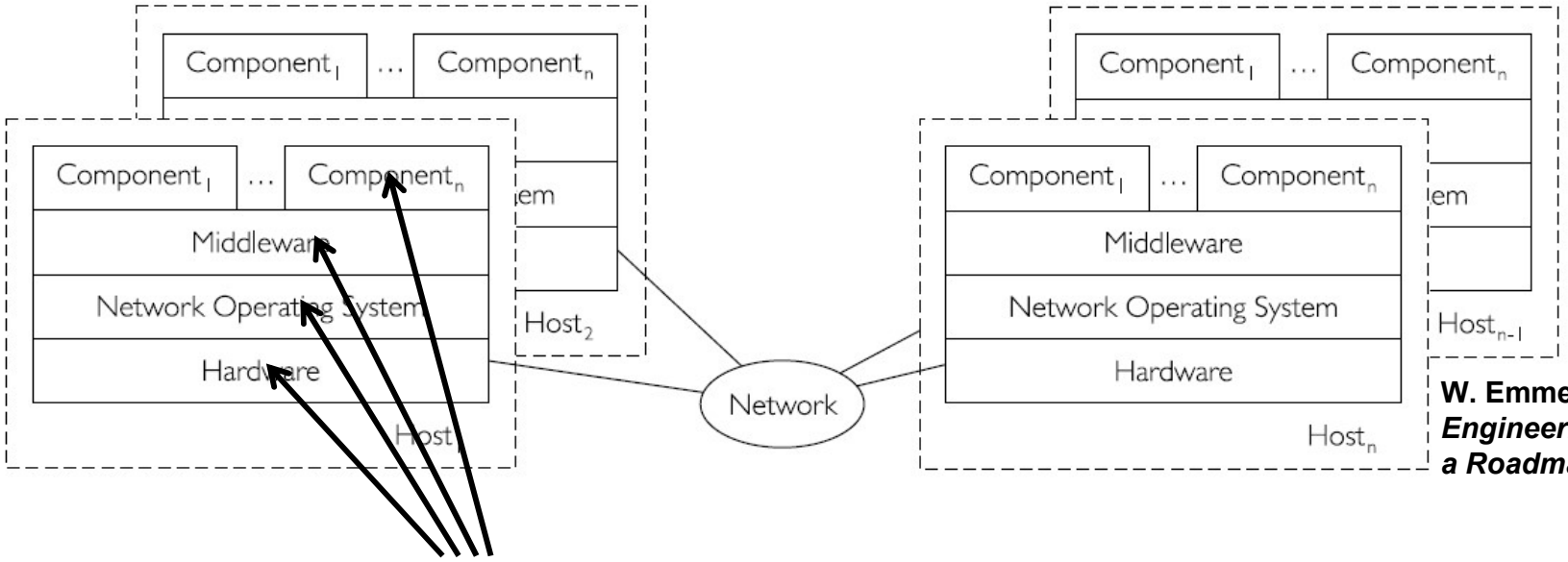
Bei den **für Ein- und Zweifamilienhäuser** ausgelegten ecoPower-Anlagen…

… **Neben dem Kundenpasswort** geben die Anlagen auf Zuruf **auch das sogenannte Experten- und das Entwicklerpasswort aus** …
… Die Situation wird dadurch verschlimmert, dass sich die ecoPower-Anlagen bei einem **DynDNS-Dienst des Herstellers anmelden**. Man kann **daher durch simples Herumprobieren sämtliche laufende Systeme aufspüren**.“

Middleware to the Rescue



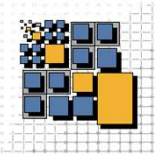
- Middleware is the classical means to manage the complexity of distributed systems, but what is middleware?
- Middleware is the software between
  - …distributed system and operating system (local view)
  - …service user and service provider (global view)



W. Emmerich: *Software Engineering and Middleware – a Roadmap*, ICSE 2000, Limerick

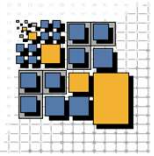
Classical Layering Strategy

Typical Middleware Services



- Middleware differs in which of the below services are offered
  - Naming (Local vs. Remote References; Reference Injection)
  - Transactions
  - Persistency
  - Security
  - Lifecycle Management
  - Scalability
  - Replication (Consistency vs. Availability)
  - Interoperability
  - Vertical/Horizontal Clustering
- Middleware differs in how services are offered
  - Explicit Use
  - Transparent Use

Sample Discussion: Clustering



- Aims
  - Failover
  - Load-Balancing
  - Resource Usage
  - Response Time
  - Scalability
- Implementation Options
  - Vertical: Multiple services on the same machine
  - Horizontal: Multiple services on multiple machines

Discuss: Which option satisfies which aim?

