Studiengang Informatik

ARCHITECTURAL EVALUATION & LECTURE CLOSURE AND OUTLOOK

Application Architecture Lesson 14,
Lecture Fall Term (dt. Herbstsemester) 2017/2018

Prof. Dr. Olaf Zimmermann

Institute for Software

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL

FHO Fachhochschule Ostschweiz

Rapperswil, December 20, 2017

Agenda

- **Architectural Evaluation/Evolution in a Nutshell**
  - Purpose, steps, decision drivers
  - Exemplary evaluation methods: ATAM, DCAR
- **Repetition of a few selected topics**
  - Some clarifications and additional template guidance
- **More patterns, styles and middleware**
  - Business Process Management (BPM)/Business Rule Management (BRM)
  - Hexagonal architecture (a.k.a. ports and adapters)
  - Cloud computing
- **“Life as an IT Architect”**
  - Skills
  - Consulting Code of Conduct (professional integrity)
  - Road stories

© Olaf Zimmermann, 2017.

Architectural Evaluation in a Nutshell

- **Top-level questions (for interviews and document reviews):**
  - Have the right decisions been made, and have they been made right?
  - More specifically:
    - Can the designed architecture meet the elicited NFRs (and other requirements, including constraints)?
    - Does the implementation match the designed architecture?

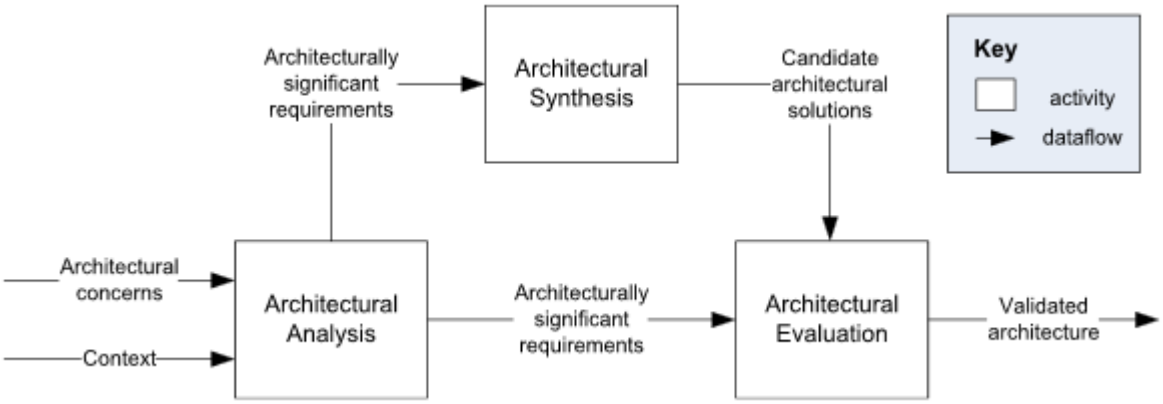
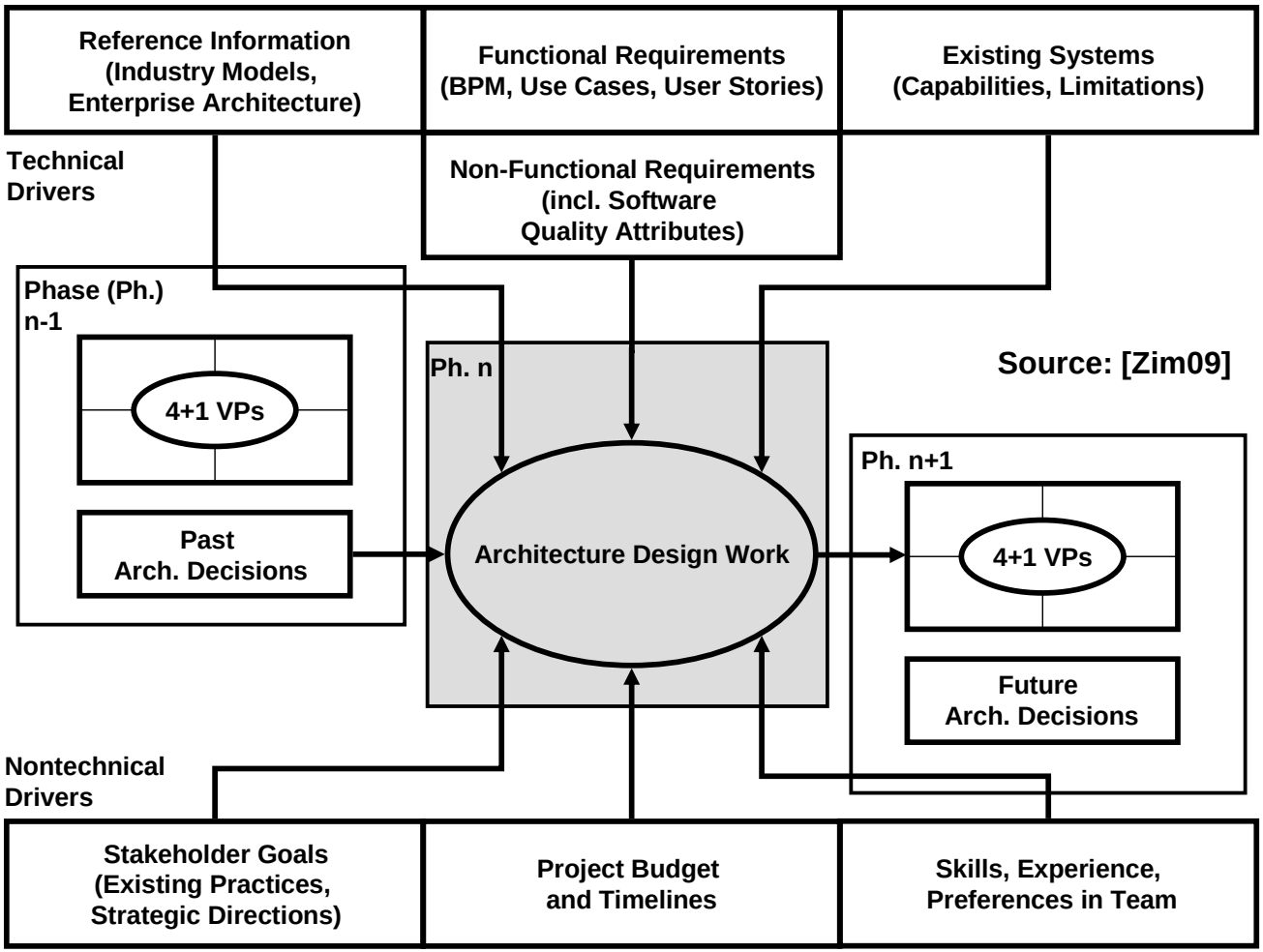


Figure reference: Christine Hofmeister, Philippe Kruchten, Robert L. Nord, J. Henk Obbink, Alexander Ran, Pierre America, *A general model of software architecture design derived from five industrial approaches*, Journal of Systems and Software, 2007

© Olaf Zimmermann, 2017.

Many decision making forces and consequences exist.



