GI-Fachgruppe Architekturen

ARCHITECTURAL REFACTORING FOR CLOUD

Jahrestagung Architekturen 2014

Prof. Dr. Olaf Zimmermann
Distinguished (Chief/Lead) IT Architect, The Open Group
Institute für Software, HSR FHO
Ladenburg, 8. Juli 2014

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

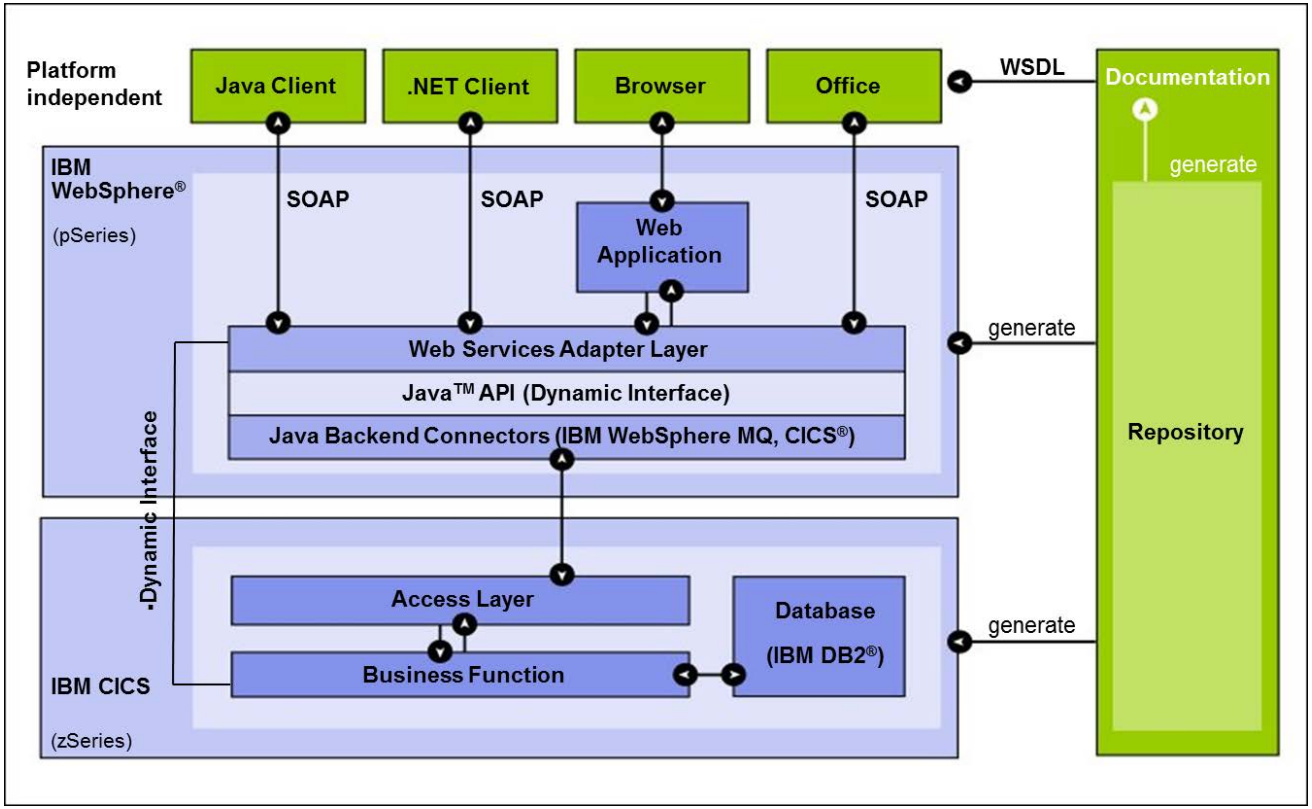
Agenda

- **Kontext & Motivation**
  - OSSM Definition for Cloud Computing, Sample Architecture
- **Cloud Computing Patterns**
  - Cloud Offerings, Cloud Application Architecture, Cloud Management
- **IDEAL Cloud Application Architectures**

- **Decision-Centric Architectural Refactoring (Vision)**
- **Cloud Refactorings – Examples and Catalog Structure**
- **Tool Support (Preview)**

Gedankenexperiment: Ist dieses verteilte Informationssystem cloudfähig?

- **Core Banking Anwendung, Shared Service/Service Provider Modell**
  - Layers Pattern, Datenhaltung im Backend, Web Frontend, Web Services



