Institut für Software

ARCHITEKTURENTSCHEIDUNGEN IM WORKFLOW-DESIGN

Softwareforen Leipzig, Arbeitsgruppe Softwarearchitekturen
8. Arbeitstreffen:
«Softwarearchitekturen für Workflow-Management»

IFS INSTITUTE FOR SOFTWARE

Prof. Dr. Olaf Zimmermann
Distinguished (Chief/Lead) IT Architect, The Open Group
ozimmerm@hsr.ch

Leipzig, 20. November 2014

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Aufbau und Inhalte dieser Session

- **Gliederung in drei Einheiten (Abschnitte):**
  - Vortrag, moderierte Gruppenarbeit, Reflektion und Ausblick

- **Vortragsthemen:**
  - Motivation: Architekturentscheidungen in einer Fallstudie aus der Praxis (Order Management)
  - Konzeptionelle Workflow-Designentscheidungen:
    - Wahl von Patterns, Schichtenbildung, Schnittstellenfragen
  - Technische und organisatorische Architekturentscheidungen:
    - Buy vs. Build, Protokolle und Sprachen, Transaktionsgrenzen
  - Leichtgewichtige Templates zur Entscheidungsdokumentation und Werkzeugunterstützung
- **Gruppenarbeit:**
  - Ihre Architekturentscheidungen (aus Workflow-Projekten)

Who am I?

- **Research & Development und Professional Services ab 1994**
  - em. IBM Executive IT Architect (& certified by The Open Group)
    - Systems & Network Management, J2EE, Enterprise Application Integration/SOA
  - em. ABB Senior Principal Scientist
    - Enterprise Architecture Management/Legacy System Modernization/Remoting
- **Auswahl Industrieprojekte und Coachings**
  - Produktentwicklung und IT Consulting für Middleware, SOA, Informationssysteme (Banken IT, Telekommunikationsbranche), SE-Tools
  - Tutorials: UNIX/RDBMS, OOP/C++/J2EE, MDSE/MDA, Web Services/XML
- **Schwerpunkt @ HSR FHO: Entwurf verteilter Systeme**
  - Cloud Computing, Web Application Development & Integration (Runtime)
  - Modellgetriebene Entwicklung, Architekturentscheidungen (Build Time)

What is SOA?

No single definition – “SOA is different things to different people”

- A *set of services* that a business wants to expose to their customers and partners, or other portions of the organization.

Business Domain Analyst

- An architectural style which requires a *service provider*, a *service requestor* (consumer) and a *service contract* (a.k.a. client/server).
- A set of architectural patterns such as *enterprise service bus*, *service composition*, and *service registry*, promoting principles such as *modularity*, *layering*, and *loose coupling* to achieve design goals such as separation of concerns, reuse, and flexibility.

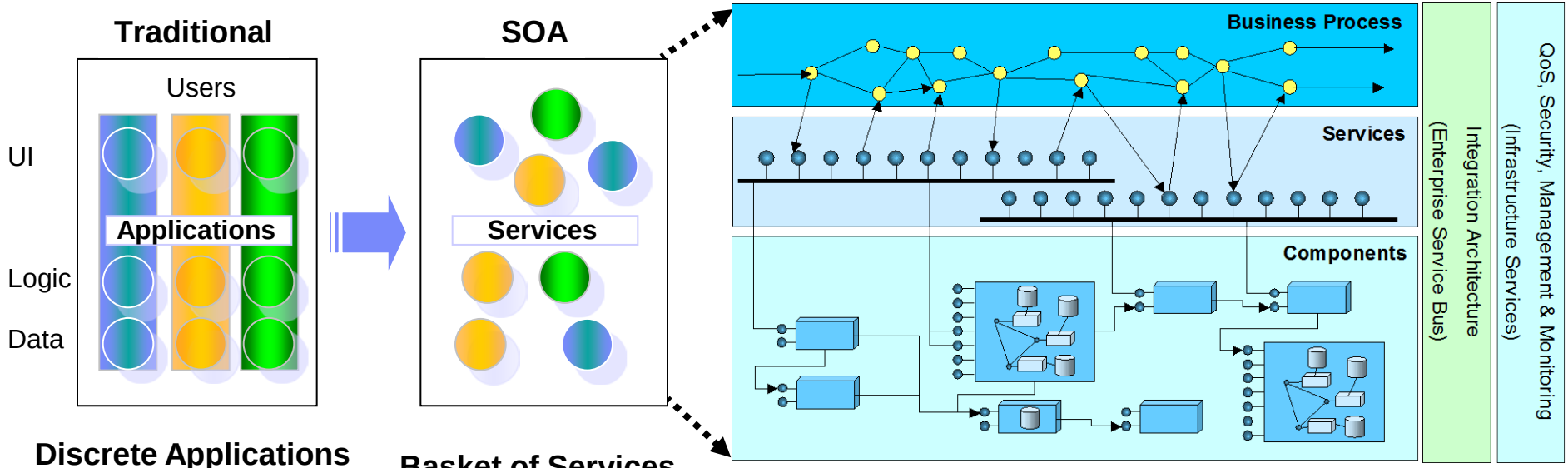
IT Architect

- A *programming and deployment model* realized by standards, tools and technologies such as Web services and Service Component Architecture (SCA).

Developer, Administrator

Reference: Adapted from IBM SSS (SOA Reference Architecture)

Partitioning into Components and Services (SOA Example)



Discrete Applications (Two or Three Tiers)

Basket of Services

Layering based on IBM SOA reference architecture

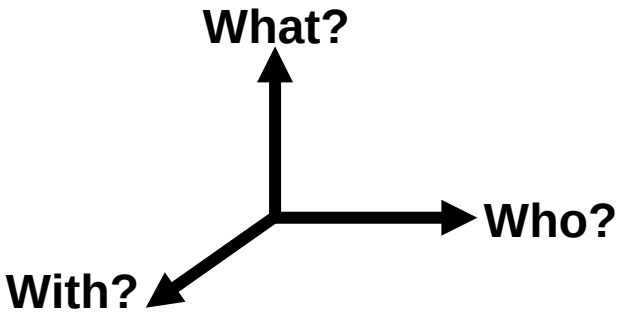
Example:
An insurance company uses three SAP R/3, MS Visual Basic, and COBOL applications to manage customer information, check for fraud, and calculate payments. The user interfaces (UIs) are the only access points.

A multi-step, multi-user business process for claim handling, executing in IBM WebSphere, is supposed to reuse the functions in the existing applications. How to integrate the new business process with the three legacy applications in a flexible, secure, and reliable way?

Reference: O. Zimmermann, SOA and Web Services Tutorials, OOPSLA 2005 - 2008

Workflow Management – Essentials

- **Workflow and service composition form upper part of business logic layer (domain layer) in layered enterprise application**
  - Programming in the large vs. programming in the small
  - Workflow not to be confused with integration flows and or HTML page flows
- **Fundamental workflow concepts**
  - Process instance, process variables
  - Control flow vs. data flow
  - Human tasks, staff assignments
- **Key BPMN constructs**
  - Start, stop events
  - Tasks (human user, service)
  - Gateways
  - Pools and lanes
  - Transactions and compensation

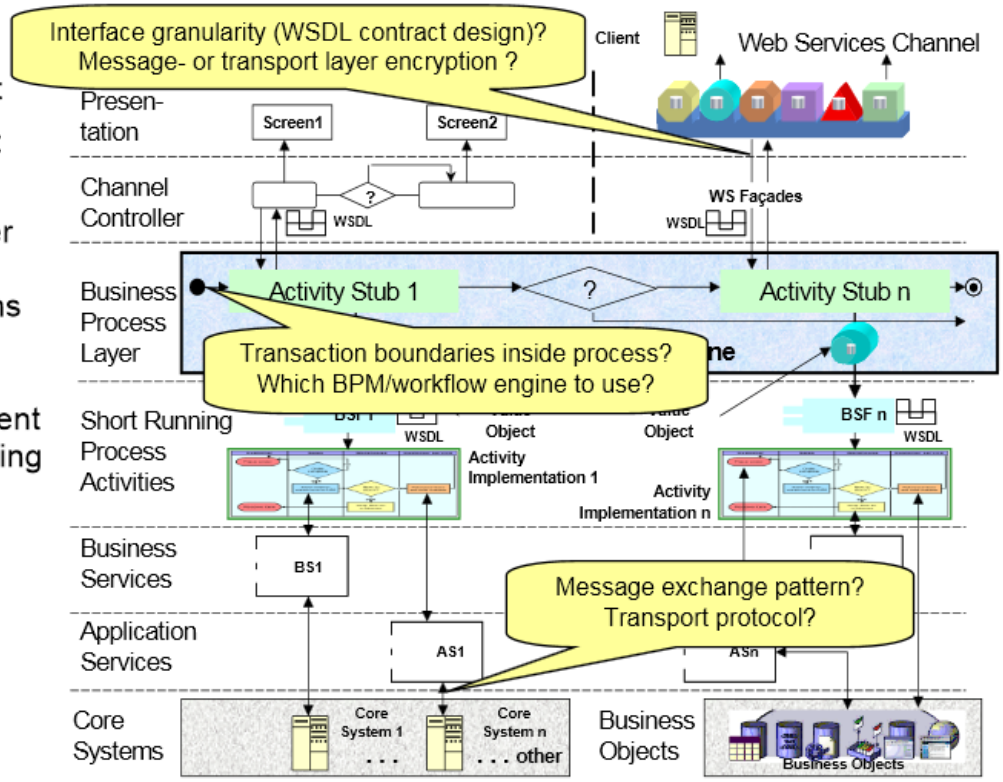


Telco Case Study (with selected Architectural Decisions)

Reference: IBM, ECOWS 2007

Multi-Channel Order Management SOA in the Telecommunications Industry (in production since Q1/2005) [OOPSLA 2005]

- **Functional domain**
  - Order entry management
  - Two business processes: new customer, relocation
  - Main SOA drivers: deeper automation grade, share services between domains
- **Service design**
  - Top-down from requirement and bottom-up from existing wholesaler systems
  - Recurring architectural decisions:
    - Protocol choices
    - Transactionality
    - Security policies
    - Interface granularity



What are Architectural Decisions (ADs)? Why Care?

Reference:: IBM, SATURN 2010

- **"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."

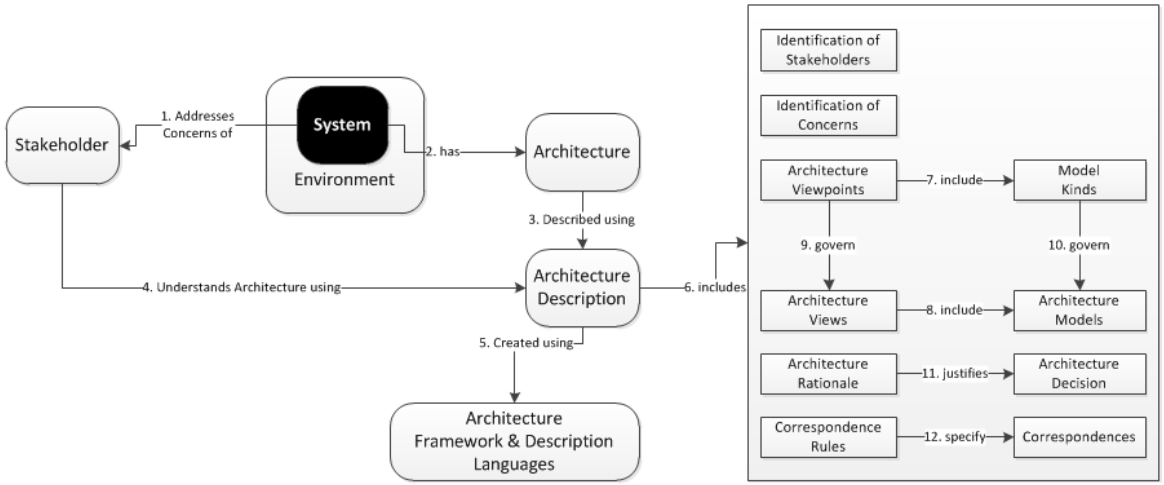
- **From IBM UMF work product description ART 0513 (since 1998):**

  "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"

From ADs to Architectural Knowledge Management (AKM)

- **IEC/IEEE/ISO 42010:2011 standard for architecture description**
  - Rationale as first class entity in architecture documentation



Reference: http://enterprise-strategy-architecture.blogspot.ch/2011/11/understanding-isoiecieee-420102011.html

- **Active research community investigating architectural decisions**
  - E.g. SOAD project: active, guiding role for recurring architectural decisions
- **See SEI SATURN 2013 BoD session report regarding state of the art**

AD Capture for Documentation Purposes (in Enterprise Architect)

Reference: ABB, SATURN 2012

Unique identifier: AD-01, **Status:** approved, **Owner:** cholzim

Link to system concerns
See architectural principles stated on the enterprise level: IT security requirements

Rationale
In the context of the historian component, facing the data privacy requirements specified in the corporate security guidelines, we decided to encrypt the persistent storage to achieve confidentiality.

Constraints and assumptions:
Performance is good enough, certificate management can be handled.