© Olaf Zimmermann, 2017.

Typical Tradeoffs (dt. Zielkonflikte) in Practice

- **Performance vs. security**
- **Security vs. usability**
- **Performance vs. supportability**
- **Reliability vs. simplicity**
- **Scalability vs. manageability**
  - CAP theorem
  - BAC theorem
- **I/O vs. CPU usage**
- **Hardware vs. software solution**

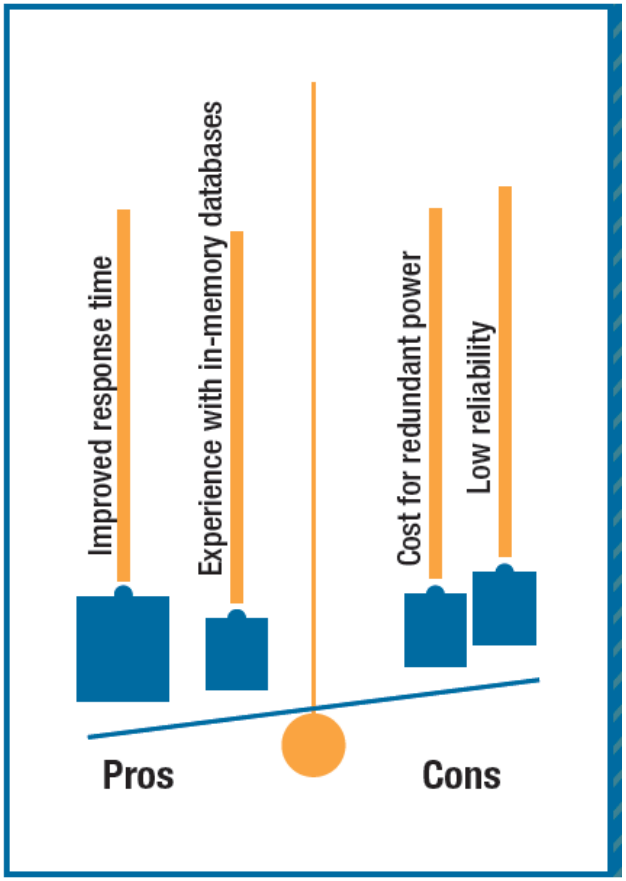


Figure reference: van Heesch, U., Eloranta, V. P., Avgeriou, P., Koskimies, K., & Harrison, N. (2014). Decision-Centric Architecture Reviews. IEEE Software, 31(1), 69-76

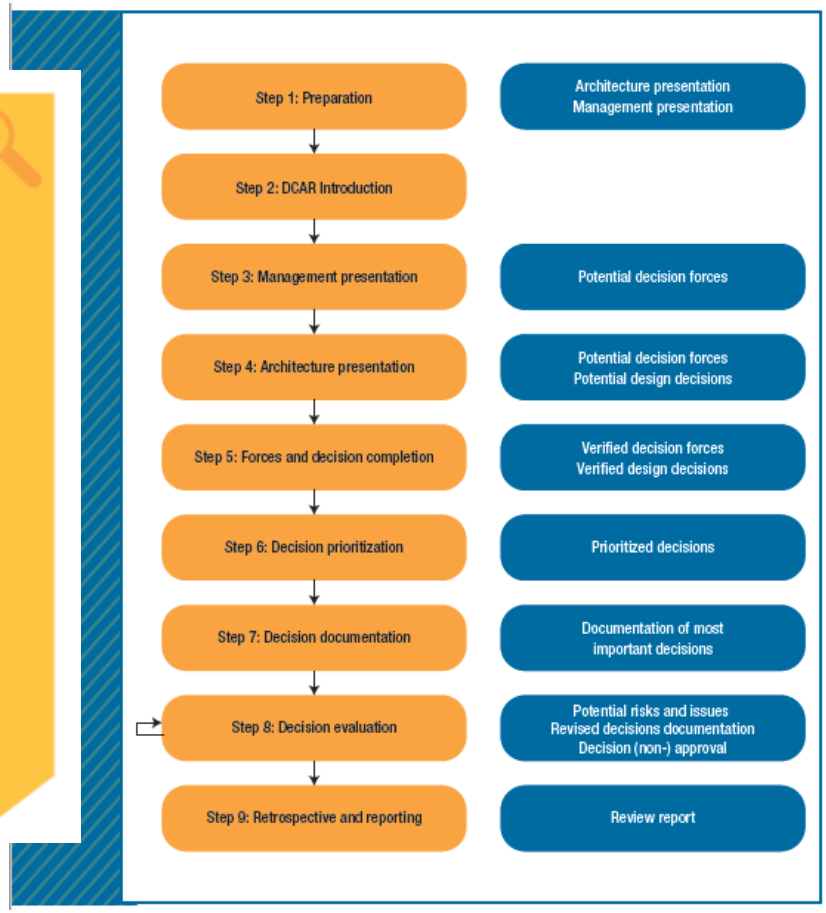
© Olaf Zimmermann, 2017.

Decision-Centric Architecture Reviews (DCAR)

- https://www.vanheesch.net/papers/dcar-ieeeSW.pdf

DCAR: SHORT PROFILE

Evaluation objectives: determine the soundness of architectural decisions that were made
Inputs for evaluation: informal description of requirements, business drivers, and architectural design
Knowledge of evaluators: general knowledge about software architecture
Output: risks, issues, and thorough documentation of the evaluated decisions and their decision forces
Priority setting of decisions: during the review
Project phase: within or after the architectural design is finalized
Reviewers: company-internal or external reviewers
Schedule: half a day preparation and postprocessing and half a day review session
Scope: a set of specific architecture decisions
Social interaction: face-to-face meeting between reviewers, architect, developers, and business representative
Tools or automation: templates, wiki, and UML tool



© Olaf Zimmermann, 2017.

Architecture Tradeoff Analysis Method (ATAM)

- **ATAM is the SEI method that covers architectural evaluation (not ADD):**
  - https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5177