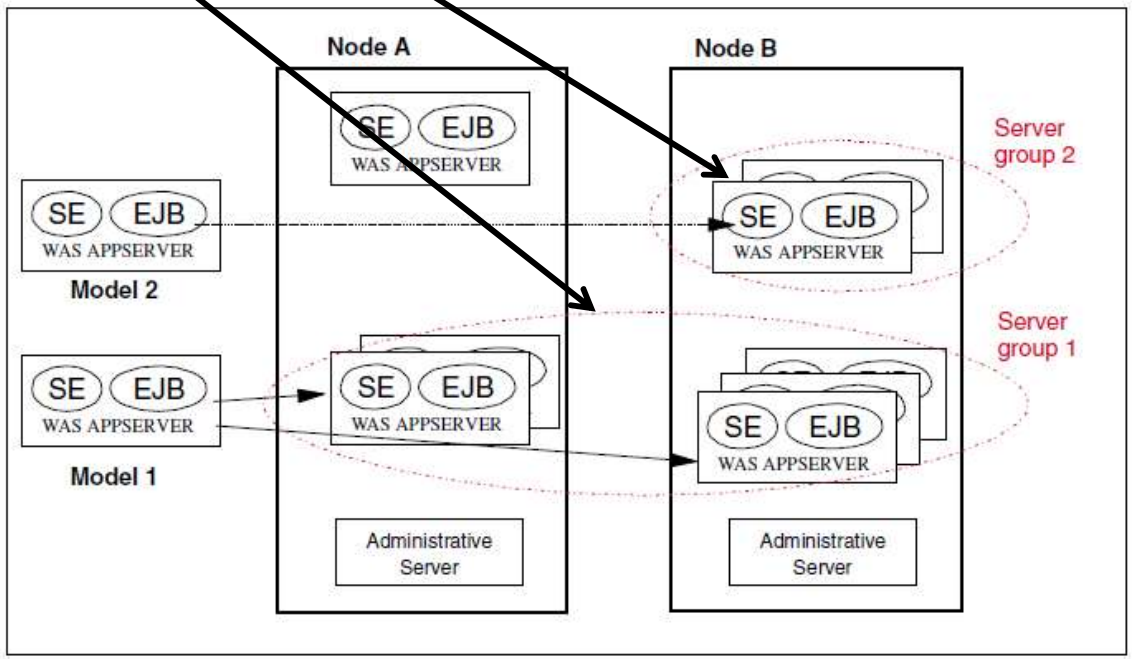
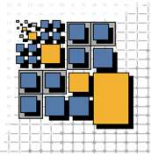