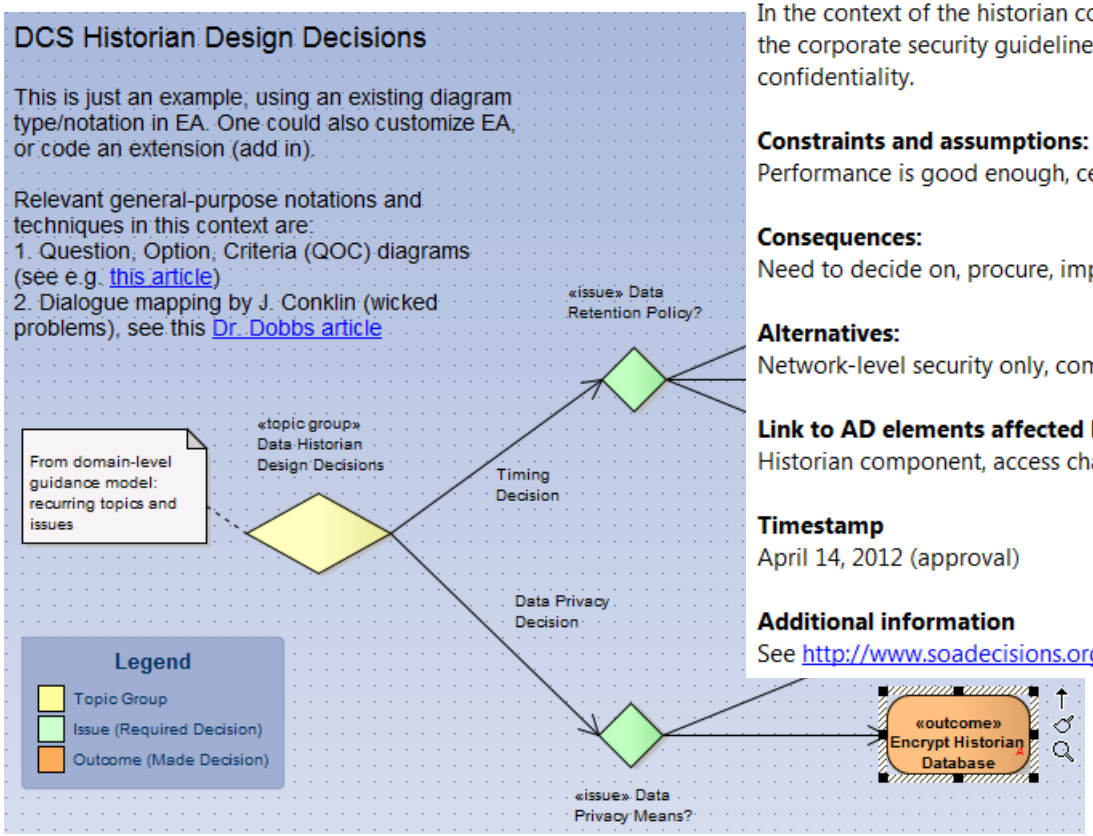
Consequences:
Need to decide on, procure, implement some crypto library and/or hardware (TBD)

Alternatives:
Network-level security only, combined with role-based access control

Link to AD elements affected by the decision:
Historian component, access channel, security subsystem

Timestamp
April 14, 2012 (approval)

Additional information
See http://www.soadecisions.org



AD about Integration Style (IBM UMF Template for Decision Log)

Reference IBM, SATURN 2010

Subject Area	Process and service layer design	**Topic**	Integration
Name	Integration Style	**AD ID**	2
Decision Made	We decided for RPC and the Messaging pattern (Enterprise Integration Patterns)		
Issue or Problem	How should process activities and underlying services communicate?		
Assumptions	Process model and NFRs/QA requirements are valid and stable		
Motivation	If logical layers are physically distributed, they must be integrated.		
Alternatives	File transfer, shared database, no physical distribution (local calls)		
Justification	This is an inherently synchronous scenario: VSP users as well as internal Telco staff expect immediate responses to their requests. Messaging system will ensure guaranteed delivery and buffer requests to unreliable data sources.		
Implications	Need to select, install, and configure a message-oriented middleware provider.		
Derived Requirements	Many finer grained patterns are now eligible and have to be decided upon: message construction, channel design, message routing, message transformation, system management (see Enterprise Integration Patterns book).		
Related Decisions	Next, we have to decide on one or more integration technologies implementing the selected two integration styles. Many alternatives exist, e.g., Java Message Service (JMS) providers.		

Y-Template (ABB Software Development Improvement Initiative)

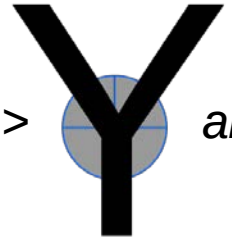
Reference ABB, SATURN 2012

- **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>



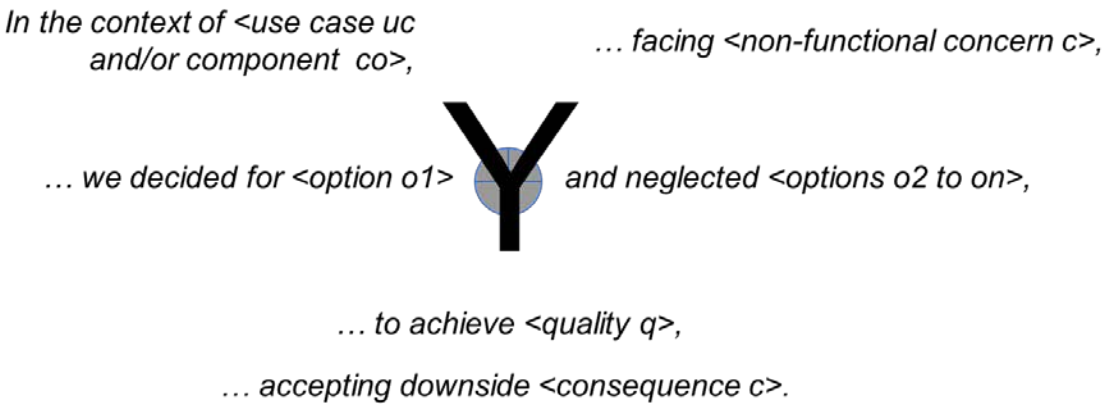
and neglected <options o2 to on>,

… to achieve <quality q>,

… accepting downside <consequence c>.

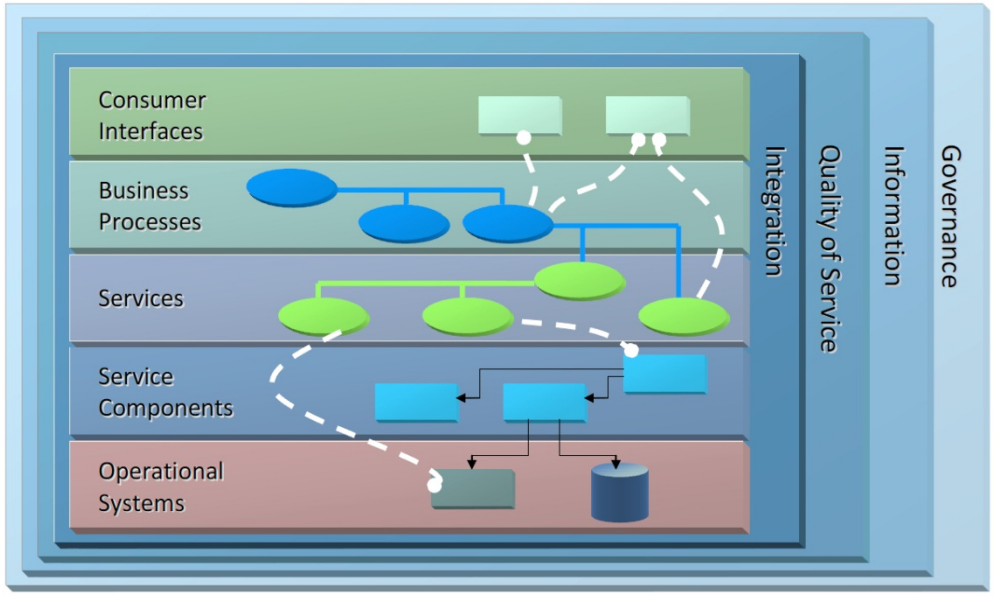
Filled Out Y-Template (Usage Example)

Reference ABB, SATURN 2012



Example: "In the context of data historian access to the archive,
… facing data privacy regulations,
… we decided to encrypt historian database content
(and neglected to not encrypt)
… to achieve confidentiality,
… accepting a negative impact on performance."

Many ADs Recur in Enterprise Application Architectures



Source: SOA Reference Architecture, The Open Group, 2009
https://www.opengroup.org/projects/soa-ref-arch/uploads/40/19713/soa-ra-public-050609.pdf

Decision made: "We decided for pattern/technology/product X to resolve issue Y because of requirement Z"

Observation (Claim):
Many architectural decisions are not specific to a case – they recur

Challenges:
1. SOA literature does not make required decisions explicit
2. Hundreds of decisions to be made
3. Decision making order unclear

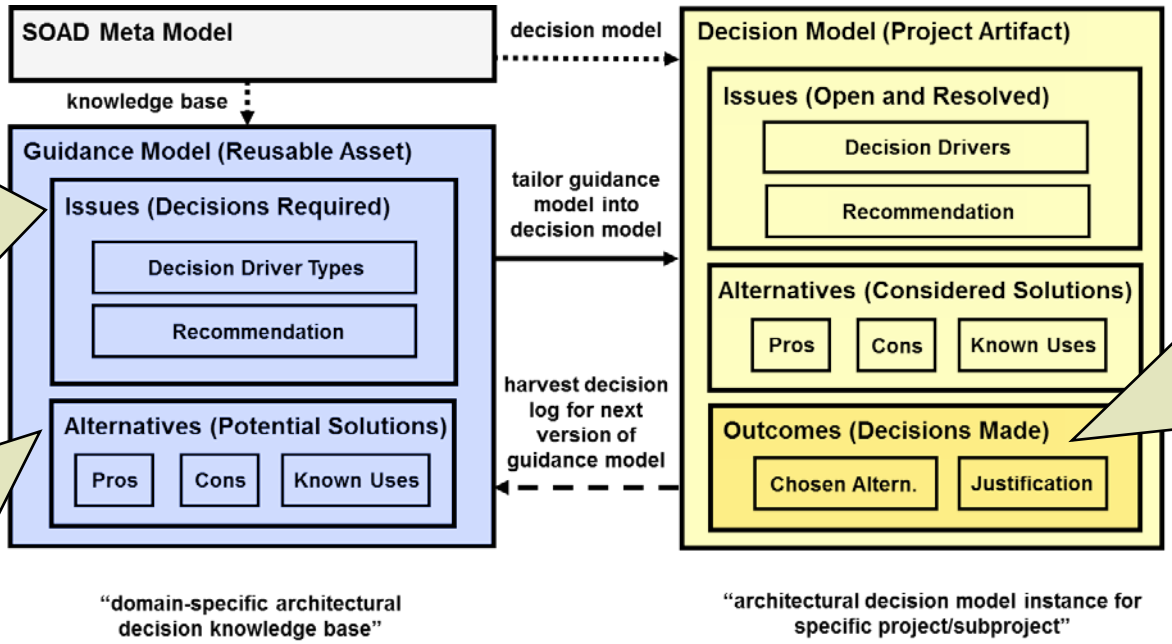
Decision required: "You will have to decide for a pattern/technology/product to resolve issue Y. X is one alternative you may want to consider, Z a decision driver"

[JSS 2009] Zimmermann O., Koehler J., Leymann F., Polley R., Schuster N., *Managing Architectural Decision Models with Dependency Relations, Integrity Constraints, and Production Rules*. Journal of Systems and Software, Elsevier. Volume 82, Issue 8 (2009)

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*."



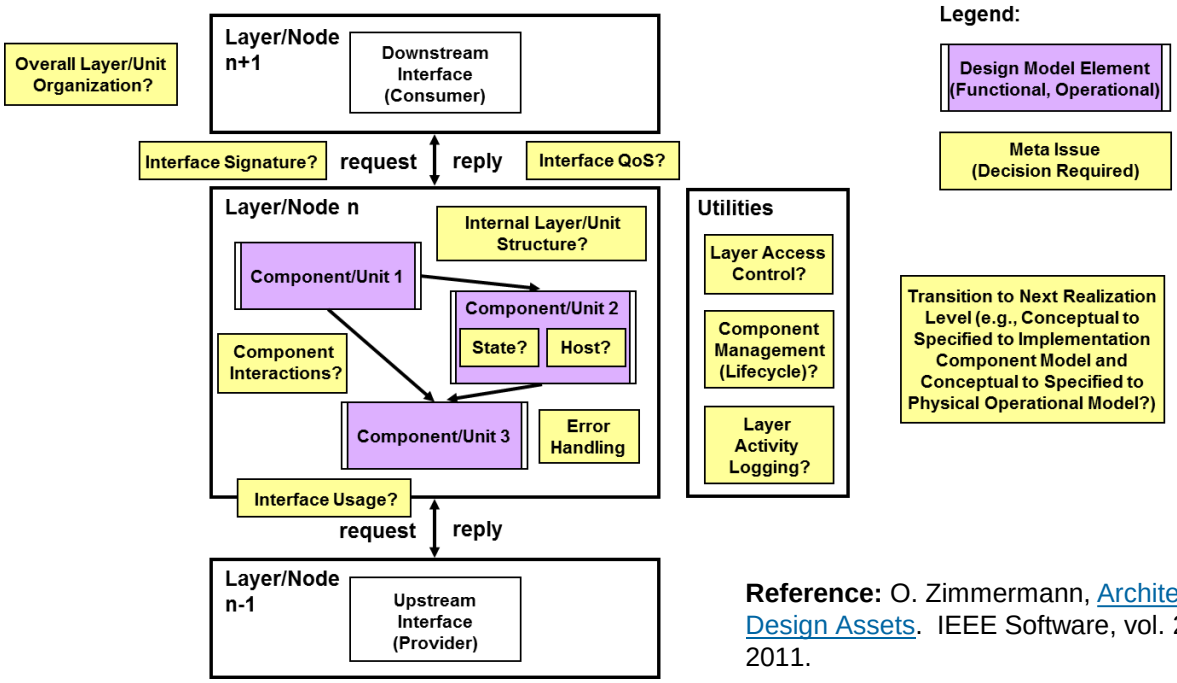
"*Model View Controller (MVC)* is a common architectural pattern to control the Web page flow."