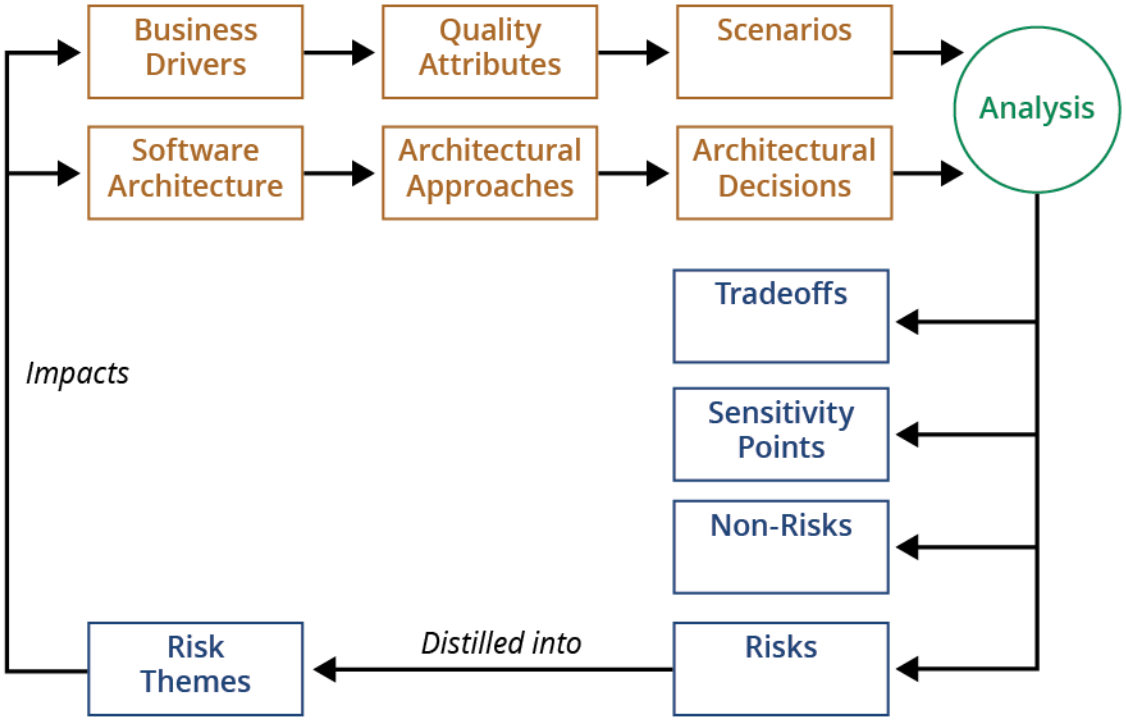


Figure reference: https://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm

© Olaf Zimmermann, 2017.

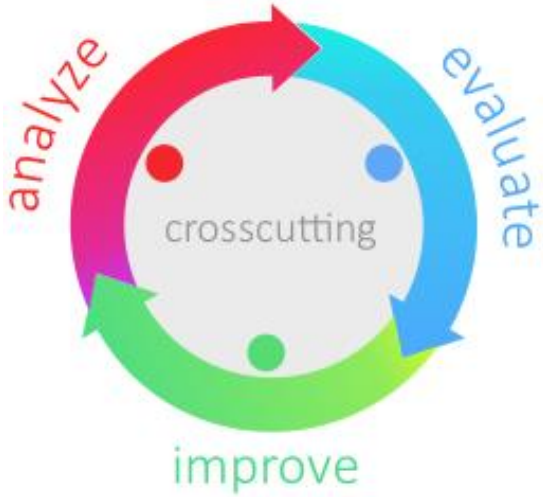
Architecural Evaluation: Questions to Ask

- **Chapter 8 in “Effektive Softwarearchitekturen” is on architectural evaluation (dt. Bewertung)**
  - Features ATAM, emphasizes role of QAS and Ads (for review)
  - Gives pointers, e.g. to SARA report
  - Questions to ask:
    - “Which architectural decisions were made to satisfy quality attribute scenario X?
    - Which architectural approach supports the satisfaction of scenario X
    - Which compromises (tradeoffs) were made along with this decision?
    - Which other quality characteristics or architectural goals does this decision influence?
    - Which risks stem from this decision or approach?
    - Which risks exist that could prevent the scenario and the related quality requirements from being satisfied??
    - Which analyses, investigations or prototypes back this decision?”

© Olaf Zimmermann, 2017.

Closing the Loop: From Evaluation to Improvement

- **Usage of a template such as the arc42 one simplifies the review task**
  - Review findings that cannot be fixed immediately should be collected as technical debt and improved over time
- **Related knowledge collected in aim42 community effort (co-initiated by G. Starke): http://aim42.org/overview**
  - Chapter 9 in "Effektive Softwarearchitekturen"



© Olaf Zimmermann, 2017.

Agenda

- **Architectural Evaluation/Evolution in a Nutshell**
  - Purpose, steps, decision drivers
  - Exemplary evaluation methods: ATAM, DCAR
- **Repetition of a few selected topics**
  - Some clarifications and additional template guidance
- **More patterns, styles and middleware**
  - Business Process Management (BPM)/Business Rule Management (BRM)
  - Hexagonal architecture (a.k.a. ports and adapters)
  - Cloud computing
- **“Life as an IT Architect”**
  - Skills
  - Consulting Code of Conduct (professional integrity)
  - Road stories

© Olaf Zimmermann, 2017.

Definition of Software Architecture

[BassClementsKazman98]

"The **structure** or structures of the system, which comprise software **elements**, the externally visible **properties** of those elements, and the **relationships** among them."

[JansenBosch05]

"A software system's architecture is the set of **principal design decisions** made about the system."

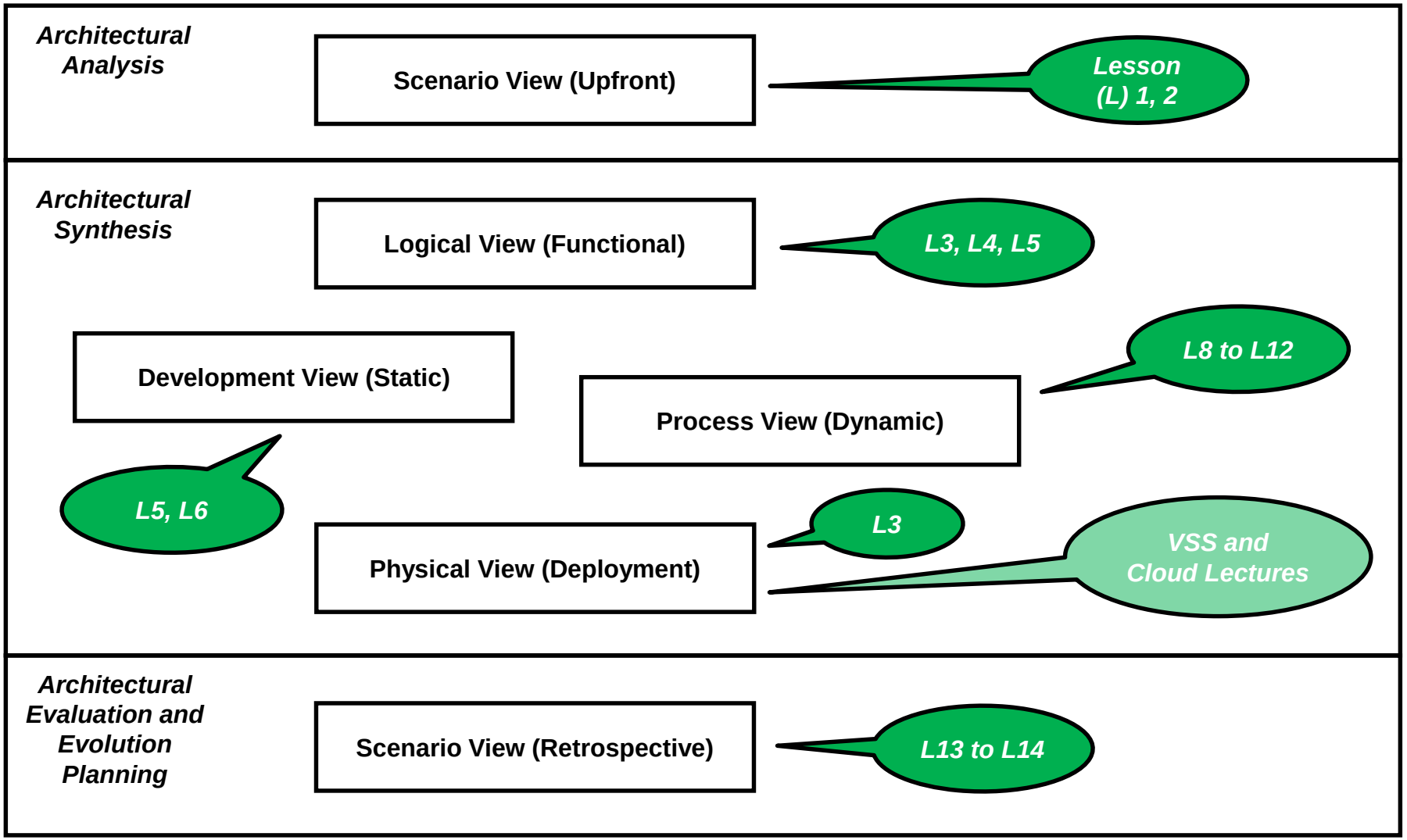
[ISO/IEC/IEEE 42010, November 2011]