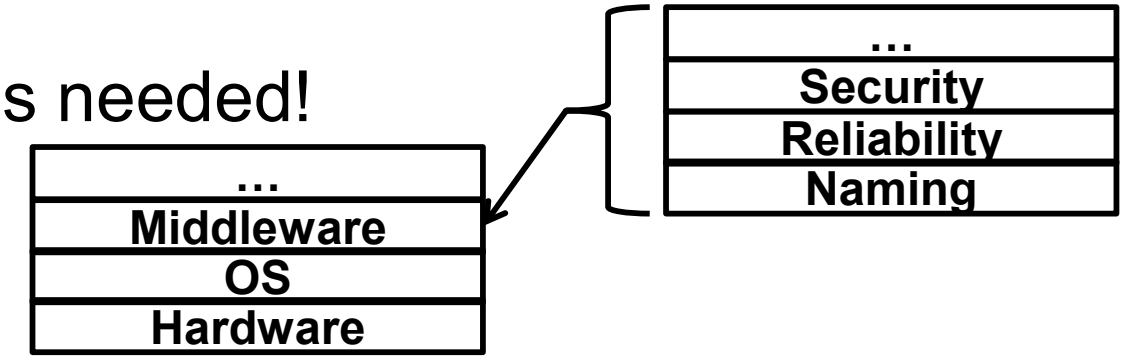
Figure 3. Models and clones

IBM Redbooks:
WebSphere Scalability: WLM and Clustering

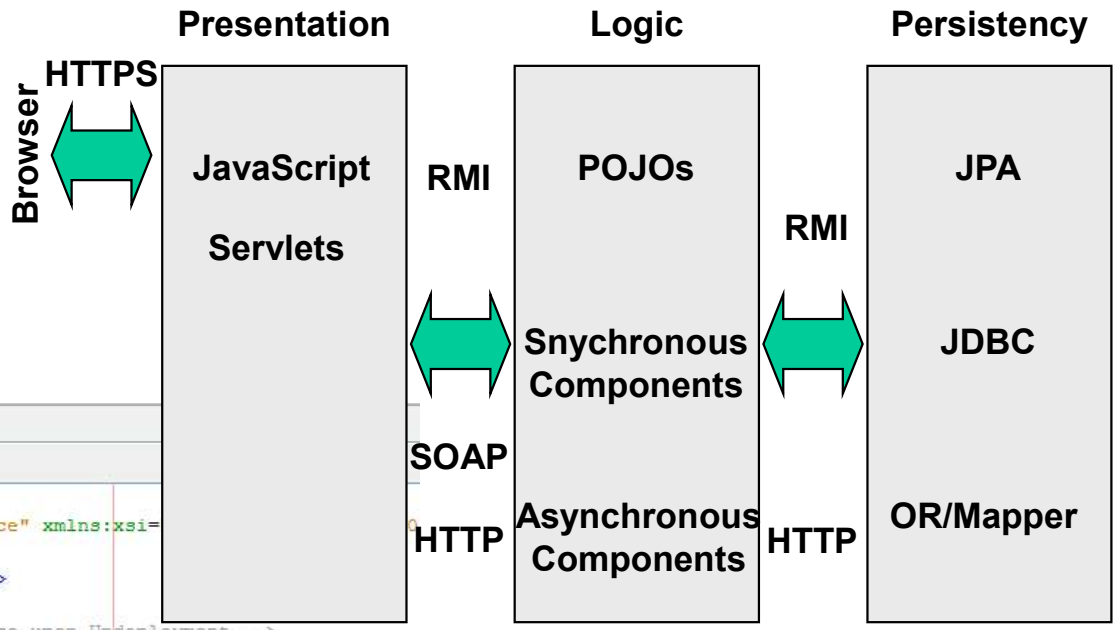
How to Provide Middleware Services I



- Layering
  - ➔ Precise assumptions needed!