"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

SOAD Project (2006-2011): Issues Recurring in SOA Design

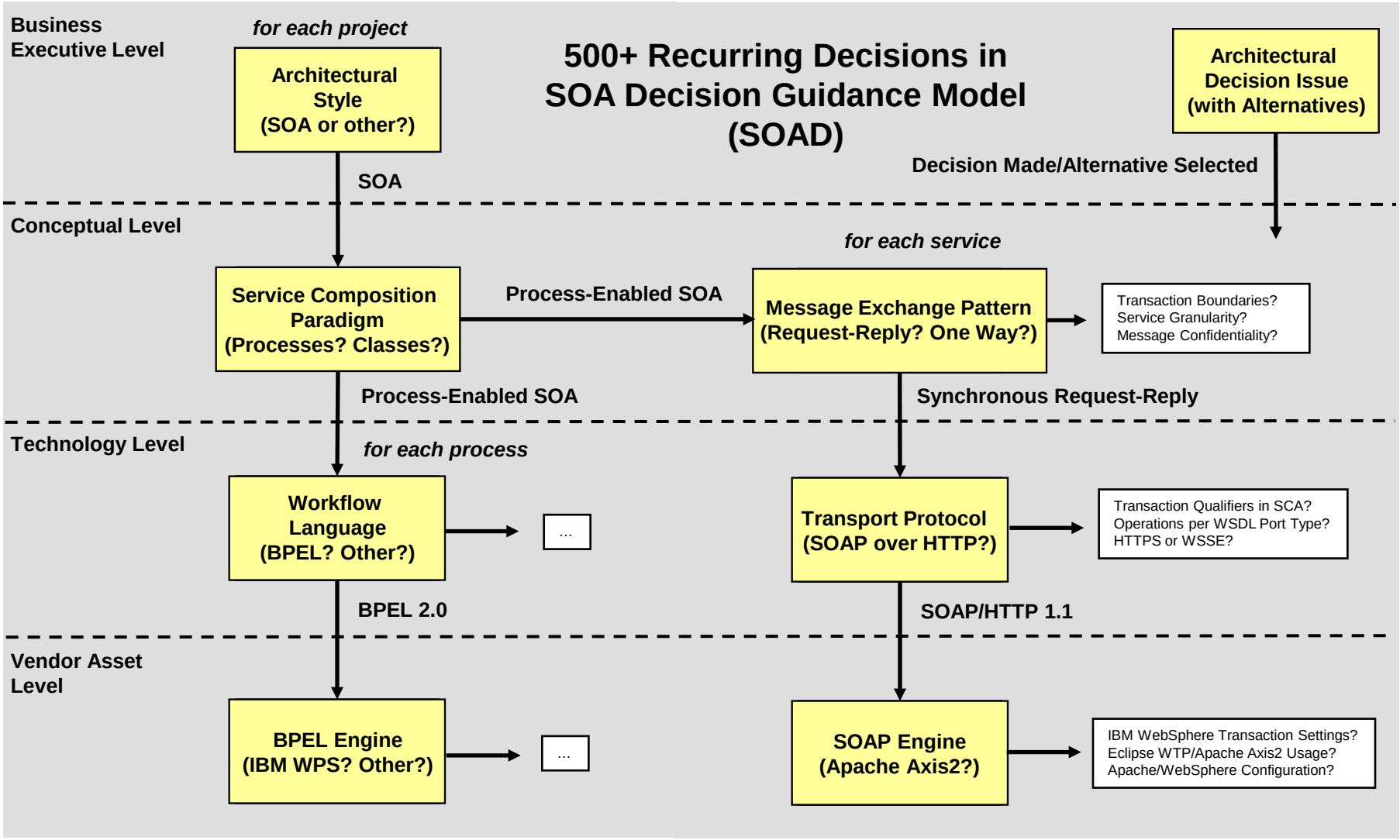
- **Patterns + recurring issues yield guidance models for a domain**
  - (Can be) applied to information system design and information integration
- **Issue catalog organized by layer/node type, by component/connector**



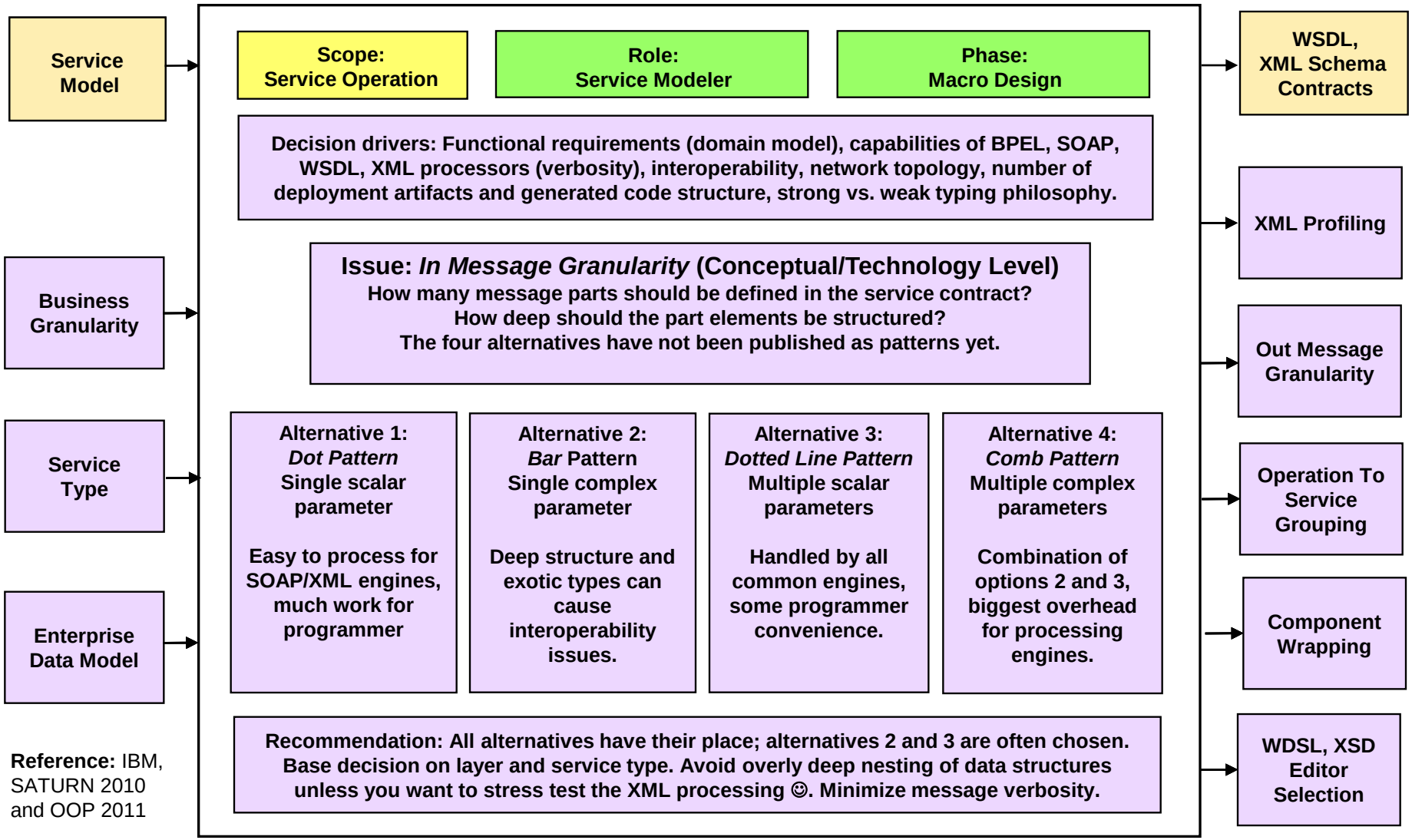
Reference: O. Zimmermann, Architectural Decisions as Reusable Design Assets. IEEE Software, vol. 28, no. 1, pp. 64-69, Jan./Feb. 2011.