Referenz: ACM OOPSLA 2004 & Informatik-Spektrum Heft 2/2004

Simple, User-Centered Definition of Cloud Computing

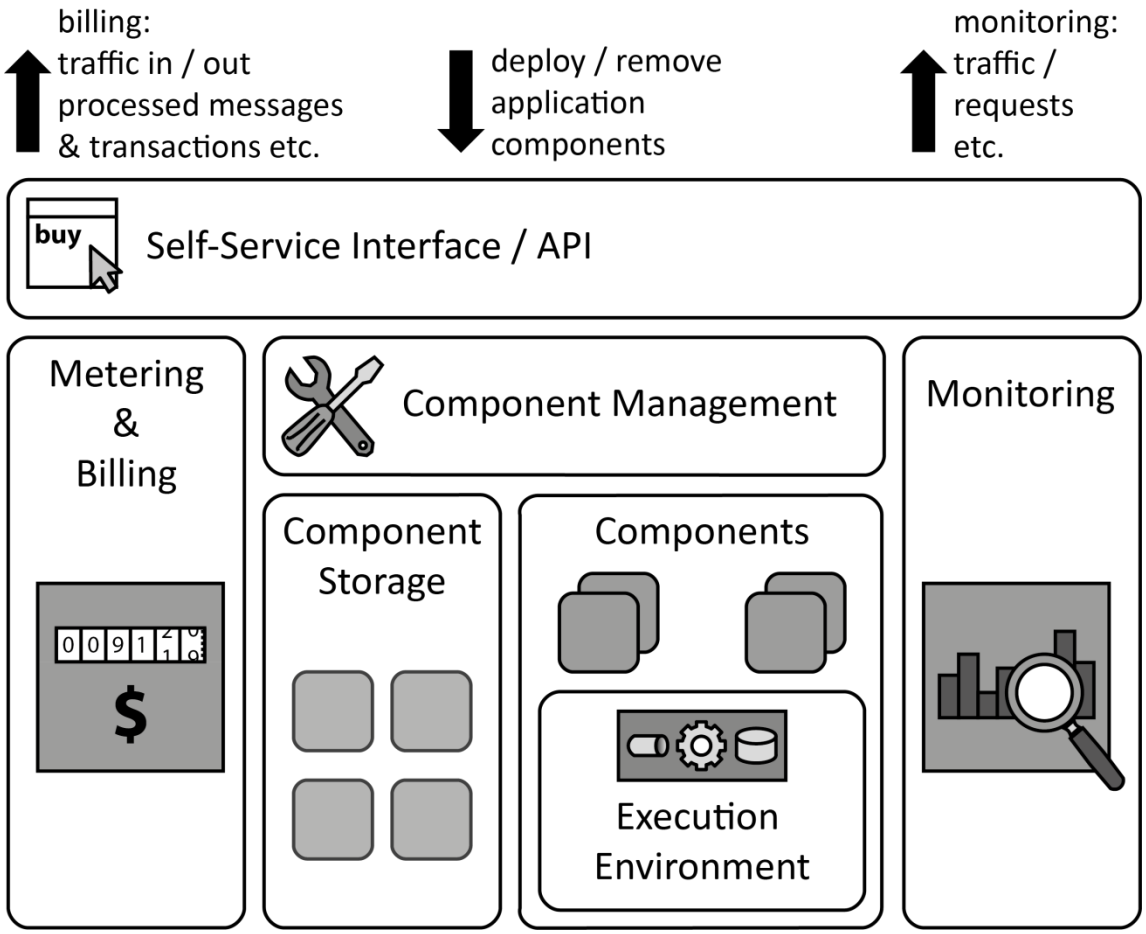
Cloud computing provides a set of computing resources with the following testable characteristics:

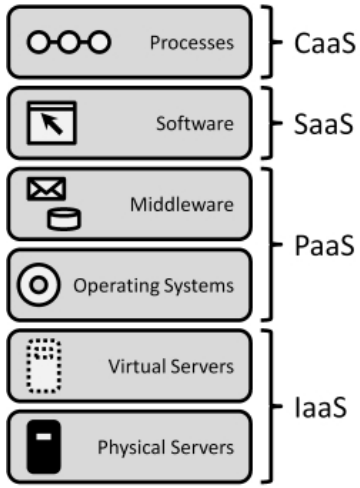
1. *On-demand*: the server is already setup and ready to be deployed (so the user can sign-up for the service without waiting)
2. *Self-service*: customer chooses what they want, when they want it (the user can use the service anytime, without waiting)
3. *Scalable*: customer can choose how much they want and ramp up if necessary (the user can scale-up the service when needed, without waiting for the provider to add more capacity)
4. *Measurable*: there's metering/reporting so you know you are getting what you pay for (the user can access measurable data to determine the status of the service)

In summary, cloud computing is OSSM (pronounced 'awesome').

Reference: B. Kepes, CloudU (online training, sponsored by RackSpace), Dave Nielsen, Cloud Camps, http://www.daveslist.com