"The fundamental organization of a system is embodied in its **components**, their **relationships** to each other, and to the environment, and the **principles** guiding its design and evolution."

What is the relationship between Software Architecture and Application Architecture?

© Olaf Zimmermann, 2017.

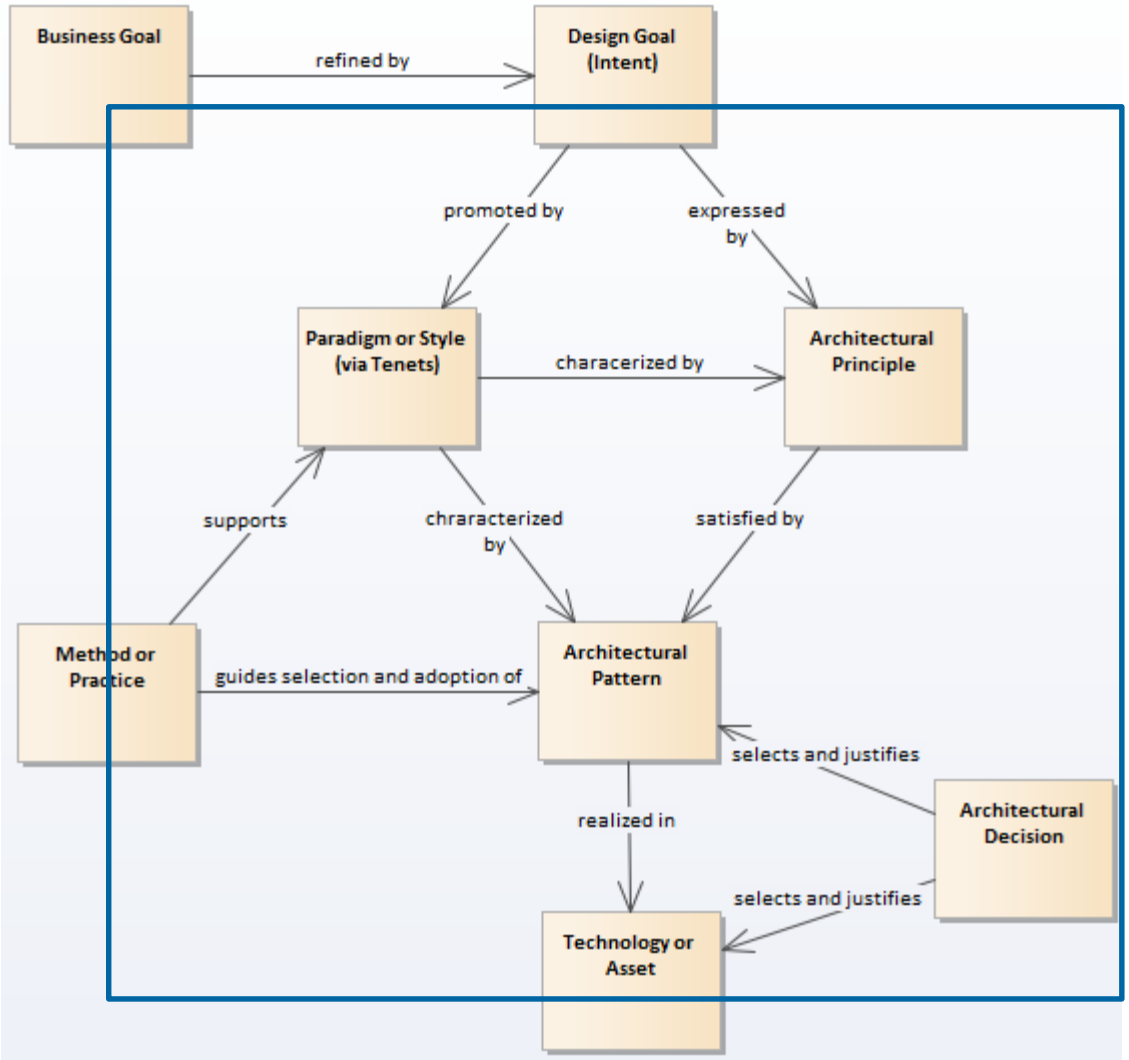
The Big Picture of this Module ("Architecture of AppArch")



© Olaf Zimmermann, 2017.

Software Architecture Essentials: Principles, Patterns, Decisions

- **Business goals and design goals**
- ***Paradigms (defined by tenets)***
- ***Principles***
- ***Patterns***
- ***Decisions***
- **Methods, practices, tools**



© Olaf Zimmermann, 2017.

SMART Nonfunctional Requirements

- **Nonfunctional Requirements (NFRs) sollten wie Projektziele SMART definiert sein, insbesondere also Specific und Measurable:**
  - S: z.B. Ist die funktionale Anforderung oder Architekturkomponente definiert, auf die sich das NFR bezieht?
  - M: Kann in Design, Programmierung und Test die Einhaltung des NFRs überprüft werden?

Buchstabe	Bedeutung	Beschreibung
S	Spezifisch	Ziele müssen eindeutig definiert sein (nicht vage, sondern so präzise wie möglich).
M	Messbar	Ziele müssen messbar sein (Messbarkeitskriterien).
A	Akzeptiert	Ziele müssen von den Empfängern akzeptiert werden/sein (auch: angemessen, attraktiv, ausführbar oder anspruchsvoll[2]).
R	Realistisch	Ziele müssen möglich sein.
T	Terminierbar	zu jedem Ziel gehört eine klare Terminvorgabe, bis wann das Ziel erreicht sein muss.

© Olaf Zimmermann, 2017.