- N-Tier-Architectures and Containers

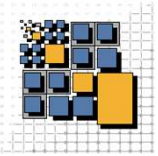


- Deployment Descriptors

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi=
        <persistence-unit name="ejb3DemoEJB">
                <jta-data-source>jdbc/ejb3DemoRoomMgt</jta-data-source>
        <properties>
                <!-- Creates relations upon deployment and deletes these upon Undeployment -->
                <property name="toplink.ddl-generation" value="drop-and-create-tables"/>
                <property name="toplink.logging.logger" value="DefaultLogger"/>
                <property name="toplink.logging.level" value="FINE"/>
                <property name="toplink.logging.thread" value="true"/>
                <property name="toplink.logging.exceptions" value="true"/>
                <property name="toplink.create-ddl-jdbc-file-name" value="create_ejb3Demo.jdbc"/>
            <property name="toplink.drop-ddl-jdbc-file-name" value="drop_ejb3Demo.jdbc"/>
        </properties>
        </persistence-unit>
</persistence>
```

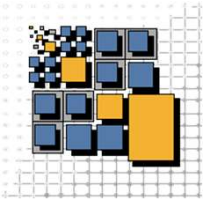
Typical Middleware Technologies



Middleware technologies reflect different system requirements!

- Implement Web Services by means of Server Programming Technologies
  - Servlets
  - Spring Boot
  - .net core
  - …
- Alternative Technologies
  - Message Queueing (WebSphere MQ, MSMQ,…)
  - Transaction Processing Systems (IBM CICS, Oracle Tuxedo,…)