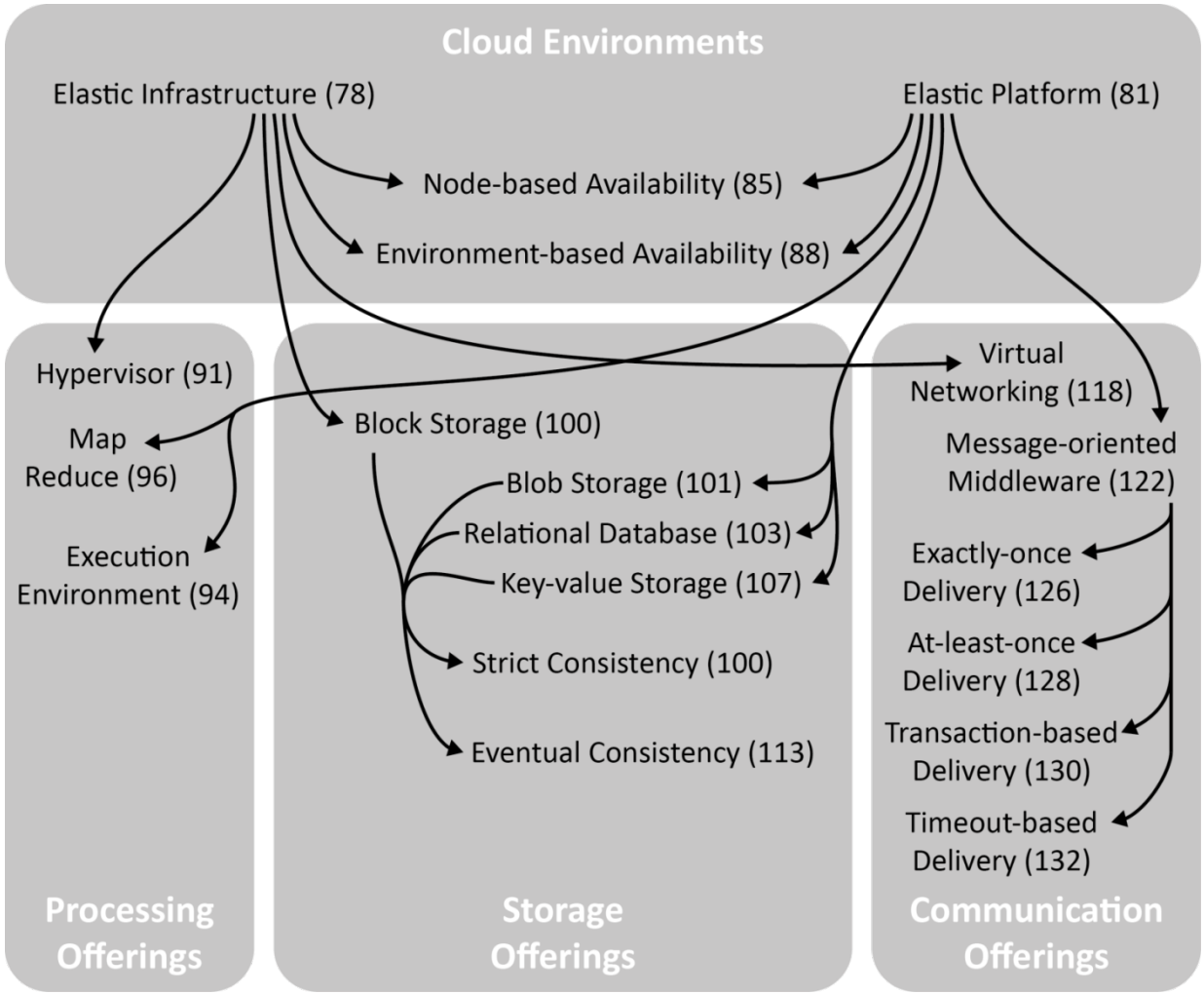
Cloud Computing Patterns (CCP)



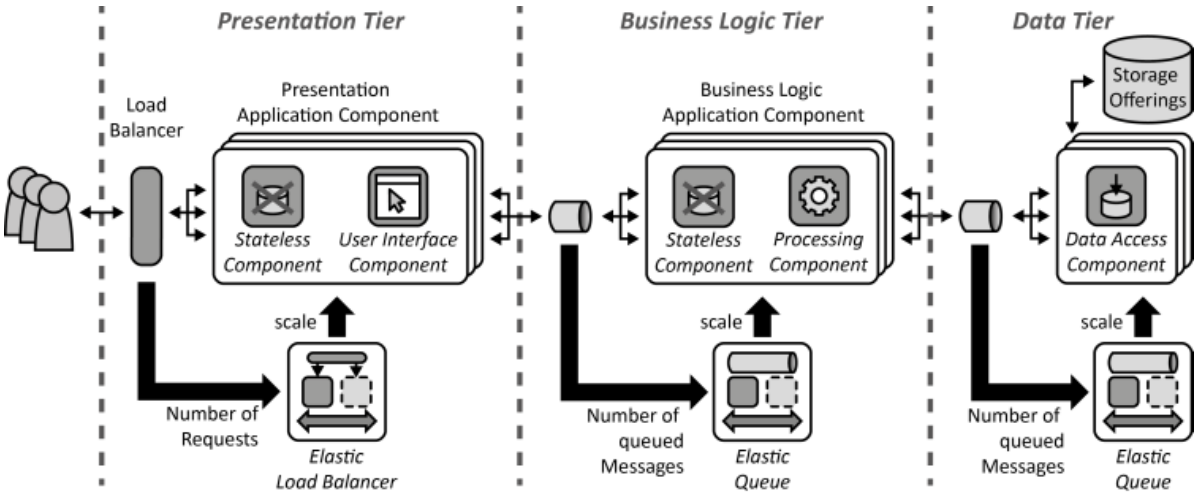


Reference: Cloud Computing Patterns, Springer 2014, http://cloudcomputingpatterns.org/

Cloud Computing Patterns (CCP) Map – Cloud Offerings



Cloud Application Components (Source: CCP)