Quality Attribute Scenario (QAS) Template

Scenario Portion	[Wie heisst das Szenario, um welche Art NFR geht es?]
Source	• Wer startet das Szenario (Quelle)? Woher kommt die Anforderung? [Bsp. Benutzer/Aktor, systeminterner Trigger]
Stimulus	• Was macht die Source, um das Szenario zu starten (Ereignis, Trigger/Auslöser)? [Use Case oder Aktivität im Geschäftsprozess]
Artifact	• Wo findet das beschriebene Response-Verhalten, das von der Source stimuliert worden ist, statt? Welcher Systemteil ist betroffen? [Ganze Anwendung oder einzelne Komponente]
Environment	• Wann kann das spezifizierte Response-Verhalten beobachtet werden? Um welche Umgebung/welchen Systemzustand geht es? [Laufzeit oder Phase im Software Engineering/Wartung]
Response	• Wie reagiert die Anwendung qualitativ auf den Stimulus? Was soll gemessen werden? [nach aussen sichtbares Verhalten, Messgrösse, noch kein Design]
RespMeasure	• Wie kann die Response quantifiziert werden, um sie überprüfbar zu machen? Welches Messergebnis wird angestrebt? [Messeinheit, konkrete Zahlenangabe im Kontext der vorherigen QAS-Template-Elemente wie Source, Stimulus, usw.]

© Olaf Zimmermann, 2017.

Components, Responsibilities, Collaborations (CRC) Card

Component: **[Name of Component]**	
Responsibilities: • [What is this component capable of doing (services provided)?] • [Which data does it deal with?] • [How does it do this in terms of key system qualities?]	*Collaborations (Interfaces):* • [Who invokes this component (service consumers)?] • [Whom does this component call to fulfill its responsibilities (service providers)?] • [Any external connections (both active and passive)?]
Known uses (implementations): • [Which technologies, products (commercial, open source) and internal assets realize the outlined component functionality (responsibilities)?]	

- **Stichworte und Kurzsätze statt langer Spezifikationsprosa oder Code**
- **Notation stammt aus der objektorientierten Programmierung und ist auf Software-Architektur und Web-Technologien sehr gut anwendbar**

© Olaf Zimmermann, 2017.

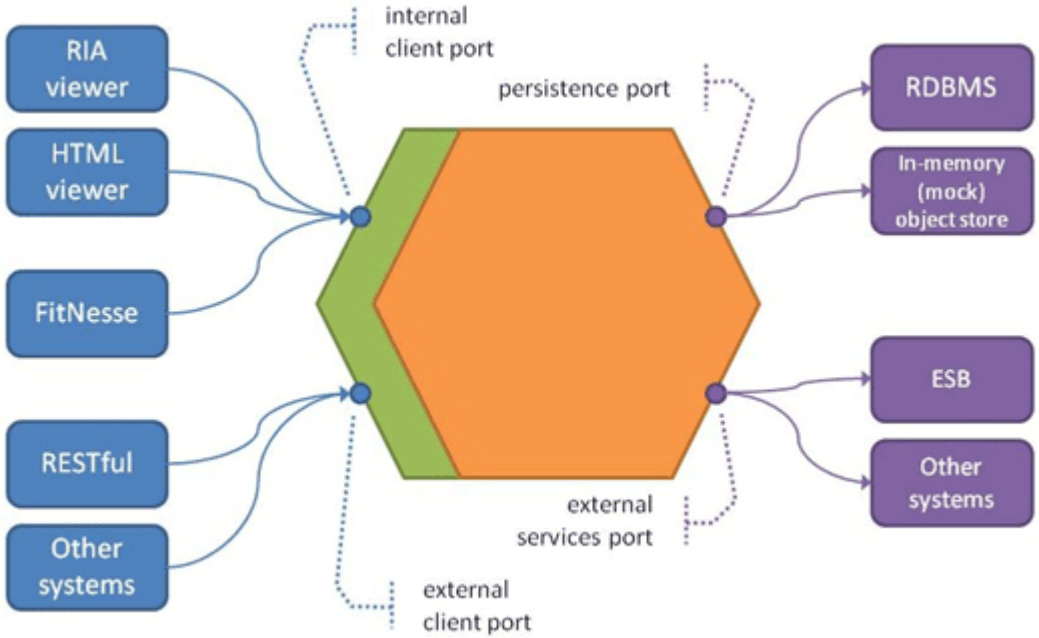
DDD Clarification: Bounded Context vs. Subdomain

- **Reality vs. wishful thinking (“Fremdbild-Eigenbild”)**
- **Seven integration and collaboration fallacies (ZIO’s view)**
  1. People know/respect/like each other, value each other’s opinions.
  2. People talk to each other and share information freely, everybody knows everything (s)he needs to know to make sound ADs.
  3. Projects are coordinated and do not overlap unnecessarily (no conflicts).
  4. Project budgets are sufficient. (time, money).
  5. Project run at the speed of the most agile and eager/ambitious/advanced/mature team.
  6. Reorganizations, mergers, acquisitions and divestitures happen rarely and take IT and team considerations into account.
  7. Technology never becomes obsolete, vendors care about their customers.

If all of these were true, the subdomain-to-BC relation was 1:1 ☺

© Olaf Zimmermann, 2017.

Enterprise Appl. Integration and DDD: Anti Corruption Layer (ACL)



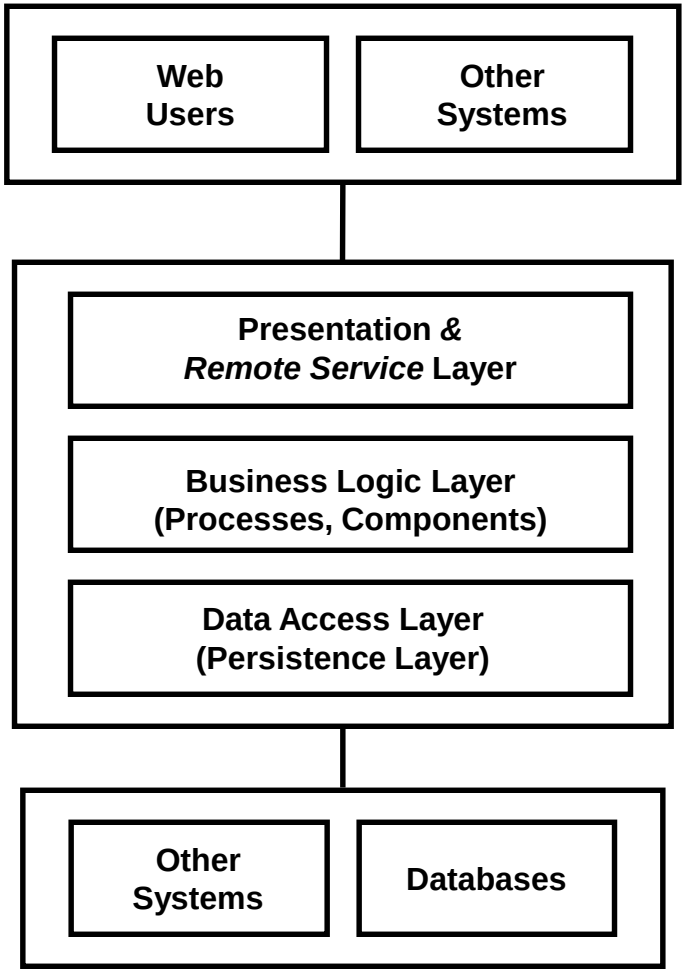
- **Architecture visualized as suggested by A. Cockburn (hexagonal)**
  - ACL concepts explained here
  - ACL case study presented here

© Olaf Zimmermann, 2017.