Recurring AD Issues Organized into 3+1 Levels of Refinement

Reference: IBM, SATURN 2010 and OOP 2011

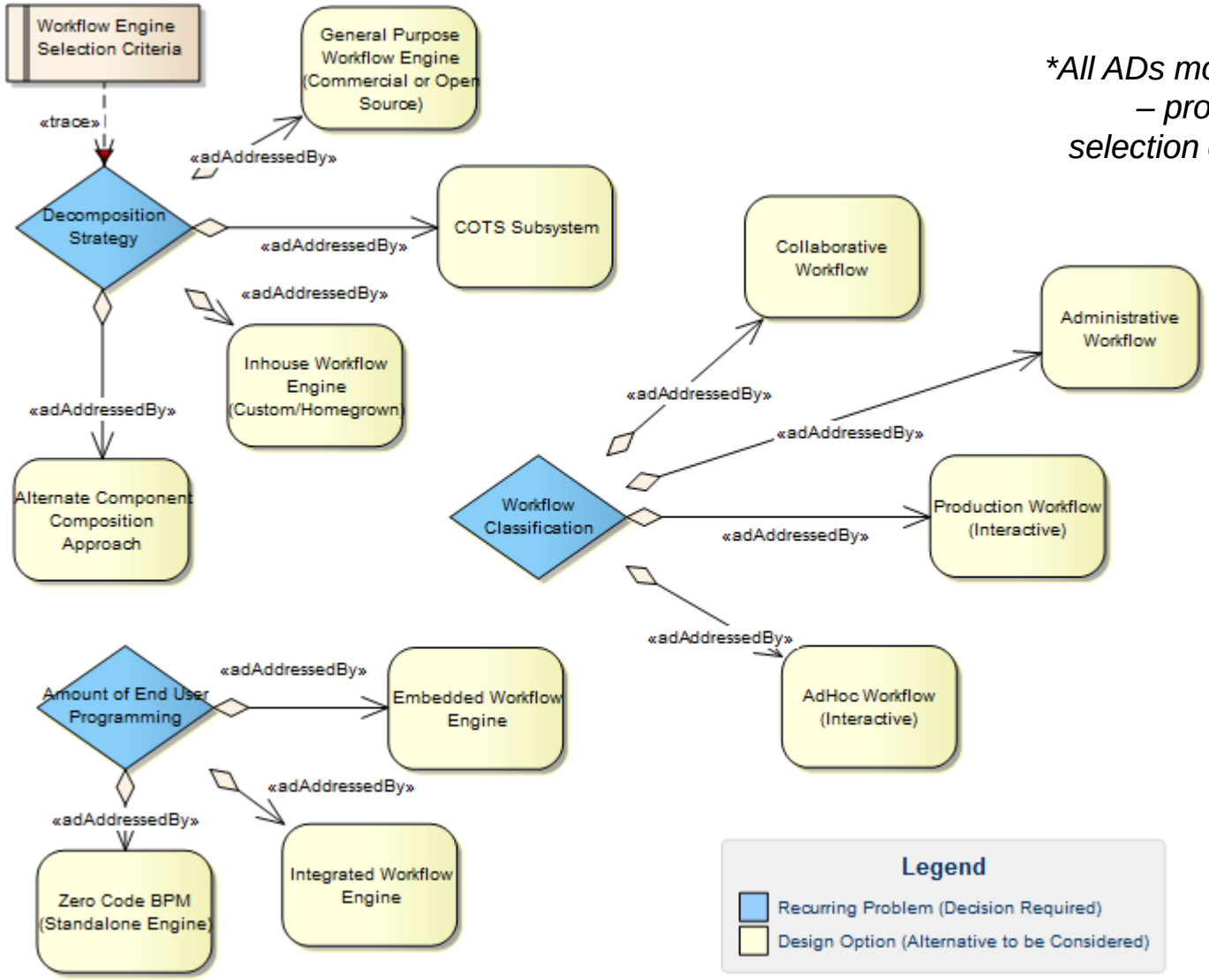


Sample AD Issue – Addressing Service Granularity Topic



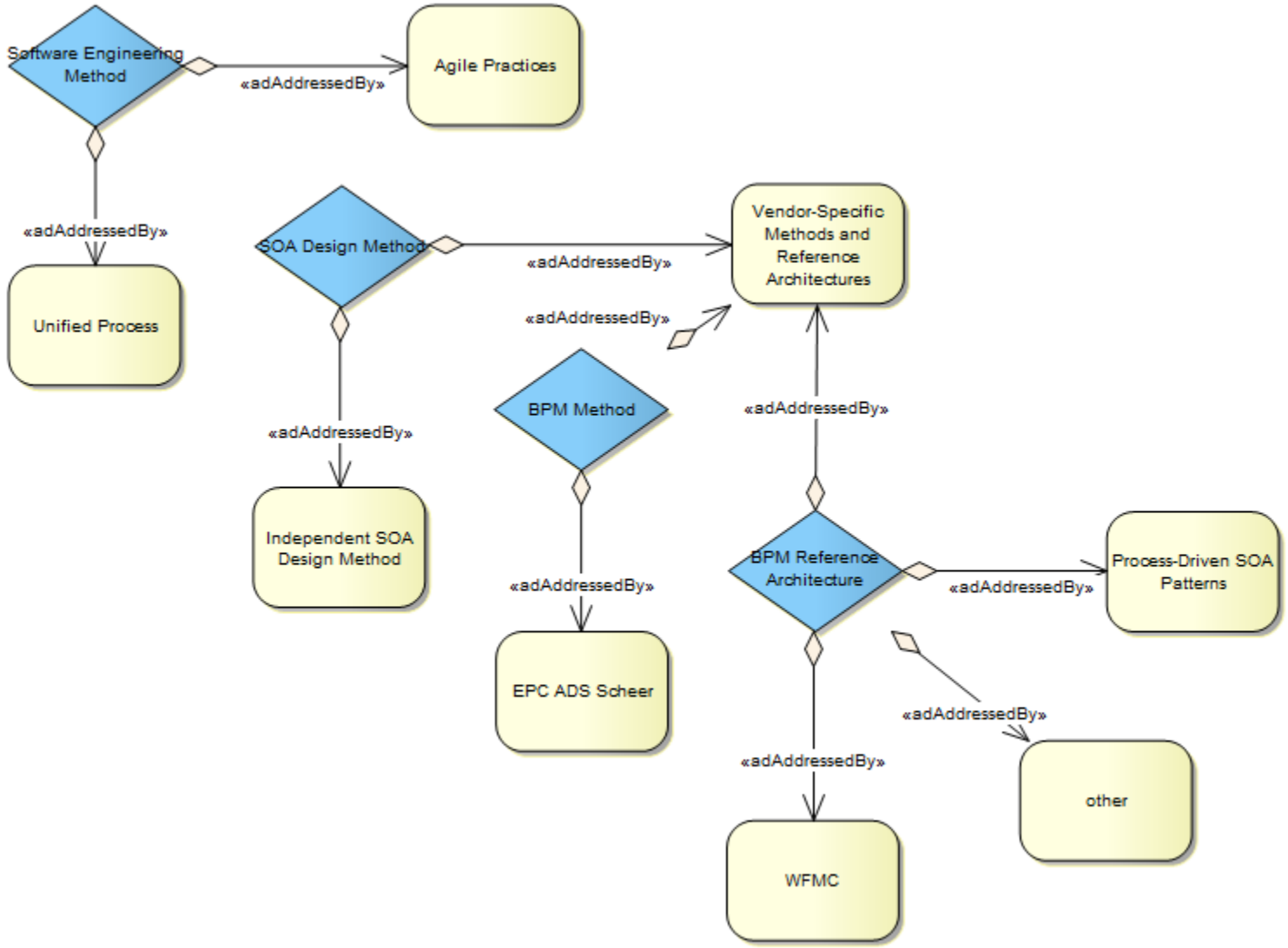
Reference: IBM, SATURN 2010 and OOP 2011

Workflow-ADs (1/9): Scenario Classification*

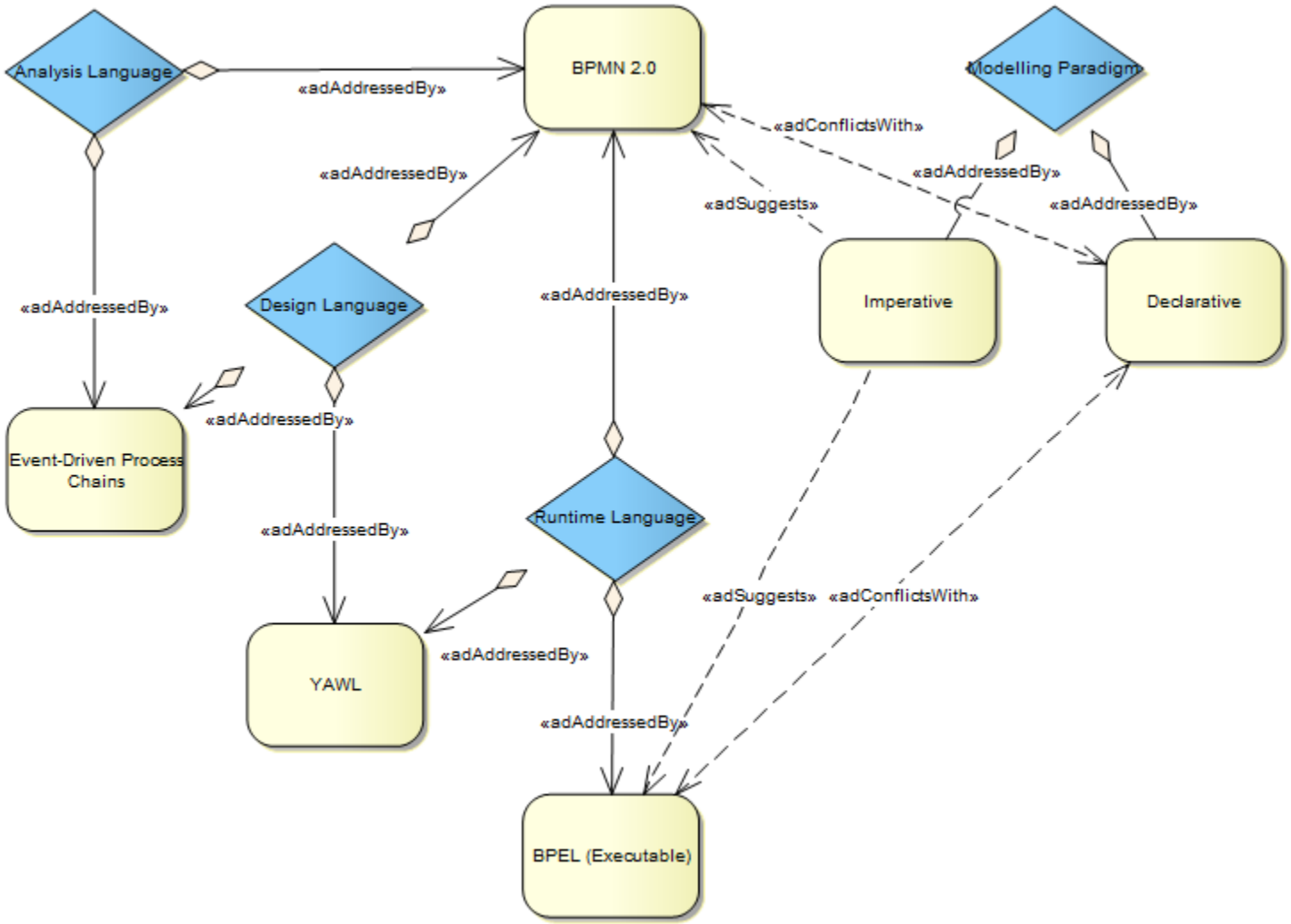


**All ADs modelled in ADMentor tool – problem statements, option selection criteria, references, tags*

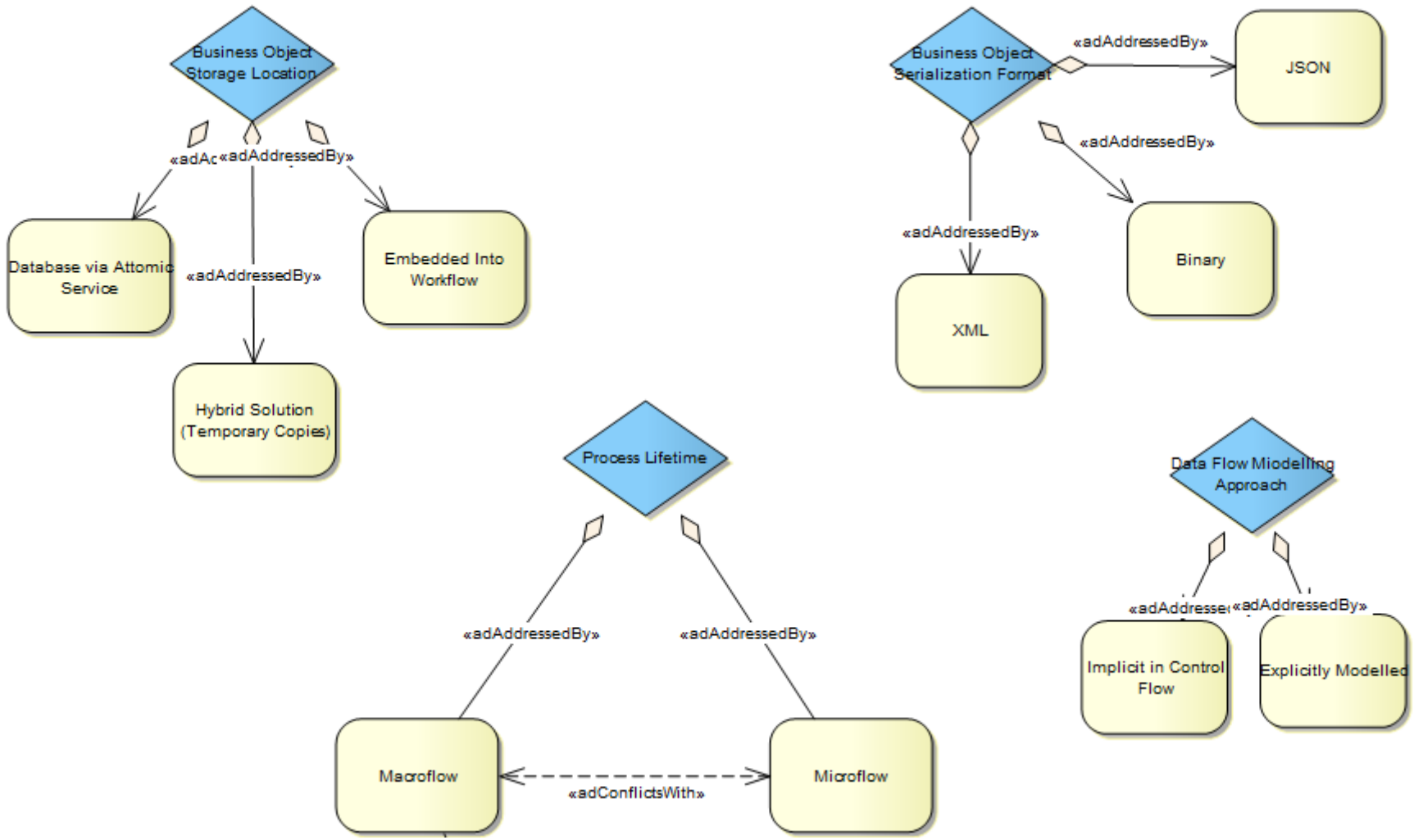
Workflow-ADs (2/9): Methods, Reference Models