- http://www.cloudcomputingpatterns.org/Category:Cloud_Application_Components

Cloud Application Architectures

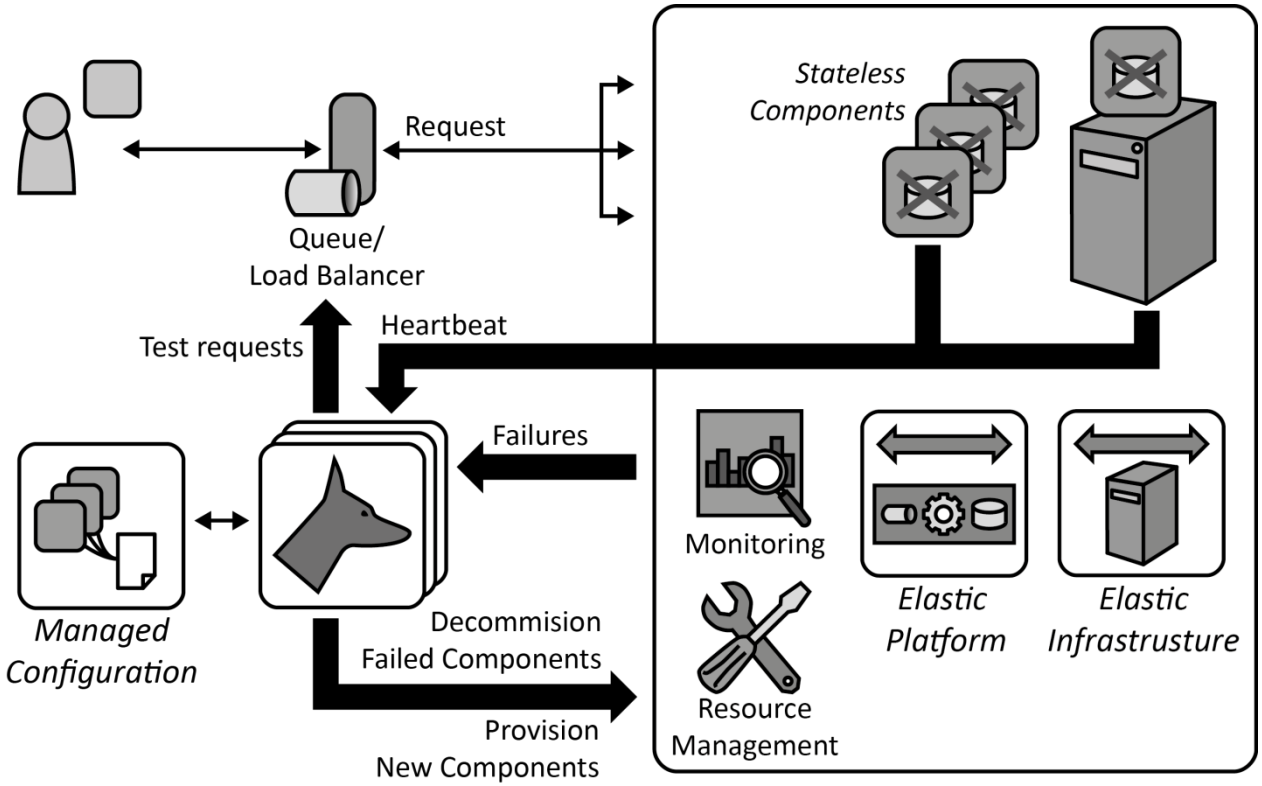
Fundamental Cloud Architectures
- Loose Coupling
- Distributed Application

Cloud Application Components
- Stateful Component
- Stateless Component
- User Interface Component
- Processing Component
- Batch Processing Component
- Data Access Component
- Data Abstractor
- Idempotent Processor
- Transaction-based Processor
- Timeout-based Message Processor
- Multi-Component Image

Multi-Tenancy
- Shared Component
- Tenant-isolated Component
- Dedicated Component

Cloud Integration
- Restricted Data Access Component
- Message Mover
- Application Component Proxy
- Compliant Data Replication
- Integration Provider

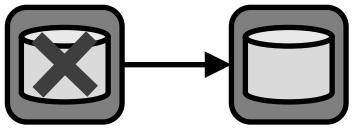
Watchdog Pattern



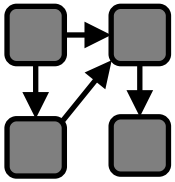
- Application components are *stateless*
- Component health is monitored
  - **Periodic heartbeats**: components notify that they are functioning
  - **Test requests**: result of test data is compared to expected results
  - **Environment**: provider-supplied reachability monitoring

IDEAL Cloud Application Properties (Fehling et al.)

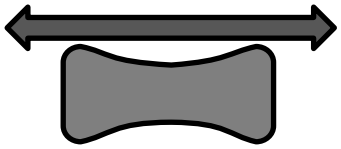
Reference: Cloud Computing Patterns, Springer 2014, http://cloudcomputingpatterns.org/



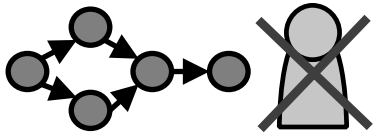
Isolated State: most of the application is *stateless* with respect to:
Session State: state of the communication with the application
Application State: data handled by the application



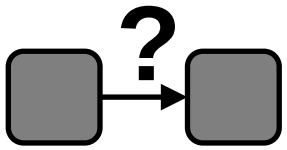
Distribution: applications are decomposed to…
… use multiple cloud resources
… support the fact that clouds are large globally distributed systems



Elasticity: applications can be scaled out dynamically
Scale out: performance increase through addition of resources
Scale up: performance increase by increasing resource capabilities

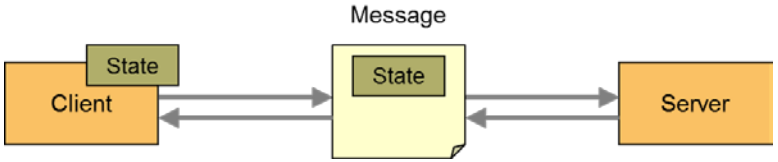


Automated Management: runtime tasks have to be handled quickly
Example: exploitation of pay-per-use by changing resource numbers
Example: resiliency by reacting to resource failures



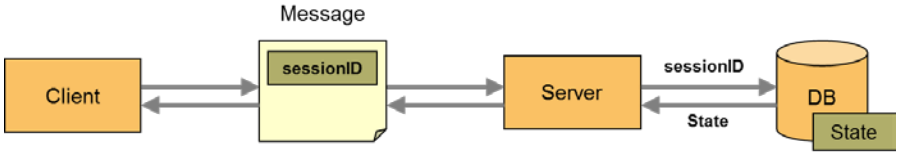
Loose Coupling: influence of application components is limited
Example: failures should not impact other components
Example: addition / removal of components is simplified

Session State Management Design – Options



- **Client Session State**
  - Scales well, but has security and possibly performance problems
  - This does not change when moving to a cloud platform.