  - Bus technologies, most notably CORBA
  - Domain specific technologies, EDIINT, AS2...

For more information: DSG-DSAM-M!



- Service-Oriented Computing as Distributed Computing Discipline -

Distributed Systems Group
Faculty Information Systems and Applied Computer Science
University of Bamberg

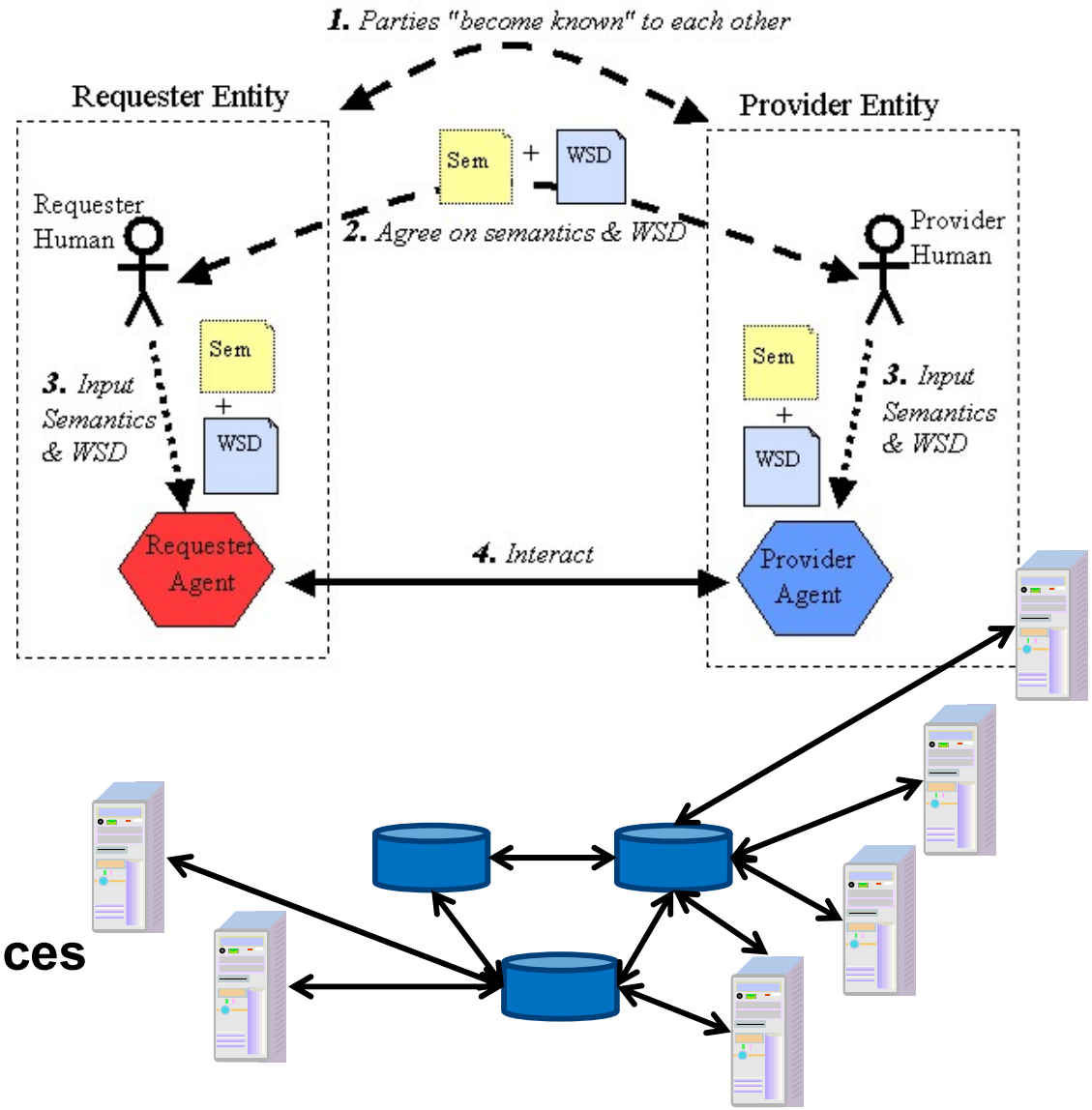
SOC as Distributed Computing Discipline

- SOC = Service Oriented Computing
- The basic service interaction style implies an underlying distributed system.

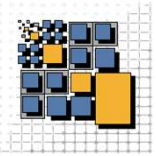
➔Service interactions are subject to all typical distributed computing problems which are driven by distributed system characteristics.

NOTE: SOC != SOA
SOC → Computing based on services
SOA → Architectural paradigm

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



SOC Doubles the Fun!

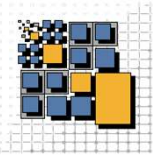


- Building secure/reliable/available… distributed systems is hard!
  → you have to know
  - Business logic
  - Server frameworks
  - Middleware configuration
- SOC is even harder!
  → you have to know
  - Business Logic
  - Server frameworks
  - Middleware configuration
  - Service Wrappers/Exposition of logic as services
  - Service Composition
- And all that in the face of distributed computing tasks
  - Data conversion
  - Distributed commit
  - Distributed transactions
  - …

Create applications and provide middleware services using a dedicated middleware technology!