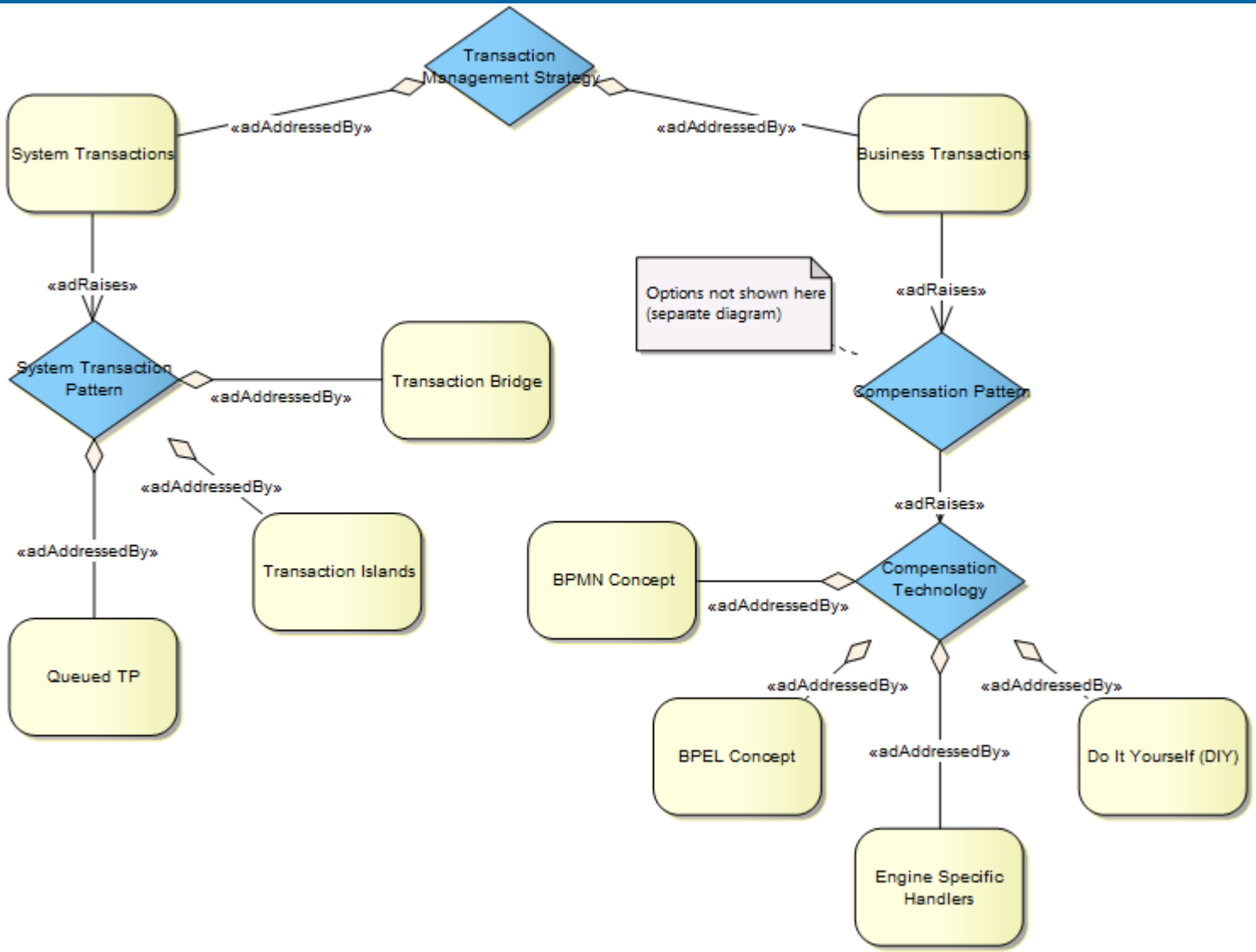
Workflow-ADs (3/9): Notation



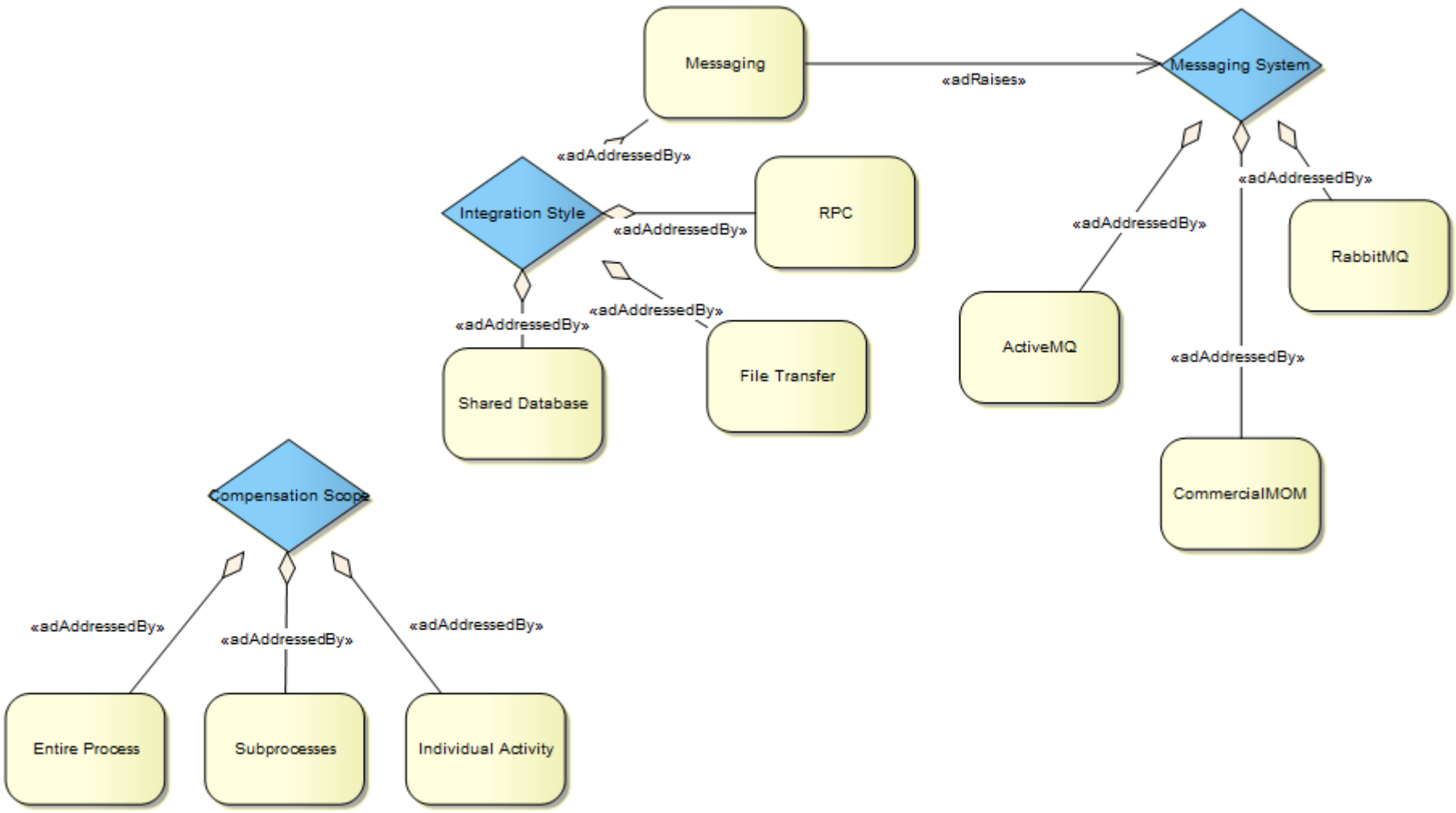
Workflow-ADs (4/9): Overall Process Design and Data Flow



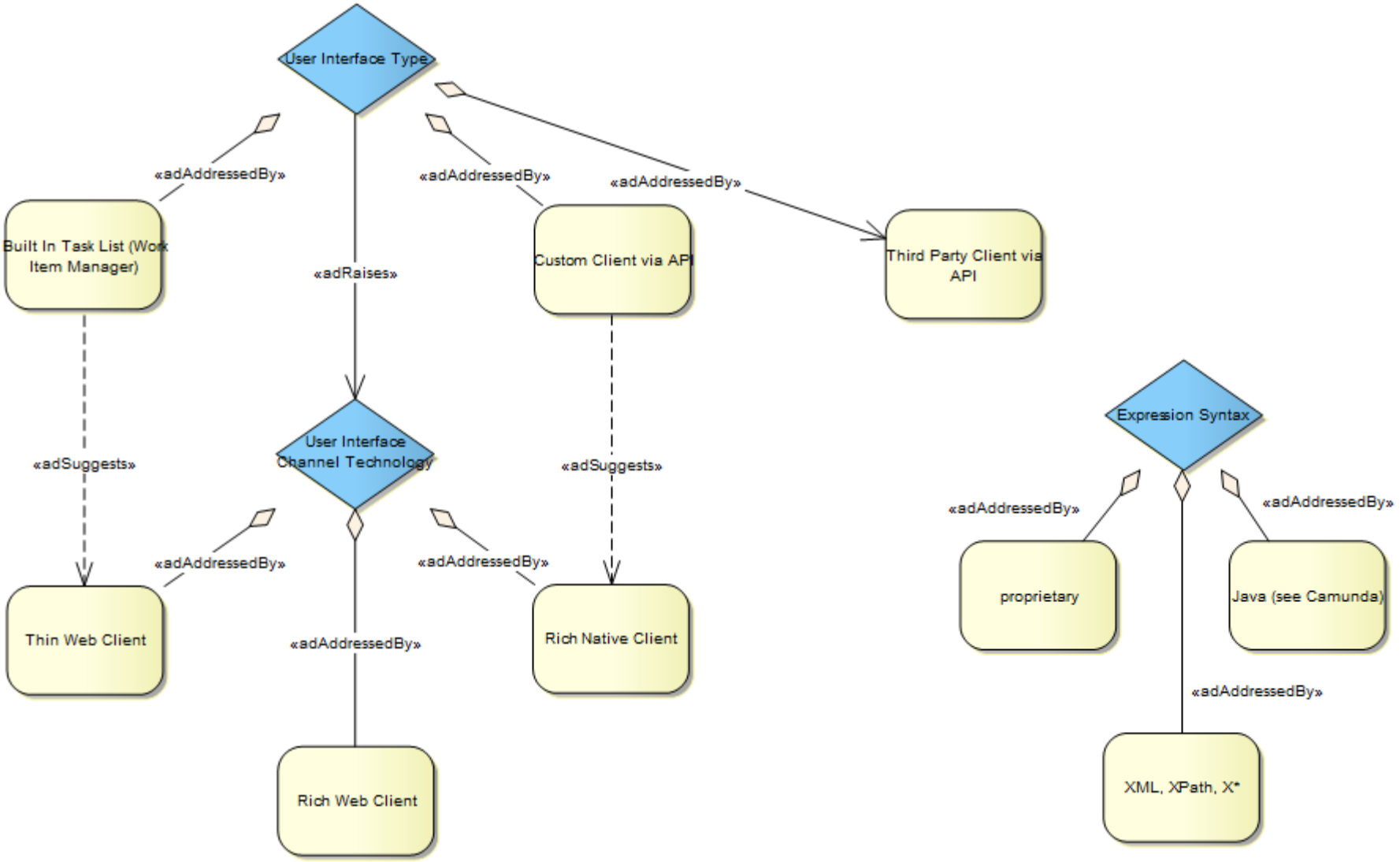
Workflow-ADs (5/9): Transaction Management



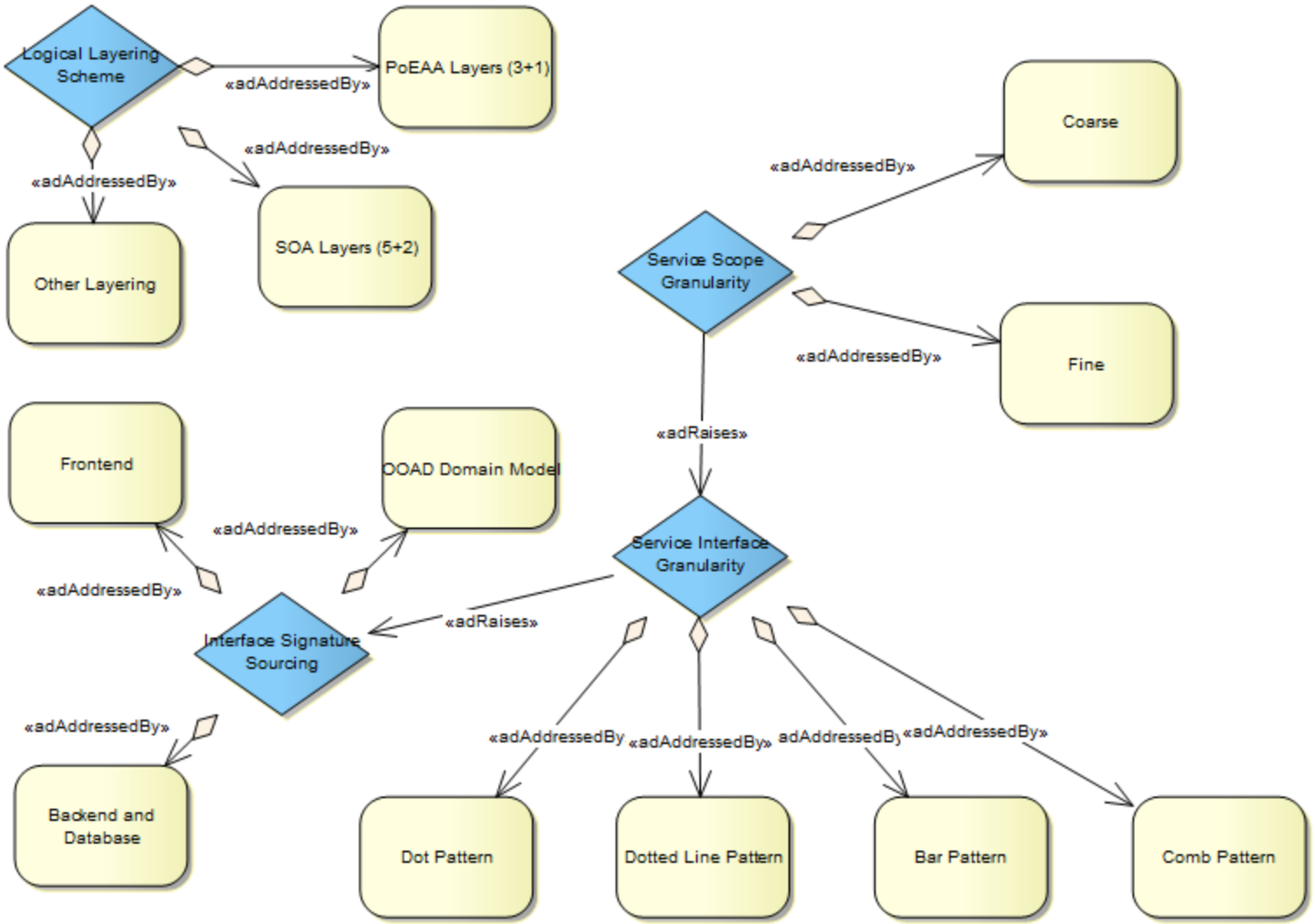
Workflow-ADs (6/9): Compensation, Integration



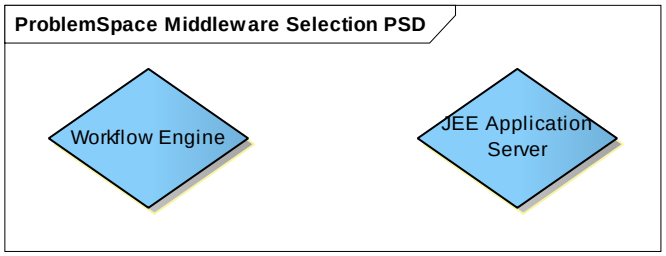
Workflow-ADs (7/9): Presentation Layer and Flow Control



Workflow-ADs (8/9): SOA and Interface Design



Workflow ADs (9/9): Detailed AD Descriptions (Two Examples)*



Notes

A recurring architectural decision in every nontrivial enterprise application development project concerns the *decomposition strategy* that is followed:

- What is the overall approach to software modularization?
- How is the business logic (layer) organized?
- Who is responsible for keeping track and persisting application state (for each instance of a business process)?