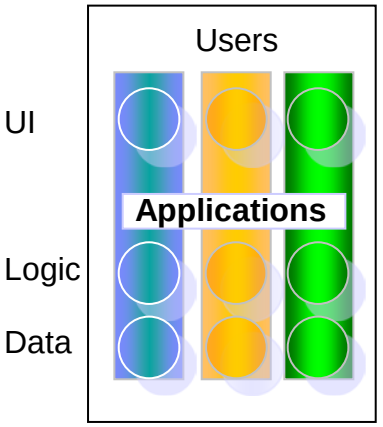
- **Server Session State**
  - Uses main memory or proprietary data stores in an application server (e.g. HTTP session in JEE servlet container)
  - Persistent HTTP sessions no longer recommended when deploying to a cloud due to scalability and reliability concerns.



- **Database Session State**
  - Is well supported in many clouds, e.g. via highly scalable key-value storage (a type of NoSQL database)

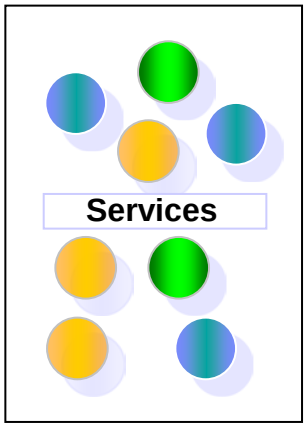
From Traditional Layer-Tier Architectures to Cloud Services

Traditional



Discrete Applications (Two or Three Tiers)

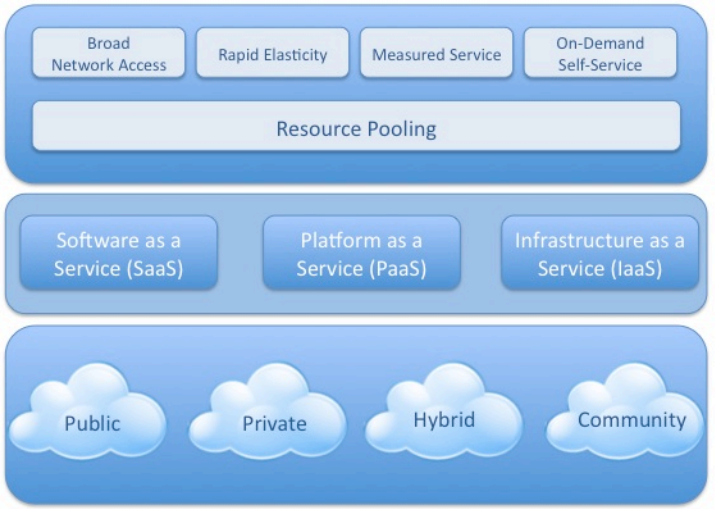
SOA



Basket of Services

?

Visual Model Of NIST Working Definition Of Cloud Computing
http://www.csrc.nist.gov/groups/SNS/cloud-computing/index.html



Decision-Centric Architectural Refactoring for Cloud (ARC)

What are Architectural Decisions (ADs)? Why Care?

Reference: SEI SATURN 2010 (IBM presentation)

- **"The design decisions that are costly to change" (Grady Booch, 2009)**
- **A more elaborate definition:**

  "Architectural decisions capture key design issues and the rationale behind chosen solutions. They are conscious design decisions concerning a software-intensive system as a whole or one or more of its core components and connectors in any given view. The outcome of architectural decisions influences the system's nonfunctional characteristics including its software quality attributes."

- **From IBM UMF work product description ART 0513 (since 1998):**

  "The purpose of the Architectural Decisions work product is to:

  - Provide a single place to find important architectural decisions
  - Make explicit the rationale and justification of architectural decisions
  - Preserve design integrity in the provision of functionality and its allocation to system components
  - Ensure that the architecture is extensible and can support an evolving system
  - Provide a reference of documented decisions for new people who join the project
  - Avoid unnecessary reconsideration of the same issues"

Y-Template for Architectural Decision Capturing

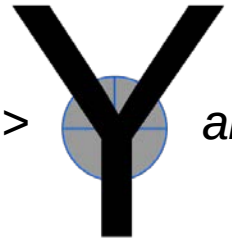
Reference: Sustainable Architectural Design Decisions, IEEE Software 30(6): 46-53 (2013)

- **Link to (non-)functional requirements and design context**
- **Tradeoffs between quality attributes**