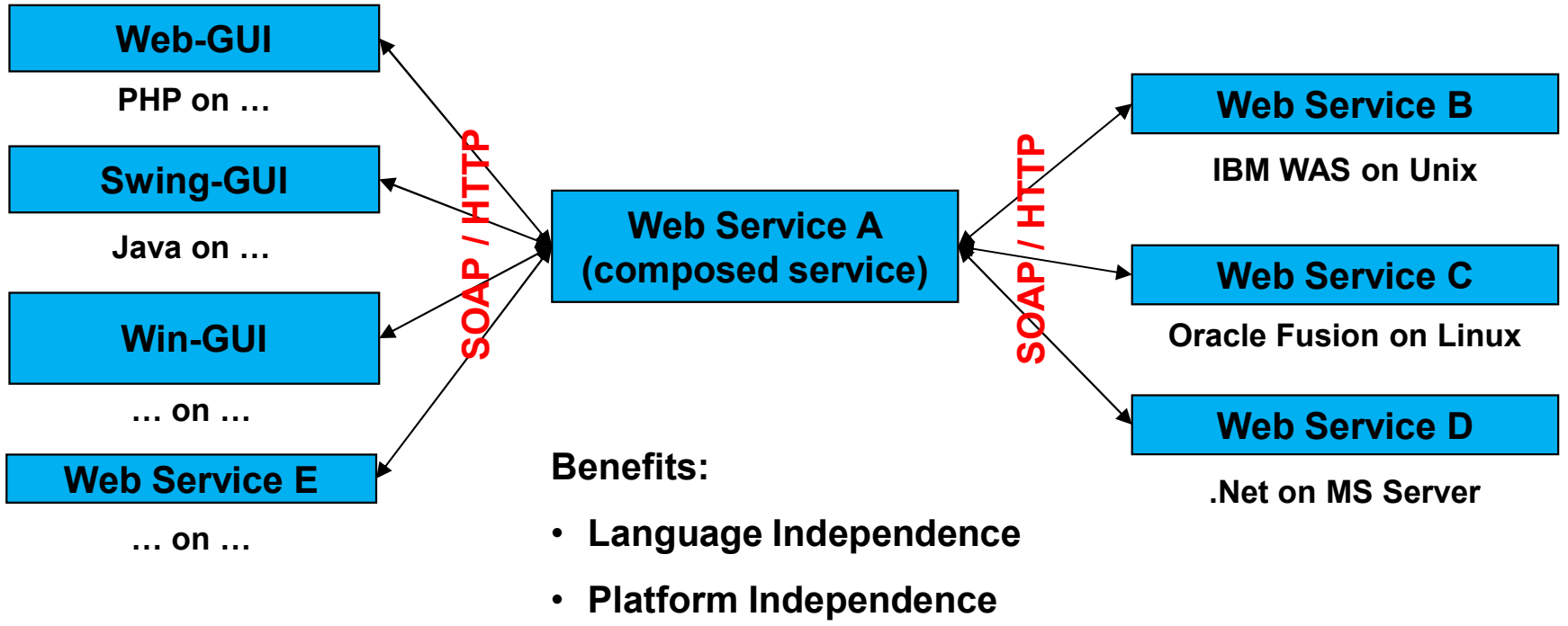
Reuse applications or build new applications from existing applications no matter what the underlying technology is!

Interoperability and Vendor Independence

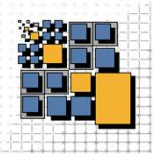


- SOC is not so much about really new middleware concepts
  → it is more about the interoperable, decoupled access to/composition of services built on various technologies

➔ Interoperability is THE KEY PROMISE of SOC



Where to Apply Interoperability



→Be clear for which LAYER you are demanding interoperability!

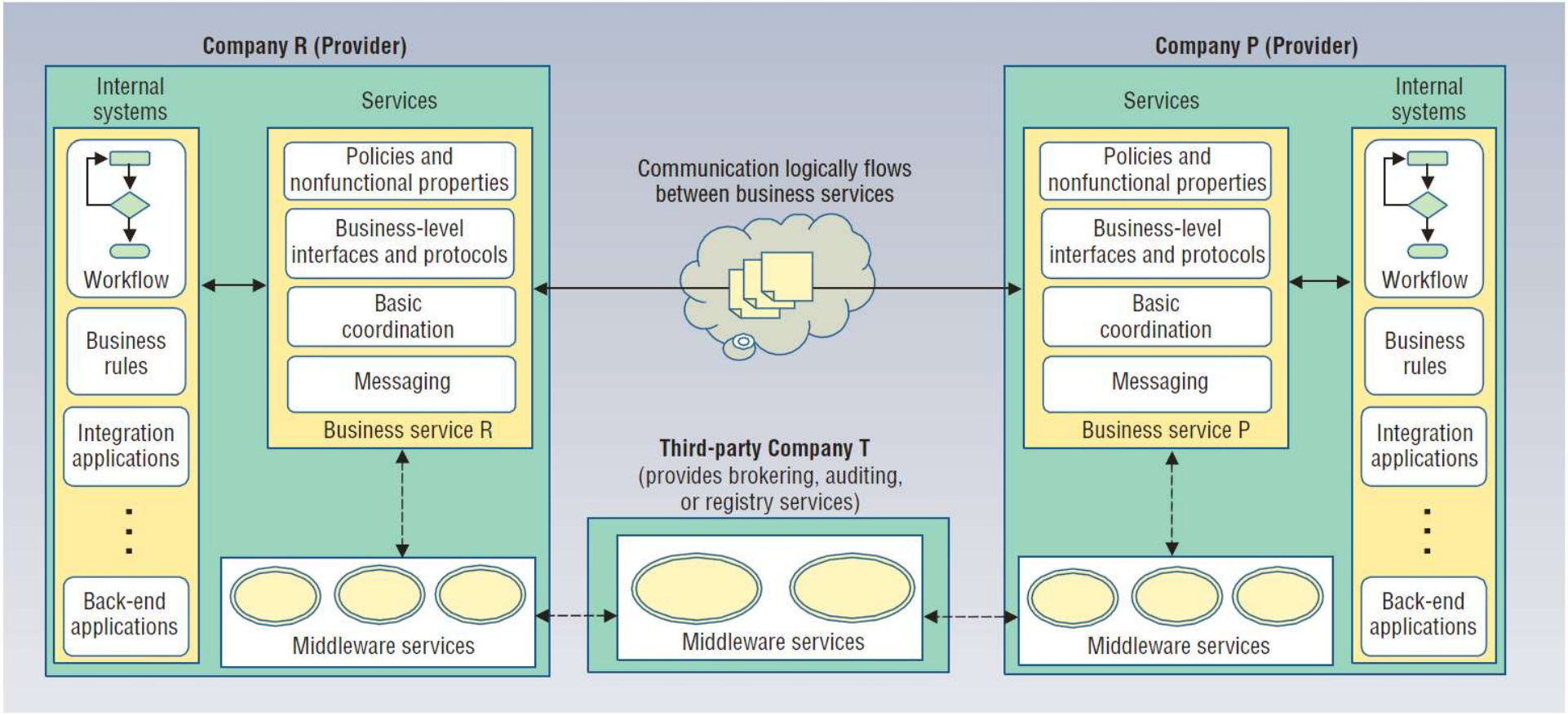
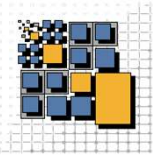


Figure 1. Service integration layers. The lower levels incorporate specifications that most interactions require, while the need for specifications in the higher levels varies depending on the application.

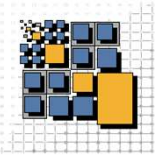
Nezhad et al., *"Web services Interoperability Specifications"*, IEEE Computer, May 2006

Is Interoperability Actually a 'Good' Thing?



- (Potential) Costs of interoperability (for whom?)
  - Limited Flexibility – When are which features available?
    JFK says:
    "Conformity is the jailer of freedom and the enemy of growth!"
    → conformity contradicts competitive advantage
  - Overhead
    - Performance Impact
    - Cost
- (Potential) Benefits of interoperability
  - Cost gain per new connection
  - Improved flexibility – Easily connect to/replace partners!