Some indicators for introducing workflow concepts and technologies are:

- Long running business processes (process lifetimes)
- Coordination of activities (use cases, user stories) by multiple actors and actor roles per "process" instance; mix of human activities and automated ones
- Complex processing logic that requires undo operations (a.k.a. compensation) to preserve "process" and resource integrity that cannot be satisfied with ACID-style system transactions
- Frequent changes in the business activity sequencing and concurrency patterns (on business level)
- Challenging audit requirements (process monitoring and mining)

Notes

Checklist "why JEE":

- Container patterns:: Inversion of Control, Dependency Injection, beans
- Transaction Management
- Application Security
- Clustering (horizontal scaling)
- Resource pooling (JDBC data sources, threads), request throttling
- JEE APIs and services: like JMS, JDBC, JAX-WS/JAX-RS (traditional and newer ones)
- Application Management (standardized deployment, monitoring, etc., both user interface and Command Line Interface (CLI); JMX instrumentation
- Conceptual integrity - e.g. same approach to adaptive capacity management (scalability management)
- Clear, standardized separation of application code and infrastructure code
- to be continued (tbc)

Wanted: "Aussteigerpogramm" (rightsizing to achieve adequate quality characteristics):

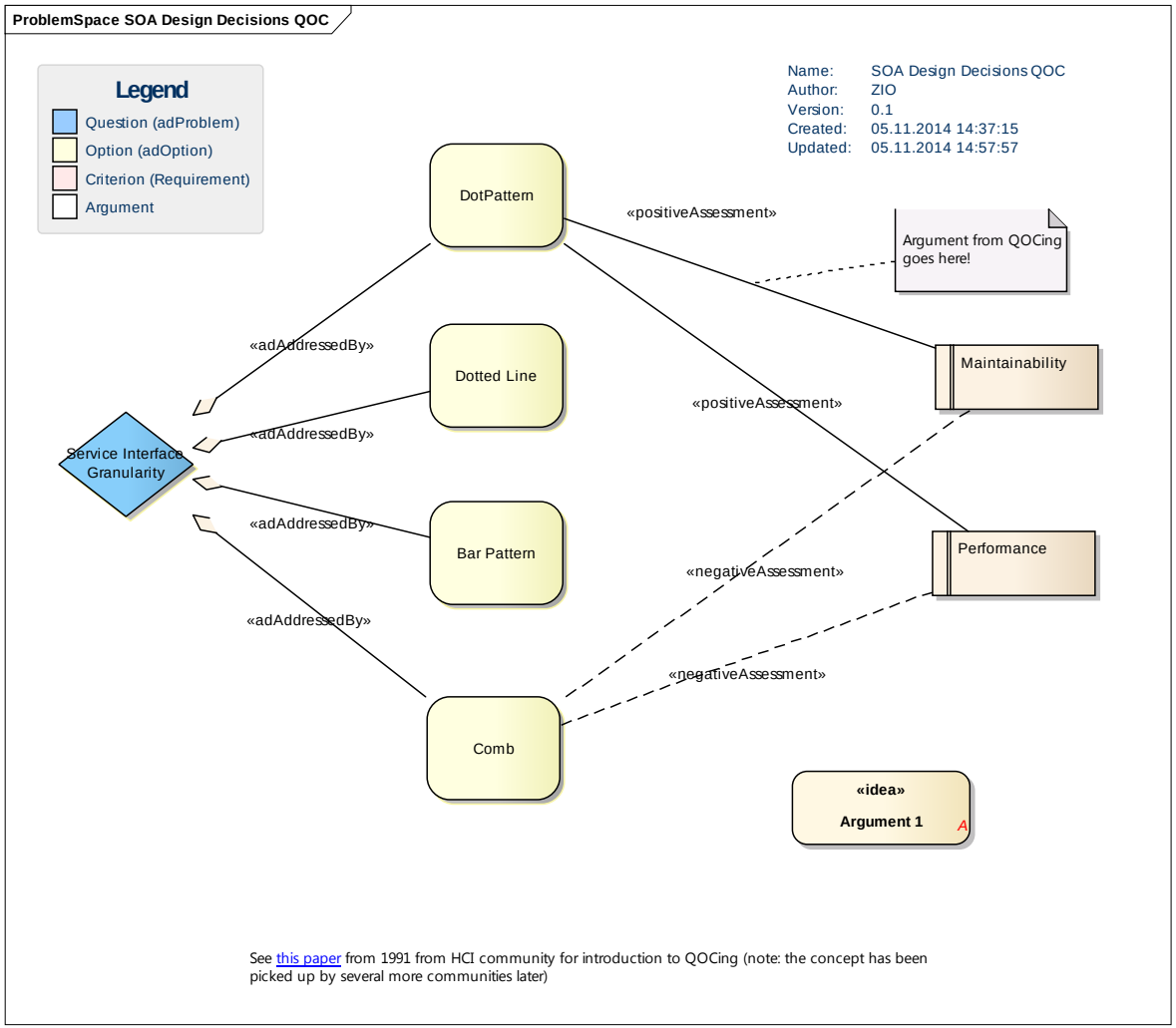
1. Confirm continued need for any feature
2. Find alternate solutions if still required
3. Criteria. cost, skills, footprint/efficiency, SLA satisfaction, complexity, error handling (fault tolerance in the broadest sense of the word)

**All ADs modelled in ADMentor tool – problem statements, option selection criteria, references, tags*

Question, Option, Criteria (QOC) Diagram (in ADMentor)

- **Questions**
  - Issues
- **Options**
  - Alternatives
- **Criteria**
  - Drivers
- **Arguments**
  - Rationale

Qualitative analysis, not quantitative!



… jetzt sind Sie an der Reihe (Gruppenarbeit, 20 min)

1. **Welche SWA-WFM Themen interessieren Sie besonders?**
   - Lassen Sich diese Themen als wiederkehrende Architectural Decisions (ADs) formulieren, evtl. bereits mit Optionen?
   - Welche ADs haben Sie in den letzten Wochen im Projekt identifiziert, diskutiert, verabschiedet, gereviewed, umentscheiden… (und warum)?
   - Welche ADs stehen demnächst an?
2. **Welchen Charakter haben Ihre ADs aus Schritt 1?**
   - Wer sind die Entscheidungsträger und Betroffenen (engl. Stakeholder)?
   - Welche Kriterien werden zugrunde gelegt (engl. Criteria, Concerns)?
   - Wie wird entschieden und begründet?
   - Wie (nachhaltig) werden die ADs dokumentiert (Bsp. Sitzungsprotokoll, Wiki)?
   - Wie werden umgesetzt und wie wird die Umsetzung nachverfolgt?

Ergebnis: Bulletliste oder 2-3 ausgefüllte Templates (42010, UMF, Y; arc42)

- Hilfestellung für z.B. zu guten Begründungen: siehe Anhang (Handouts)

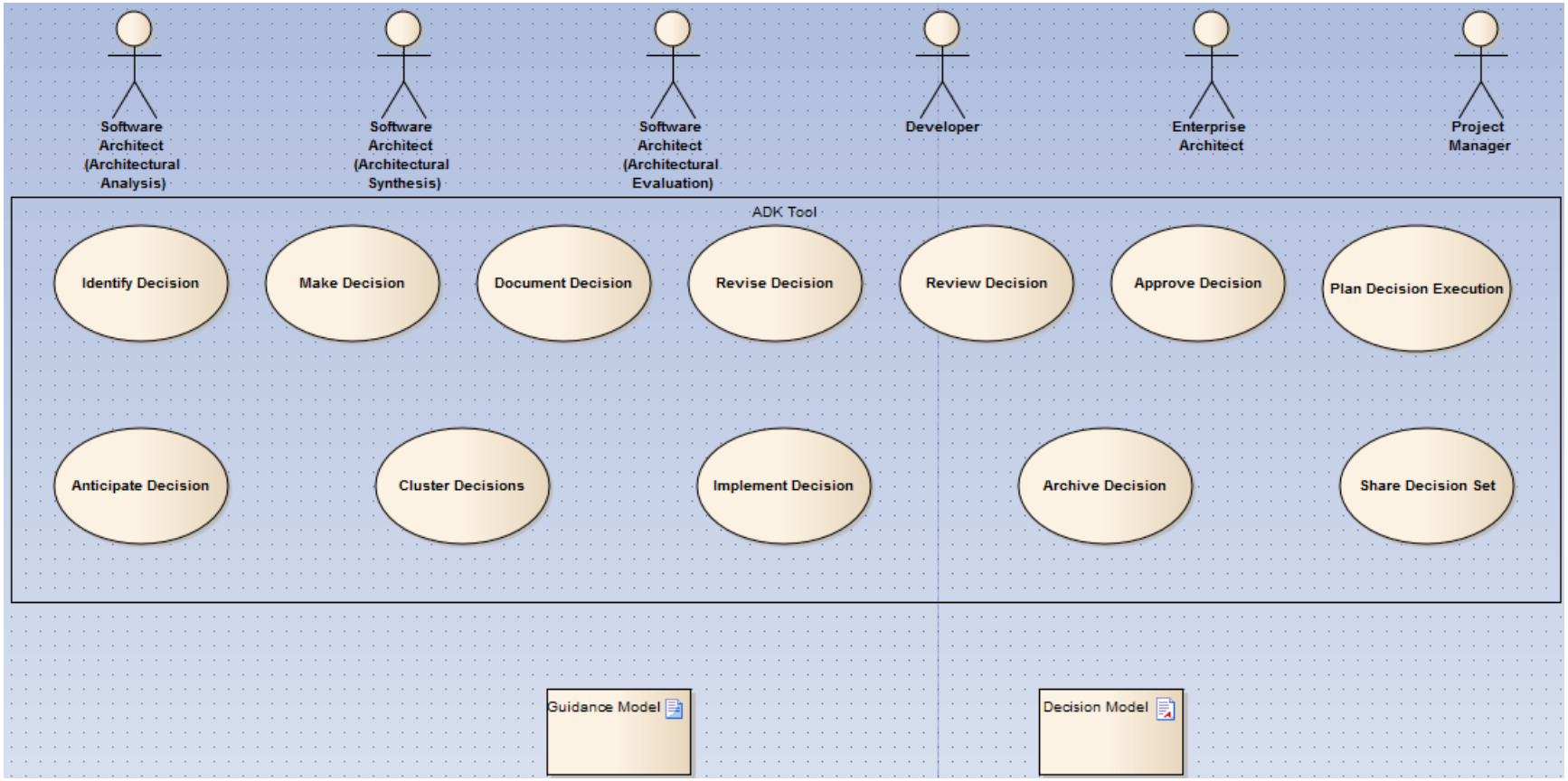
Reflektion und Praktische Tipps zu Architekturentscheidungen

- **Entscheidungen aktiv identifizieren**
  - ggfs. mit Reuse a la SOAD/ARC und Tool wie ADMentor
- **Entscheidungen priorisieren und bewusst treffen**
  - “Worst First” vs. “Defer to Last Responsible Moment” (lean/agile principle)
- **Tradeoffs abwägen und managen**
  - Designmethoden z.B. SEI ADD und Evaluationsmethoden z.B. SEI ATAM

- **Entscheidungen dokumentieren**
  - IEEE 42010 Template oder IBM-Template oder Y-Statements
- **Entscheidungsumsetzung einfordern und begleiten**
  - Coaching, Code Reviews
  - Architectural Evidence und Architectural Templates in Code

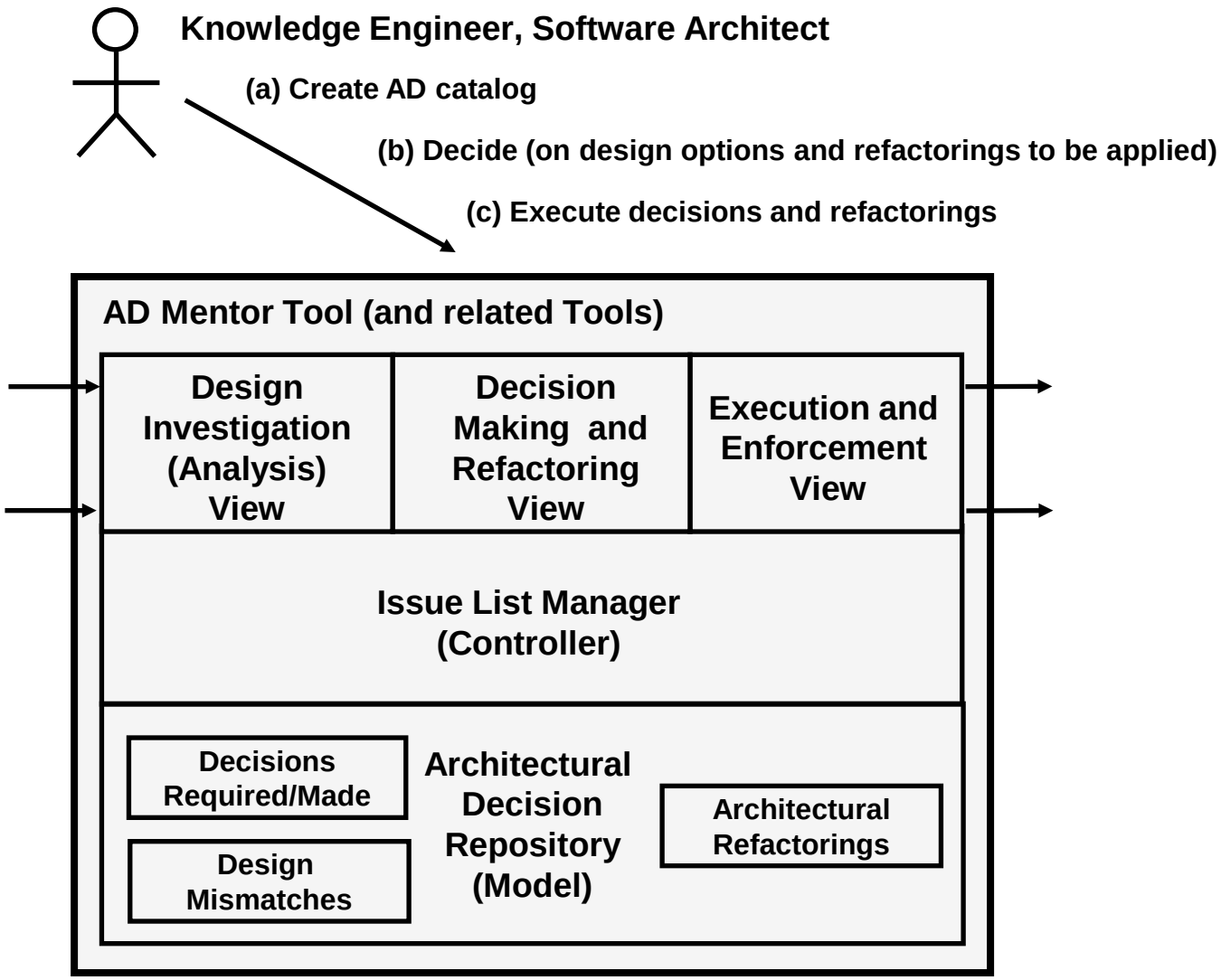
Vision: Integrated Decision/Design Tool Chain