In the context of <use case uc and/or component co>,

… facing <non-functional concern c>,

… we decided for <option o1>



and neglected <options o2 to oN>,

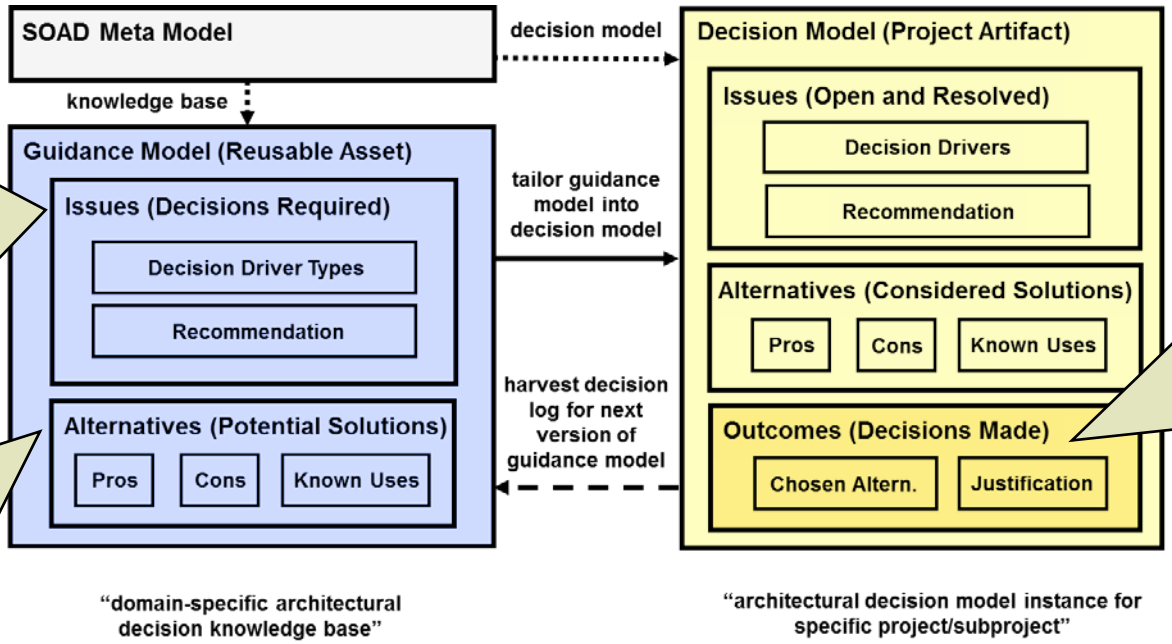
… to achieve <quality q>,

… accepting downside <consequence c>.

SOA Decision Modeling (2006-2011): Generic Metamodel

- **Existing metamodels and templates refactored and extended for reuse**
  - Before: documentation – after the fact (past tense)
  - With SOAD: design guidance – forward looking (future tense)

"When designing a presentation layer, you will have to select a *pattern to control the Web page flow*."

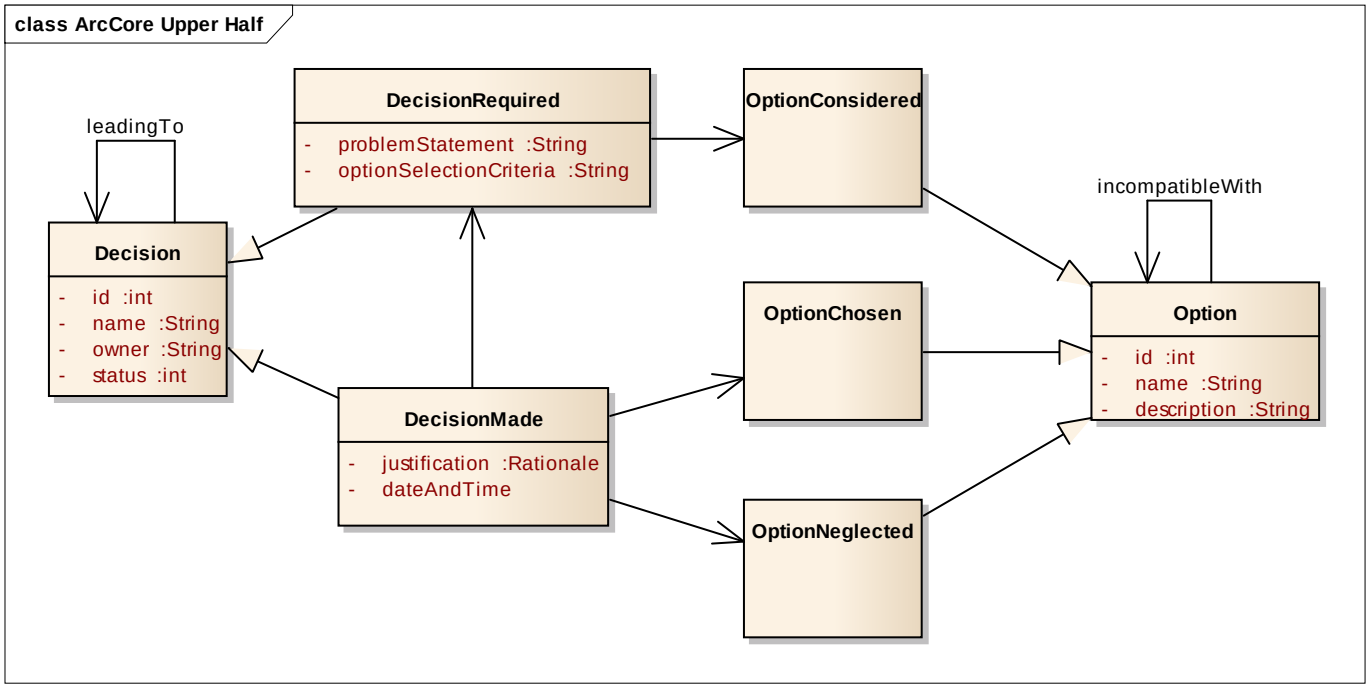


"*Model View Controller (MVC)* is a common architectural pattern to control the Web page flow."

"We decided for the MVC alternative to resolve the web page flow issue because we gained positive experience with it on many similar projects."