Modifiziertes Schichtenbild (Wdh.)

Wohin in der Dreischichtenarchitektur im Mid Tier gehört der Service Layer?

- **Service Layer kann zwischen PL und BLL positioniert werden**
- **Alternativ kann der Service Layer auch als oberer Eingang des BLL angesehen werden**
- **Dritte Modellierungsmöglichkeit: Service Layer kann auch als neuer, paralleler PL gesehen werden**
  - nicht für Human User mit Browser
  - sondern für Machine (System) mit Remote API, z.B. Web API



© Olaf Zimmermann, 2017.

Recurring Architectural Decision (AD) Issues (1/2)

Artifact	Decision Topic	Recurring Issues (Decisions Required)
Enterprise architecture documentation	IT strategy	Buy vs. build strategy, open source policy
	Governance	Methods (processes, notations), tools, reference architectures, coding guidelines, naming standards, asset ownership
System context	Project scope	External interfaces, incoming and outgoing calls (protocols, formats, identifiers), service level agreements, billing
Other viewpoints	Development process	Configuration management, test cases, build/test/production environment staging
	Physical tiers	Locations, security zones, nodes, load balancing, failover, storage placement
	Data management	Data model reach (enterprise-wide?), synchronization/replication, backup strategy
Architecture overview diagram	Logical layers	Coupling and cohesion principles, functional decomposition (partitioning)
	Physical tiers	Locations, security zones, nodes, load balancing, failover, storage placement
	Data management	Data model reach (enterprise-wide?), synchronization/replication, backup strategy
Architecture overview diagram	Presentation layer	Rich vs. thin client, multi-channel design, client conversations, session management
	Domain layer (process control flow)	How to ensure process and resource integrity, business and system transactionality
	Domain layer (remote interfaces)	Remote contract design (interfaces, protocols, formats, timeout management)
	Domain layer (component-based development)	Interface contract language, parameter validation, Application Programming Interface (API) design, domain model
	Resource (data) access layer	Connection pooling, concurrency (auto commit?), information integration, caching
	Integration	Hub-and-spoke vs. direct, synchrony, message queuing, data formats, registration

Source: O. Zimmermann, Architectural Decision Identification in Architectural Patterns. WICSA/ECSA Companion Volume 2012, Pages 96-103.

© Olaf Zimmermann, 2017.

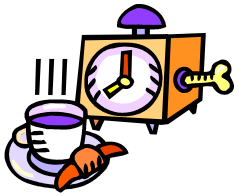
Recurring Architectural Decision (AD) Issues (2/2)

Artifact	Decision Topic	Recurring Issues (Decisions Required)
Logical component	Security	Authentication, authorization, confidentiality, integrity, non-repudiation, tenancy
	Systems management	Fault, configuration, accounting, performance, and security management
	Lifecycle management	Lookup, creation, static vs. dynamic activation, instance pooling, housekeeping
	Logging	Log source and sink, protocol, format, level of detail (verbosity levels)
	Error handling	Error logging, reporting, propagation, display, analysis, recovery
Components and connectors	Implementation technology	Technology standard version and profile to use, deployment descriptor settings (QoS)
	Deployment	Collocation, standalone vs. clustered
Physical node	Capacity planning	Hardware and software sizing, topologies
	Systems management	Monitoring concept, backup procedures, update management, disaster recovery

Source: O. Zimmermann, Architectural Decision Identification in Architectural Patterns. WICSA/ECSA Companion Volume 2012, Pages 96-103.

© Olaf Zimmermann, 2017.

PAUSE



© Olaf Zimmermann, 2017.

Agenda

- **Architectural Evaluation/Evolution in a Nutshell**
  - Purpose, steps, decision drivers
  - Exemplary evaluation methods: ATAM, DCAR
- **Repetition of a few selected topics**
  - Some clarifications and additional template guidance
- **More patterns, styles and middleware**
  - Business Process Management (BPM)/Business Rule Management (BRM)
  - Hexagonal architecture (a.k.a. ports and adapters)
  - Cloud computing
- **“Life as an IT Architect”**
  - Skills
  - Consulting Code of Conduct (professional integrity)
  - Road stories

© Olaf Zimmermann, 2017.