Reference: O. Zimmermann, Making Architectural Knowledge Sustainable, IEEE Software Talk at SATURN 2012



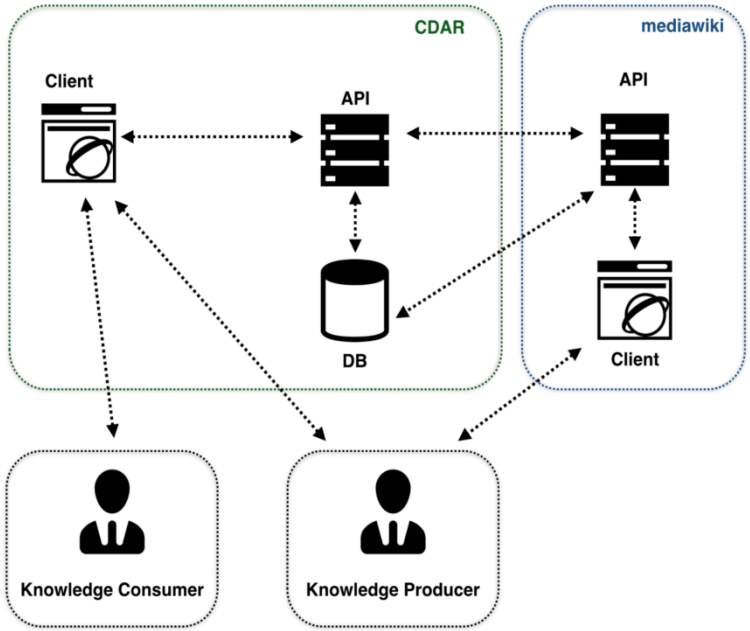
- **Note: Tool builders should justify capture their design decisions (like any architect)… and share them with their collaborators!**

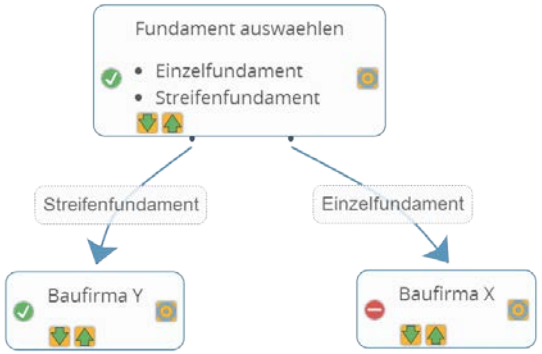
Towards Tool Support for Architectural Refactoring



Completed Thesis Project (HSR FHO): CDAR Tool

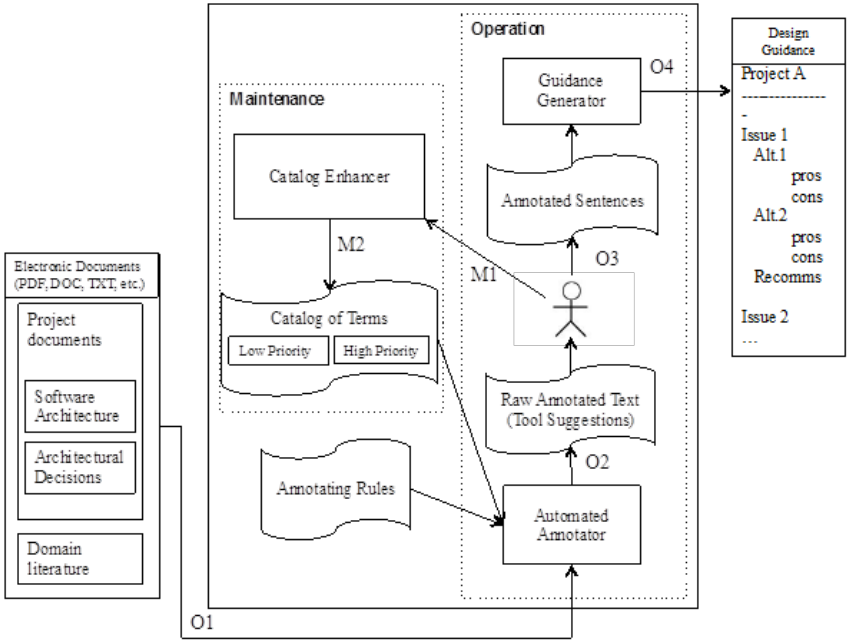
- **Collaborative Decision Management and Architectural Refactoring (CDAR) Tool**
  - RESTful integration of Browser user interface/workflow engine with MediaWiki (the wiki engine behind Wikipedia) via semantic links





Ongoing Research (NTNU): SADGE

- **Joint work with NTNU Trondheim (M. Anvaari, PhD candidate)**
- **Goal: Investigate and apply (extend?) big data techniques and tools (information retrieval, natural language processing) to AD domain (e.g. GATE, ANNIE)**
  - Look for keywords, suggest text passages with high reuse potential to knowledge engineer



High Priority Terms

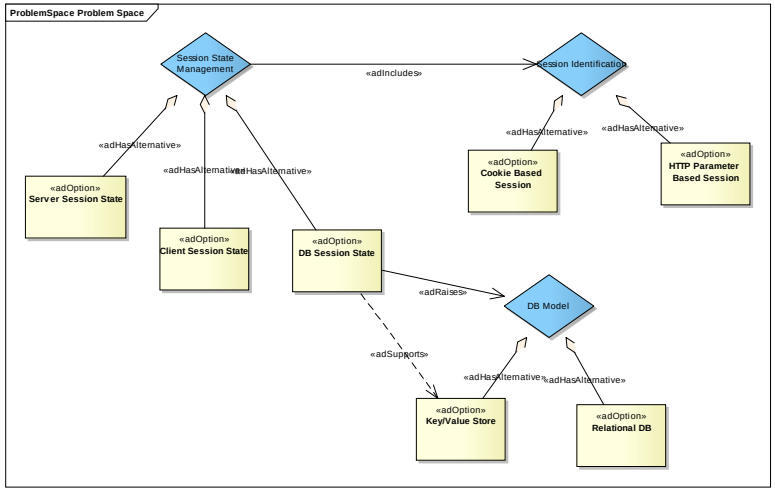
agree on, choose

Low Priority Terms

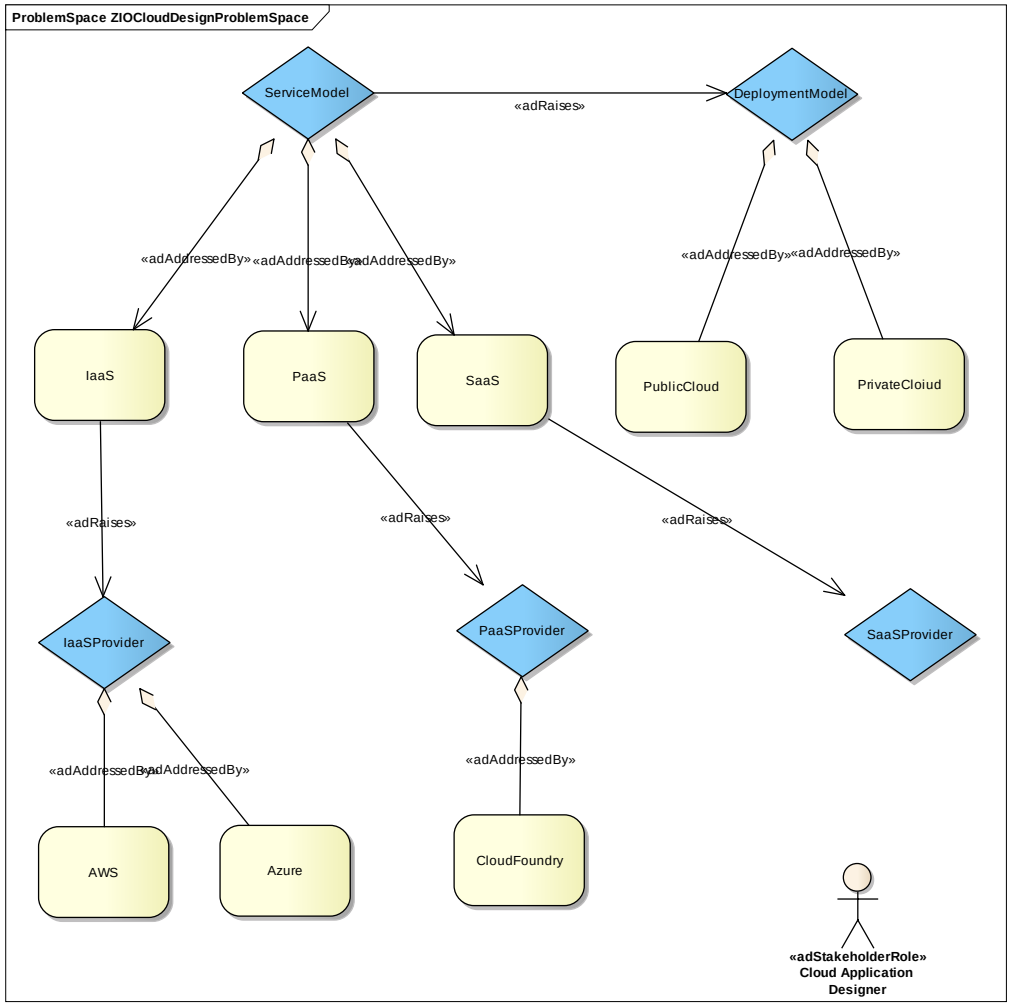
approach, articulate, class, component, construct, concern, define, design, determine, different employ, establish, evaluate, exchange, facilitate, framework, investigate, limitation, make philosophy, principle, profile, provide, protocol, recommend, refactor, require, schema select, service, several, strategy, support, topology, transaction management, type, various

Ongoing Research and Development: ADMentor

- **Joint work, HSR FHO and ABB Corporate Research**
  - Tool website: http://www.ifs.hsr.ch/ADMentor-Tool.13201.0.html?&L=4
- **Goal: Develop Add In for Sparx Enterprise Architect that supports AD reuse and sharing (on top of AD documentation features of other tools)**
  - Problem and Option vs. Problem Occurrence and Option Occurrence
  - Leverage standard product features as much as possible (e.g. rich text editor, reporting, model refactoring, links)



Alpha Version of AD Mentor



- **Problem Space Modelling**
- **Problem Space Tailoring**
  - General Properties
  - Tagged Values
- **Solution Space Creation**
  - Batch mode (full problem space)
  - Incremental
- **Decision Making**
  - With state propagation

Summary – Architectural Decisions (AD) *im Workflow Design*

- **Capture the rationale justifying a design**
  - Answers to “why” questions
- **Example:**
  - “We selected the Layers pattern to make the core banking SOA future proof, e.g., to be able to add user channels in a flexible manner”
  - See this presentation for full example and decision capturing templates
- **Practical challenges (can be overcome):**
  - Retrospective decision capturing takes time and does not yield sufficient benefits *-> lightweight templates, e.g. Y-statements*
  - Relation to other architectural concepts and viewpoints (quality attributes, patterns) not understood well and not supported in methods and tools *-> decision modeling with reuse and ADMentor tool*
- ***Many Recurring Workflow Design Decisions***
  - *Languages, transactions, integration patterns, human tasks, …*

Discussion: Strengths and Weaknesses of Workflow Technology

- **Doodleware controversy**
  - Can (and should) domain experts write programs (e.g., process flows)?
  - IDE integration (code completion, quick fixes, refactoring, etc.)?
  - Debugging and testing support?
  - Is full code generation of executable process model from graphical, business-level model possible (and desired)?
- **Is XML a good programming language and/or integration DSL?**
  - Or should an embedded workflow engine be used (JEE/Spring integration)
  - Expressions – Java Expression Language vs. XPath
  - Communication (service composition) – REST or Web services, messaging
- **Risk of vendor lock in**
  - Does the engine support all features in standard (syntax/semantics)?
  - Which proprietary modeling extensions are available in the engine?

Architectural Knowledge about Workflow Management (Sources)

- **Patterns:**
  - Workflow Patterns, http://www.workflowpatterns.com/
  - U. Zdun, C. Hentrich,
    Process-Driven SOA: Patterns for Aligning Business and IT
    - http://www.crcpress.com/product/isbn/9781439889299
- **Vendor Information:**
  - Vendor Developer Portals:
    - e.g. http://msdn.microsoft.com/en-us/library/ee658122.aspx
  - Vendor Method (in IT Service Management Product Context):
    - E.g. IBM ISTM tool guidance
- **Case Studies:**
  - Industrial IT:
    - http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&tp=&arnumber=6300859
  - Business IT:
    - http://soadecisions.org/soad.htm#oopsla05

More Information: Project Websites @ IFS HSR

(screen captions clickable)

Architectural Knowledge Hubs

Online Resources for Software Architects

Websites by thought leaders that the ARC team frequently consults (among many others):

1. Martin Fowler's Bliki
2. Gregor Hohpe's Ramblings
3. Philippe Kruchten's Weblog
4. Eoin Wood's website and blog at Artechra
5. Michael Stal's software architecture blog
6. The Software Architecture Handbook website by Grady Booch
7. Personal page of Gernot Starke (mostly in German) - arc42, aim42, IT architect profession
8. Technical Reports and other publications in the Digital Library of the Software Engineering Institute (SEI)
9. The Open Group website - IT Architect Certification, TOGAF, ArchiMate, XA
10. Object Management Group (OMG) - UML, SPEM, MDA, CORBA, ADM, KDM
11. IEEE Software, as well as SWEBOK and the very readable standard for architecture descriptions, ISO/IEC/IEEE 42010
12. Academic conferences (software architecture research): WICSA, QoSA, ECSA and online archives: ACM Digital Library, IEEE Xplore and ScienceDirect.