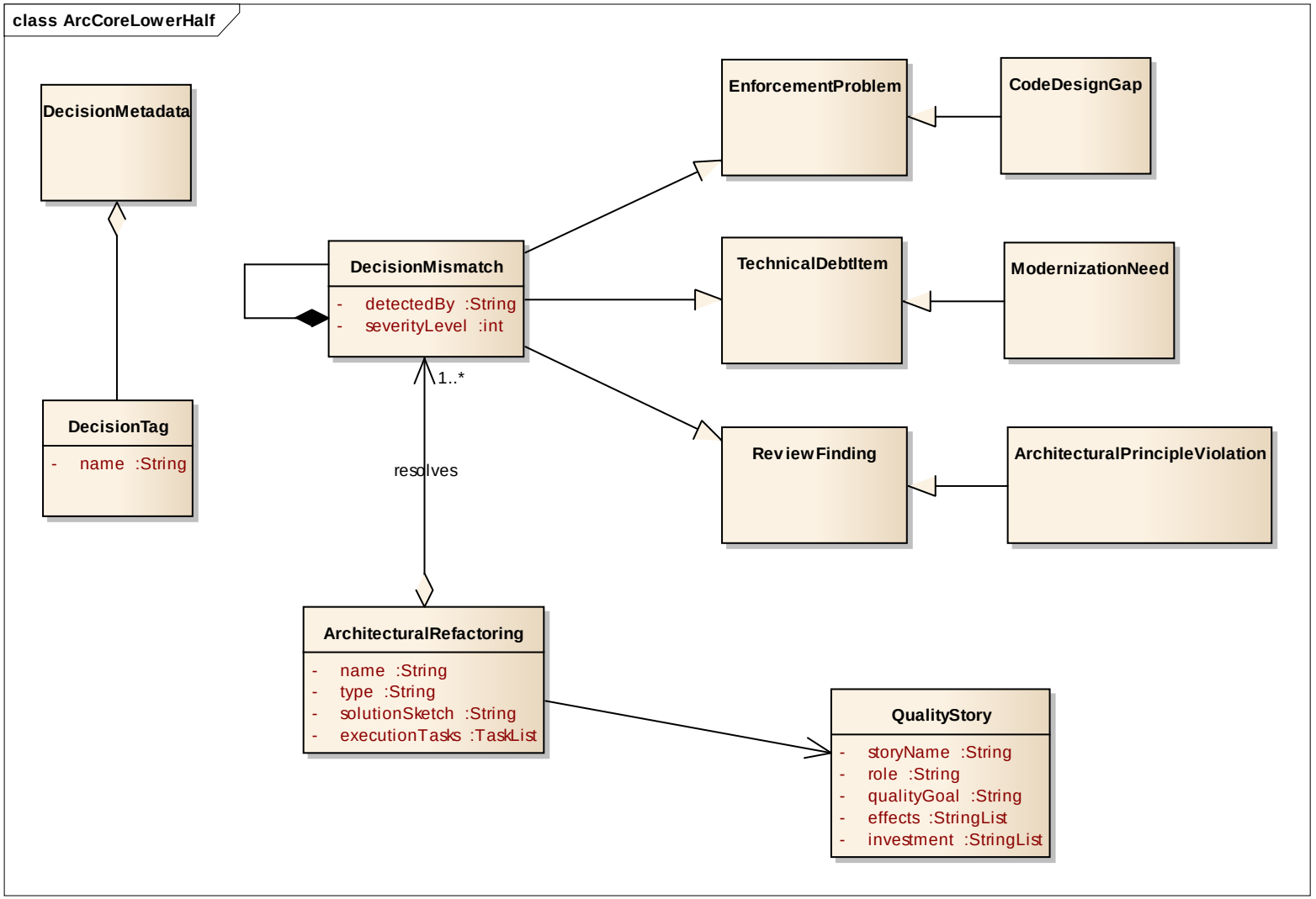
References: Architectural Decisions as Reusable Design Assets. IEEE Software 28(1): 64-69 (2011)
Reusable Architectural Decision Models for Enterprise Application Development. Proc. of QOSA 2007

ARC Metamodel (at an Initial State of Elaboration) (1/2)



- **Refactoring need arises from decision *mismatches***
  - Decision actually made vs. recommended decision (IDEAL)
  - Same problem (to be) solved differently (choosing different option)
  - Refactoring improves at least one quality attribute and preserves functionality

ARC Metamodel (at an Initial State of Elaboration) (2/2)



Architectural Refactoring for Cloud – Example: De-SQL

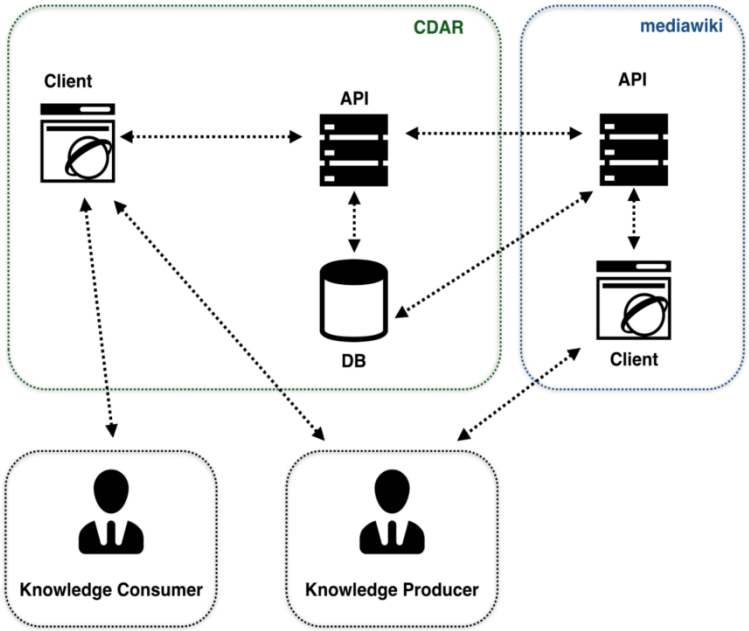
***Architectural Refactoring:* De-SQL**	
Context (viewpoint, refinement level): • Logical viewpoint, data viewpoint (all levels)	*Quality attributes and stories (forces):* • Flexibility, data integrity
Smell (refactoring driver): • It takes rather long to update the data model and to migrate existing data	
Architectural decision(s) to be revisited: • Choice of data modeling paradigm (current decision is: relational) • Choice of metamodel and query language (current decision is: SQL)	
Refactoring (solution sketch/evolution outline): • Use document-oriented database such as MongoDB instead of RDBMS such as MySQL • Redesign transaction management and database administration	
Affected components and connectors (if modelled explicitly): • Database • Data access layer	
Execution tasks (in agile planning tool and/or full-fledged design method): • Design document layout (i.e., the pendant to the machine-readable SQL DDL) • Write new data access layer, implement SQLish query capabilities yourself • Decide on transaction boundaries (if any), document database administration (CRUD, backup)	