  - Error-proof technologies
  - Vendor independence

Identify Interoperability Threats to your APP

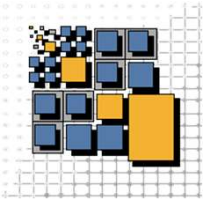


- Threats caused by users/developers
  - Performance optimizations
  - Special use cases
  - Bad code structure
  - Never read the specification
- Threats caused by standardization efforts
  - Optional features, MAY and SHOULD
  - Vendor participation in committees
  - No realistic validation use cases
  - No reference implementation

http://www.ietf.org/rfc/rfc2119.txt

Is Web Services technology known to be interoperable?

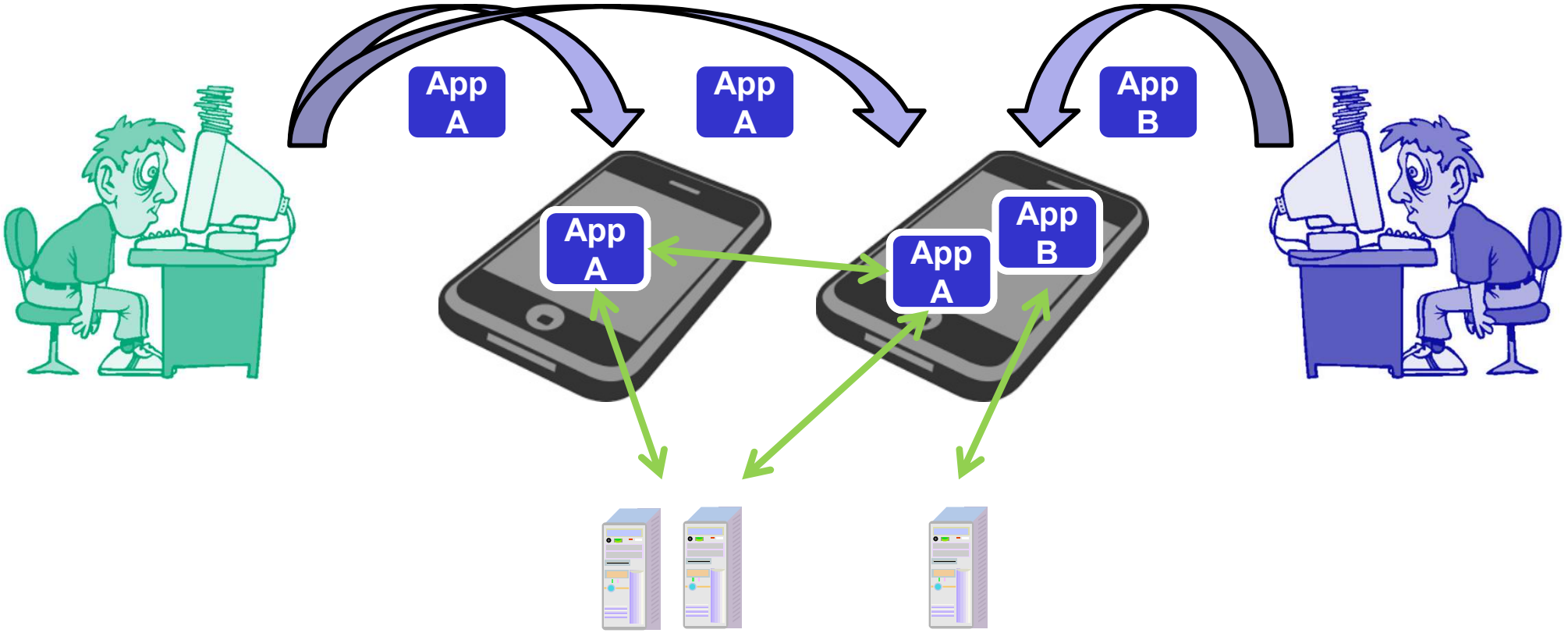
→ Schönberger A., Schwalb J., Wirtz G.: *Has WS-I's Work Resulted in WS-* Interoperability?*
Proceedings of the 9th International Conference on Web Services (ICWS), Washington DC, USA, July 4-9, 2011



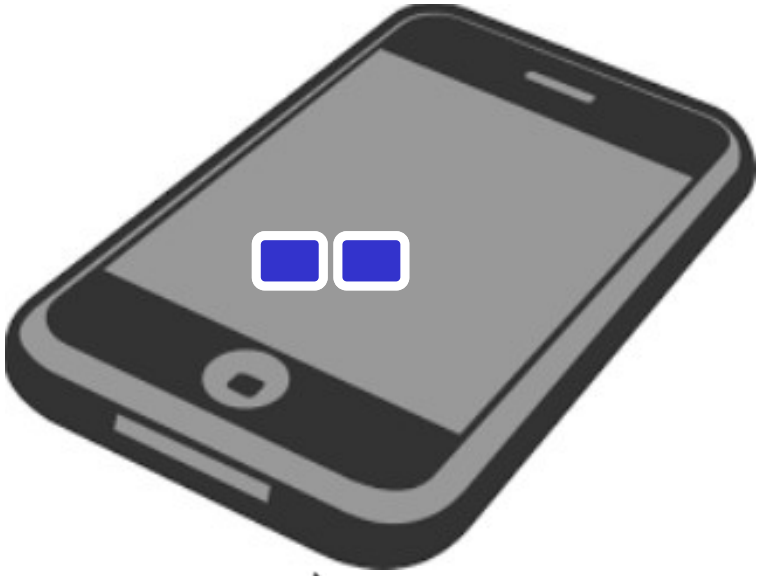
- Software Ecosystems and Services -

Distributed Systems Group
Faculty Information Systems and Applied Computer Science
University of Bamberg

Software Ecosystems and Services

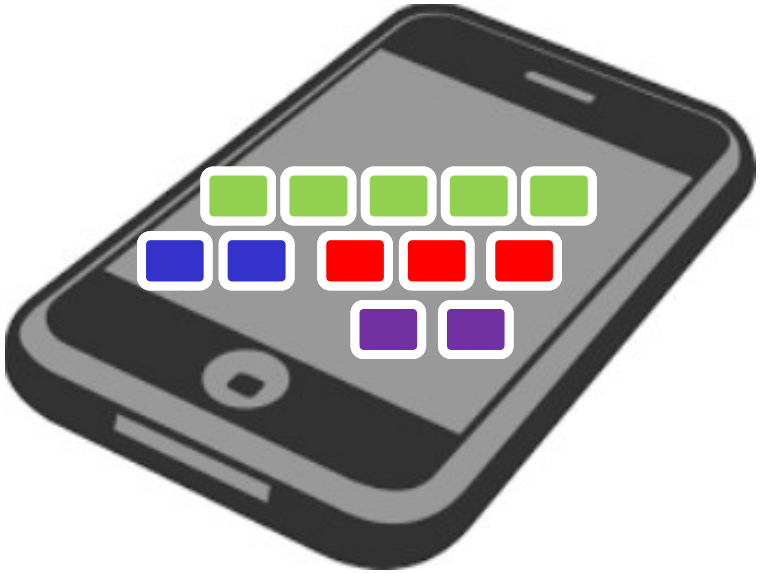


SECOs take you to new borders



Classical Approach

Nokia / Microsoft



SECO Approach

Apple / Google

TeLESM
Three Layer Ecosystem Strategy Model

Internal	Collaborative	External
	Innovation Ecosystem	
• Me-myself-and-I-strategy • Be-my-friend-strategy	• Customer Co-Creation strategy • Supplier Co-Creation Strategy • Peer Co-Creation Strategy • Expert Co-Creation Strategy	• Copy-Cat Strategy • Cherry-Picking Strategy • Orchestration Strategy • Supplier Strategy • Preferred Partner Strategy