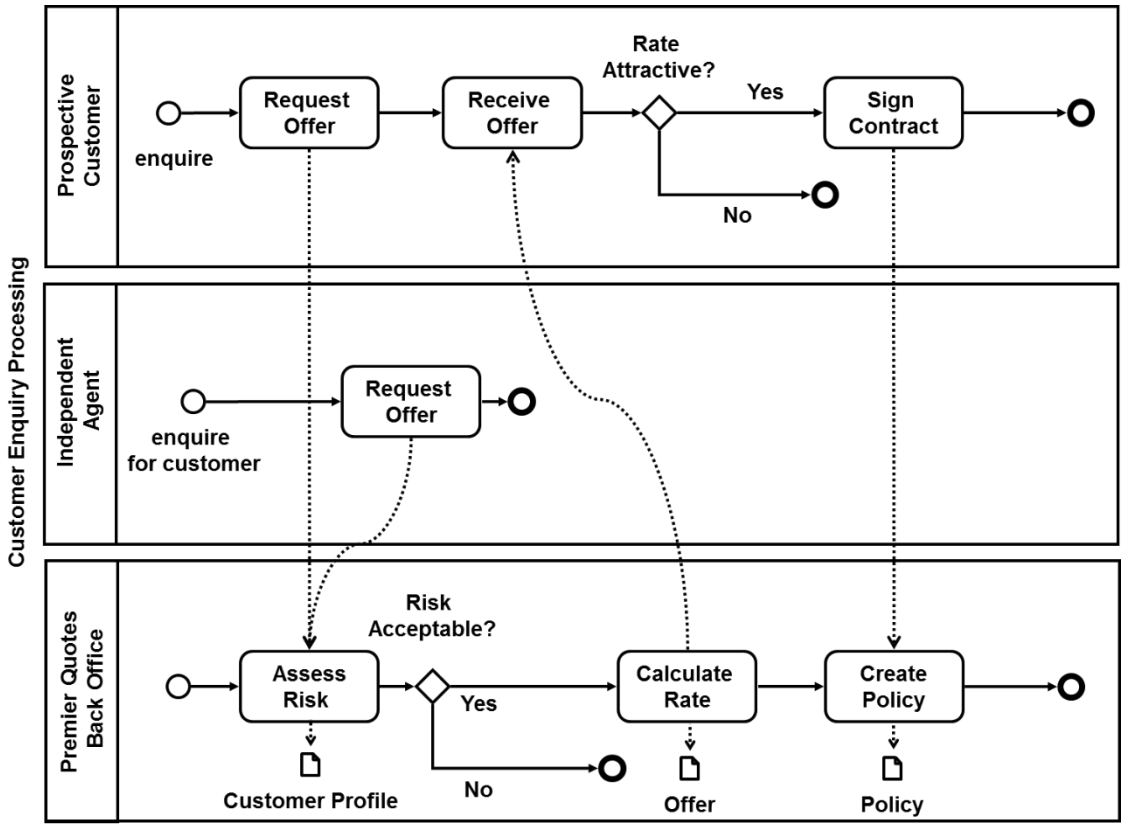
Additional Topics and Candidate Assets

- **Further Patterns of Enterprise Application Architecture by M. Fowler**
  - http://martinfowler.com/eaaDev/index.html
  - Presentation Layer Updates, Accounting Patterns
- **Time and Money Library by Domain-Driven Design followers (original book author E. Evans involved)**
  - http://sourceforge.net/projects/timeandmoney/
- **Batch processing**
  - E.g. Quartz Job Scheduling Library
- **Business rules and business event management**
  - E.g. Drools platform (JBoss), JRules (ILOG/IBM), Bosch Visual Studio
- **Model-Driven Software Engineering (MDSE) and Remote API design**
  - Eclipse Modeling Framework (EMF), Domain-Specific Languages (e.g. built with Eclipse Xtend/Eclipse Xtext)

© Olaf Zimmermann, 2017.

Business Process Model and Notation (BPMN)

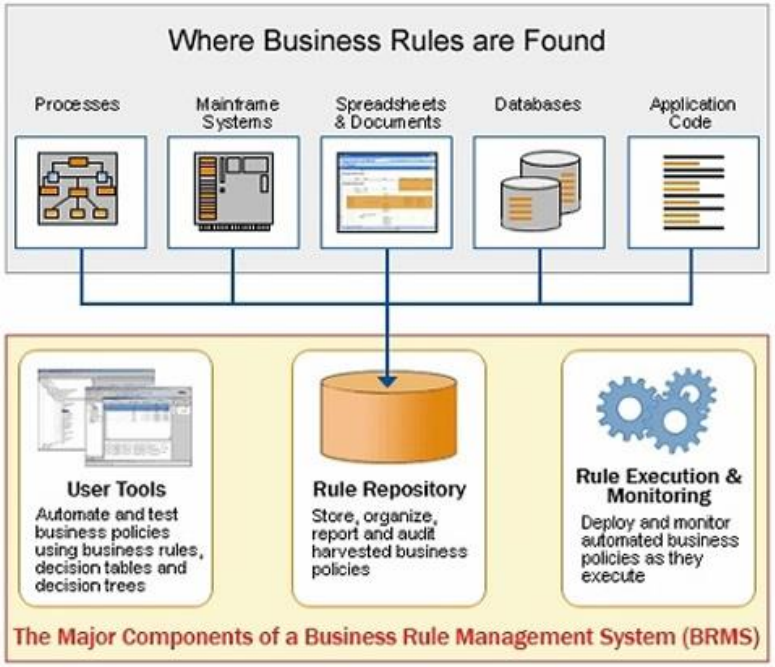
- **Business Process Model and Notation (BPMN) is an OMG standard for modeling business processes**
  - Previously also called Business Process Modeling Notation
- **Current version: BPMN 2.0 (final adopted specification)**
  - Previous versions: BPMN 1.2, BPMN 1.1, BPMN 1.0



© Olaf Zimmermann (Jochen Küster), 2017.

Business Rules Management Systems (BRMS)

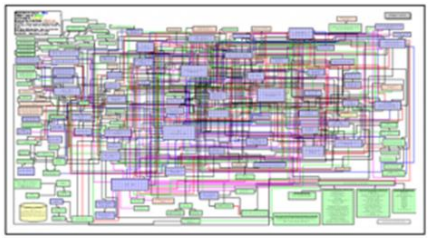
- **Decision trees and other abstractions, interpreted by a rules engine**
  - Various reasoning/inferencing algorithms (logical programming paradigm)

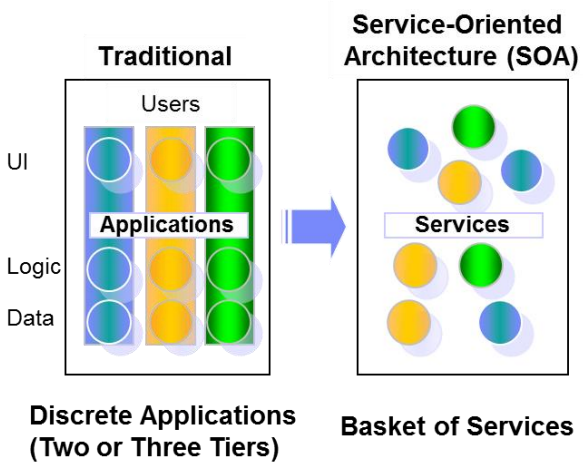


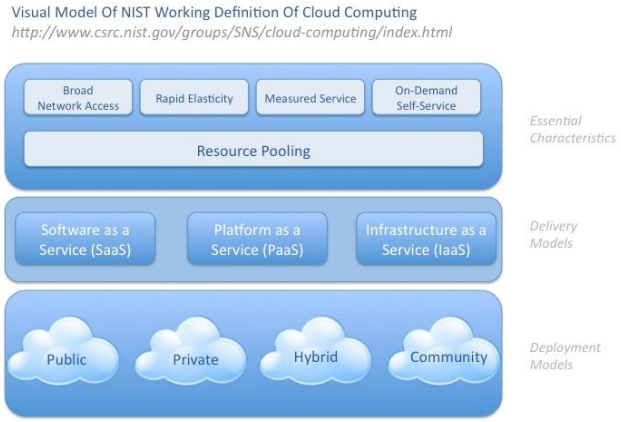
- **Sample product architecture, product presentation; free online book**

© Olaf Zimmermann, 2017.