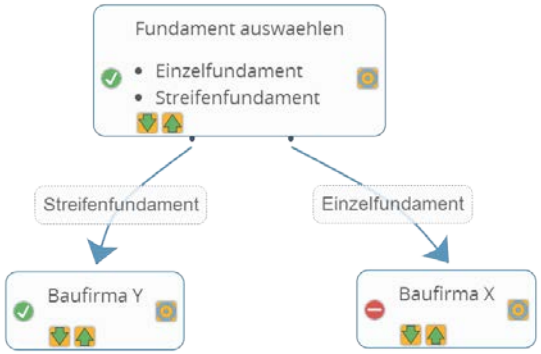
Candidate Architectural Refactorings for Cloud (Draft Catalog)

Category	Refactorings		
IaaS	Virtualize Server	Virtualize Storage	Virtualize Network
IaaS, PaaS	Swap Cloud Provider	Change Operating System	Open Port
PaaS	“De-SQL”	“BASEify” (remove ”ACID”)	Replace DBMS
PaaS	Change Messaging QoS	Upgrade Queue Endpoint(s)	Swap Messaging Provider
SaaS/application	Increase Concurrency	Add Cache	Precompute Results
SaaS/application	(CCP book, CBDI-SAE)	(all Stal refactorings)	(PoEAA/Fowler patterns)
Scalability	Change Strategy (Scale Up vs. Scale Out)	Replace Own Cache with Provider Capability	Add Cloud Resource (xaaS)
Performance	Add Lazy Loading	Move State to Database	
Communication	Change Message Exchange Pattern	Replace Transport Protocol	Change Protocol Provider
User management	Swap IAM Provider	Replicate Credential Store	Federate Identities
Service/deployment model changes	Move Workload to Cloud (use XaaS)	Privatize Deployment, Publicize Deployment	Merge Deployments (Use Hybrid Cloud)

Ausblick: Decision Collaboration & Refactoring Knowledge Tool

- **Collaborative Decision Management and Architectural Refactoring (CDAR) Tool**
  - RESTful integration of Browser user interface/workflow engine with MediaWiki (the wiki engine behind Wikipedia) via semantic links





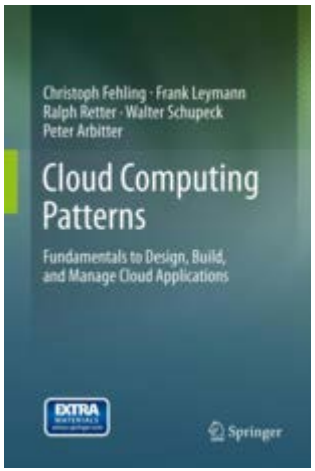
- **Add In for Sparx Enterprise Architect under construction**
  - Joint work with ABB Corporate Research

Summary and Discussion

- **Cloud computing is OSSM, pattern exist – but many open research and development questions remain (some of which resemble those in I4.0)**
  - E.g. cloud application lifecycle, cloud service management, cloud interoperability, cloud security,
- **Architectural decision making and architectural refactoring a key responsibilities of IT architects which are often underestimated and underrepresented in existing methods and tools.**
  - New task-centric templates and knowledge-centric tools required
- **In cloud design and other domains (including automation), many architectural decisions recur. This makes it possible to share architectural decision and refactoring knowledge including best practices (design acceleration and quality assurance).**
  - Cloud decision and refactoring catalog under construction
- **Tool support for decision modeling with reuse and for architectural refactoring is emerging**
  - Decision management, planning of decision execution (project planning)

More Information – Cloud Computing

- **Cloud research project and deployment lab at HSR**
  - Via http://www.ifs.hsr.ch/Olaf-Zimmermann.11623.0.html?&L=4 and ozimmerm@hsr.ch
- **Cloud Computing Patterns (Springer 2014)**
  - http://cloudcomputingpatterns.org
- **Online-Schulung**
  - E.g. Rackspace Cloud University (CloudU), http://www.rackspace.com/knowledge_center/cloudu/
- **Analysten-Reports und Knowledge Hubs**
  - z.B. InfoWorld, DZone
- **Blogs**
  - http://searchcloudcomputing.techtarget.com/feature/Top-five-must-read-cloud-computing-blogs



More Information – Architectural Decisions & Refactoring

- **Architectural Decision (AD) Capturing and Reuse:**
  - J. Tyree/A. Akerman, Architecture Decisions: Demystifying Architecture. IEEE Software, 22/2, March/April 2005
- **Architectural Refactoring**
  - M. Stal, Refactoring Software Architecture, Chapter 3 in Agile Software Architecture, Elsevier 2013 (also see his blog posts and OOP tutorial)
- **Cloud Reengineering Knowledge**
  - IAAS (University of Stuttgart), http://www.cloud-data-migration.com/
  - T. Höllwarth, Cloud Migration, http://www.cloud-migration.eu/
  - CBDI-SAE, Cloud Migration Patterns, http://everware-cbdi.com/index.php?cID=pattern-index&tab=520
  - Migration patterns
  - Cloud migration research