	Differentiating Ecosystem	
• Increase Control Strategy • Incremental Change Strategy • Radical Change Stragey		
	Commoditizing Ecosystem	
• Rationalized in-sourcing • Push-out Strategy	• OSS Creation Strategy • Partnership Strategy • OEM Partnerships	• COTS Adoption Strategy • OSS Integration Strategy • Outsourcing

Google play

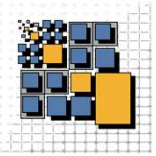
CODESYS

open handset alliance

Src.: Adapted from janbosch.com

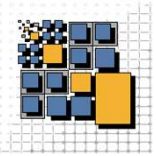
LION5

SECOs take you to new challenges



- Business challenges
  What is your contribution to the overall system and how does it pay off?
- Technical challenges
  For example, how do you provide suppliers with an abstraction of the runtime environment?
- Quality engineering
  For example, how do you avoid unwanted side effects of foreign software?
- Testing challenges
  For example, how do you test against performance requirements if you only have part of the system available?
- Guidance and Governance
  For example, how do you guide developers in using the platform correctly?
- Managing the network
  For example, how do you check the healthiness of your ecosystem?

What is next?



- Concurrency & Synchronization
- Software Architecture
- Web Services
- Describing Services
- REST
- Container Technology & K8S
- Microservices