New Operational Models and Deployment/Hosting Options





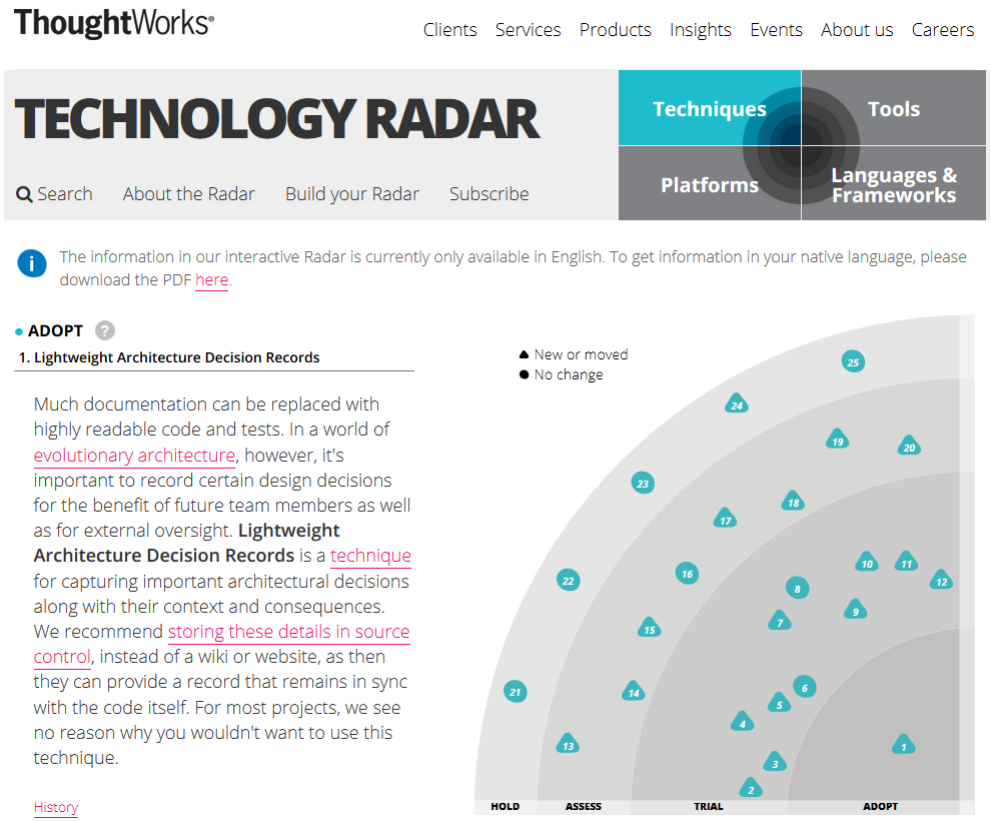


- **Own Data Center**
- **Strategic Outsourcing**
- ***Cloud Computing* (Managed Hosting)**

© Olaf Zimmermann , 2017

Technology Trends as Seen by a Global Consultancy

- **Somewhat biased but still informative**
  - All things software engineering (including software/application architecture)



© Olaf Zimmermann, 2017.

Agenda

- **Architectural Evaluation/Evolution in a Nutshell**
  - Purpose, steps, decision drivers
  - Exemplary evaluation methods: ATAM, DCAR
- **Repetition of a few selected topics**
  - Some clarifications and additional template guidance
- **More patterns, styles and middleware**
  - Business Process Management (BPM)/Business Rule Management (BRM)
  - Hexagonal architecture (a.k.a. ports and adapters)
  - Cloud computing
- **“Life as an IT Architect”**
  - Skills
  - Consulting Code of Conduct (professional integrity)
  - Road stories

© Olaf Zimmermann, 2017.

Skills and Traits of Consultants (Consulting IT Architects)

- **Ability to listen**
  - Active, multiple times, …
- **Ability to ask**
  - Ask the right questions, and ask them right
- **Ability to say no**
  - In a constructive way – almost everything can be built if budget is there

- **Ability to deal with incomplete and conflicting information**
- **Curiosity (domains, people, business models, …)**
- **Get-the-job-done mentality (due date, bug fix, political turmoil)**
- **Ability to travel (schedule, location issues?)**
- **Humor, flexibility, helper attitude; other social skills**

© Olaf Zimmermann, 2017.

Key Performance Indicators (KPIs) for Consulting IT Architects

- **Client satisfaction (CSAT)**
- **Utilization rate**
- **Total contract volume, project profitability**
- **Project health (budget, completion rate, end user feedback)**
- **Code metrics, software architecture metrics, process metrics**

- **Contributions (to the service business of employer)**
  - Note: one more overloaded term!
- **Giveback (to the community)**
  - If you still have time
- **Work-life balance**
  - If you achieve all other KPIs (over several years), this will be hard!

© Olaf Zimmermann, 2017.

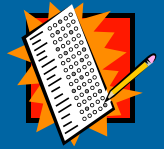
IT Consultant Tenets and Code of Conduct (Subjective View)*

- **Marketing Opens Doors – *Technical Excellence* Creates Opportunities**
  - Conferences, articles, academic degrees, continued education
- ***Responsiveness* Expected by Client**
  - "Blitz chess" metaphor: when the client has been active, you get active too
- ***Context* Matters and *Wants vs. Needs* Differ**
  - Kruchten's Octopus dimensions: do not blindly transfer "best" practices
  - Articulated requirements do not always equal actual requirements
- **End-to-End *Systems Thinking* Required**
  - DevOps, maintenance team, education needs when new tech. is used
- ***Trust* is Foundation for Long Term Success**
  - Establish early and sustain it (needed for critical project situations)
  - Transfer knowledge (to client, to peers), do not hide it

* Inspired by Gregor Hohpe's Beliefs (ECSA 2014 keynote)

© Olaf Zimmermann, 2017.

Themen (prüfungsrelevanter Stoff)



- **Vorlesung und Übung sowie Leseaufträge Selbststudium**
  - Foliensätze (13)
  - Fact Sheets (18)
  - Übungsblätter (13)
  - Wiederholungsfragen (146)
  - Musterlösungen (12 bisher)

- **Fragestil und Aufgabenbeispiele:**
  - Übung 14 (diese Woche)
  - Bespielklausuren aus früheren Semestern
  - *Achtung:* Themen und Beispiele nicht identisch!

© Olaf Zimmermann, 2017.

Feedback (Both Ways)

- **Fragestellung:**
  - Was war gut, was sollte so bleiben?
  - Was sollte im nächsten Jahr anders gemacht werden (warum, wie)?
  - Some rules:
    - https://stat.ethz.ch/education/semesters/ss2014/seminar/index/edit/feedback_rules.pdf
- **Weitere Kommentare?**
- **Diskussion**

© Olaf Zimmermann, 2017.

Learning Objectives Revisited

- **IT architect is not an entry-level position typically**
  - but many (if not all) developers are involved with the architecture of a system – and they interact with architects
- **Therefore the module introduced core concepts…**
  - such as practices and patterns
- **… in the context of agile software development.**
  - starting from a developer point of view
- **So that you can ask/answer hard questions about software designs ☺**

Also see related section(s) in online module description:
http://studien.hsr.ch/allModules/public/28236_M_AppArch.html

© Olaf Zimmermann, 2017.

Die letzte Folie (in 2017)…

- **Viel Erfolg bei der Prüfungsvorbereitung (und dann in der Prüfung)!**
  - Termin in der Beratungswoche:
    - 10. 1. 2018, 10:10, im VL-Raum
    - Vorher per email eingehende Fragen werden dann besprochen
  - Prüfung:
    - Siehe unterricht.ch
    - Open Book (aber keine elektronischen Hilfsmittel, keine alten Prüfungen)
- **Literaturhinweise:**
  - Siehe Vorlesungsfolien der Wochen 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14
  - Siehe Fact Sheets
  - Siehe Übungsblätter und Musterlösungen dazu
  - Siehe Wiederholungsfragen im Folder `5-exam-preparation` auf Skriptserver
  - Siehe Beispielprüfungen (z.T. andere Inhalte!)

© Olaf Zimmermann, 2017.