The following conferences have a practitioner focus on all things software architecture are (most of the presentations are available online and can be accessed from the conference websites):

1. SEI SATURN, e.g. SATURN 2013
2. Industry Day at CompArch/WICSA 2011
3. ECSA 2014 also had an Industry Day
4. OOP (most talks in German, presentations not available online by default)
5. SPLASH and OOPSLA (e.g. practitioners reports program at OOPSLA 2008)

Studiengang Informatik

HINTERGRUNDINFORMATIONEN

SOAD Project Results (2006-2011)

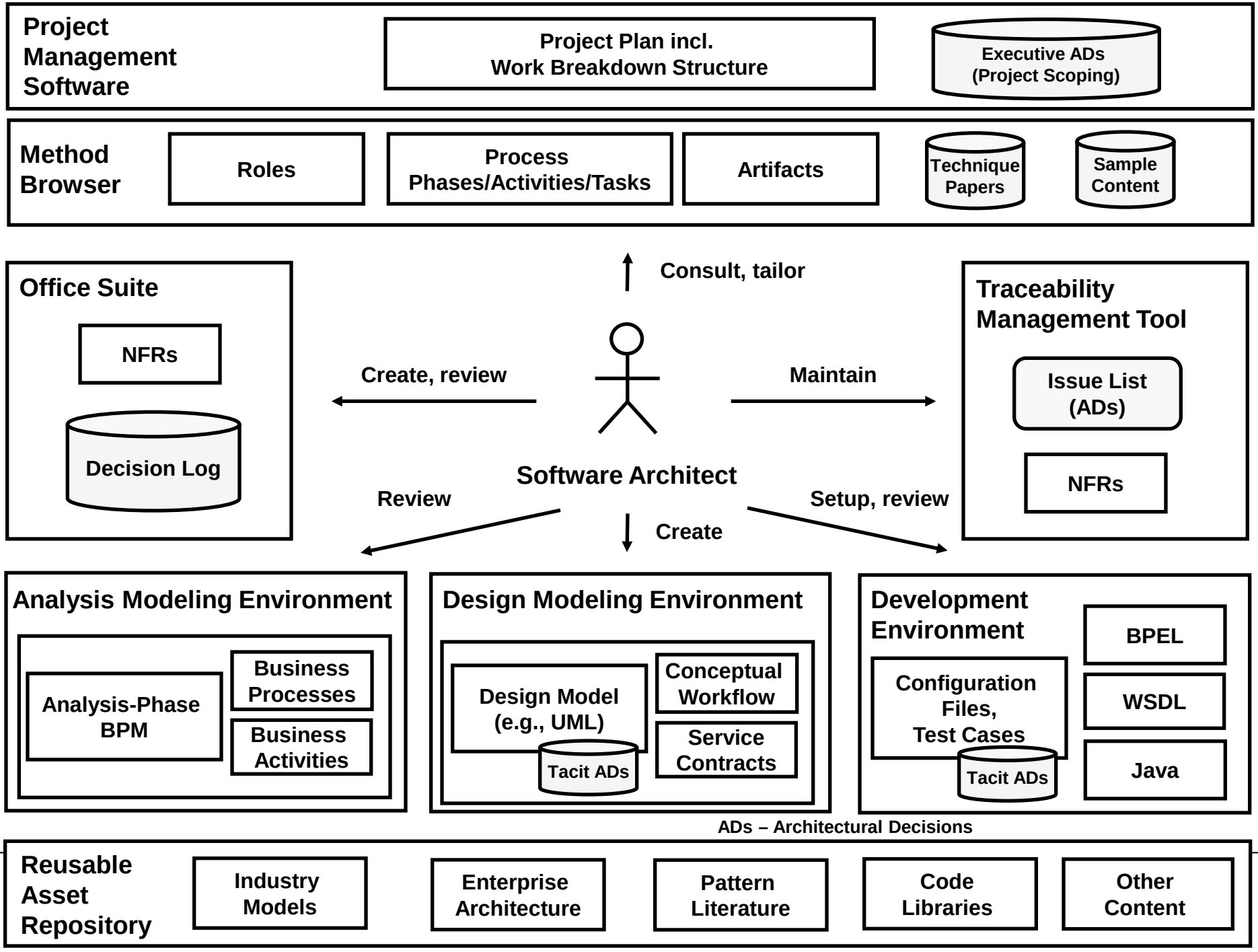
Examples

References

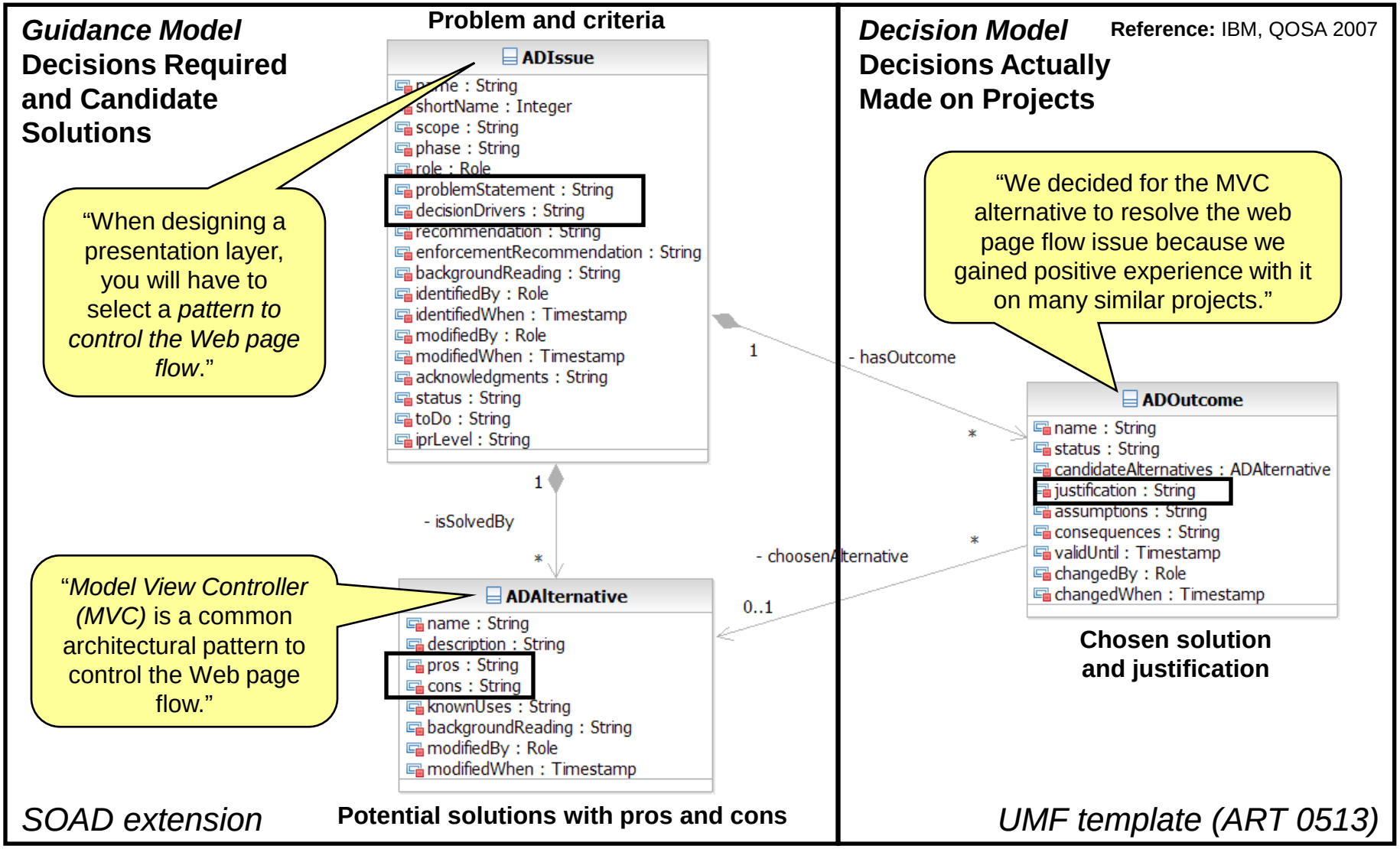
IFS INSTITUTE FOR SOFTWARE

Prof. Dr. Olaf Zimmermann
Institut für Software
Leipzig, 20. November 2014.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz


Project Management Software
Project Plan incl. Work Breakdown Structure
Executive ADs (Project Scoping)
Method Browser
Roles
Process Phases/Activities/Tasks
Artifacts
Technique Papers
Sample Content
Consult, tailor
Office Suite
NFRs
Decision Log
Create, review
Maintain
Traceability Management Tool
Issue List (ADs)
NFRs
Software Architect
Review
Setup, review
Create
Analysis Modeling Environment
Analysis-Phase BPM
Business Processes
Business Activities
Design Modeling Environment
Design Model (e.g., UML)
Tacit ADs
Conceptual Workflow
Service Contracts
Development Environment
Configuration Files, Test Cases
Tacit ADs
BPEL
WSDL
Java
ADs – Architectural Decisions
Reusable Asset Repository
Industry Models
Enterprise Architecture
Pattern Literature
Code Libraries
Other Content

Entity Types and Associations in UML Metamodel



Decisions Required vs. Decisions Made

Property	Issue (Decision Required)	Alternative (Solution Considered)	Outcome (Decision Made)
Semantics (attributes)	Need for architectural decision (motivation), technical problem, best practices recommendations	Design options (e.g. patterns) with pros and cons	Option selection, justification with rationale relative to pros and cons
Role (Owner)	Knowledge engineer (community)	Knowledge engineer	Project architect
Created when	Before/after project	Before/after project	On project
Consumed when	On project	On project	On/after project
Updated when	Periodically	Periodically	On demand
Retired when	Never	Never	Project termination

Recurring 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.

Recurring 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.

Good and Bad Justifications, Part 1

Decision driver type	Valid justification	Counter example
Wants and needs of external stakeholders	Alternative A best meets user expectations and functional requirements as documented in user stories, use cases, and business process model.	End users want it, but no evidence for a pressing business need. Technical project team never challenged the need for this feature. Technical design is prescribed in the requirements documents.
Architecturally significant requirements	Nonfunctional requirement XYZ has higher weight than any other requirement and must be addressed; only alternative A meets it.	Do not have any strong requirements that would favor one of the design options, but alternative B is the market trend. Using it will reflect well on the team.
Conflicting decision drivers and alternatives	Performed a trade-off analysis, and alternative A scored best. Prototype showed that it's good enough to solve the given design problem and has acceptable negative consequences.	Only had time to review two design options and did not conduct any hands-on experiments. Alternative B does not seem to perform well, according to information online. Let's try alternative A.

Source: Zimmermann O., Schuster N., Eeles P., Modeling and Sharing Architectural Decisions, Part 1: Concepts. IBM developerWorks, 2008

Good and Bad Justifications, Part 2

Decision driver type	Valid justification	Counter example
Reuse of an earlier design	Facing the same or very similar NFRs as successfully completed project XYZ. Alternative A worked well there. A reusable asset of high quality is available to the team.	We've always done it like that. Everybody seems to go this way these days; there's a lot of momentum for this technology.
Prefer do-it-yourself over commercial off-the-shelf (build over buy)	Two cornerstones of our IT strategy are to differentiate ourselves in selected application areas, and remain master of our destiny by avoiding vendor lock-in. None of the evaluated software both meets our functional requirements and fits into our application landscape. We analyzed customization and maintenance efforts and concluded that related cost will be in the same range as custom development.	Price of software package seems high, though we did not investigate total cost of ownership (TCO) in detail. Prefer to build our own middleware so we can use our existing application development resources.
Anticipation of future needs	Change case XYZ describes a feature we don't need in the first release but is in plan for next release. Predict that concurrent requests will be x per second shortly after global rollout of the solution, planned for Q1/2009.	Have to be ready for any future change in technology standards and in data models. All quality attributes matter, and quality attribute XYZ is always the most important for any software-intensive system.

Source: Zimmermann O., Schuster N., Eeles P., Modeling and Sharing Architectural Decisions, Part 1: Concepts. IBM developerWorks, 2008

… your AD Tool Requirements?

- **Functional (usage scenarios, use cases, user stories)**
  - …
- **Non-functional**
  - …

References

- **Architecture Documentation**
  - ISO/IEC/IEEE 42010, http://en.wikipedia.org/wiki/ISO/IEC_42010
- **Architectural Decision (AD) Capturing and Reuse:**
  - J. Tyree/A. Akerman, Architecture Decisions: Demystifying Architecture. IEEE Software, 22/2, March/April 2005
  - O. Zimmermann, Architectural Decisions as Reusable Design Assets. IEEE Software, 28/1, Jan./Feb. 2011, http://soadecisions.org/soad.htm
  - Uwe Zdun, Rafael Capilla, Huy Tran, Olaf Zimmermann: Sustainable Architectural Design Decisions. IEEE Software, 30/6, Nov./Dez. 2013

AD Coverage at SATURN 2010-2013

- **One presentation and tutorial at SATURN 2010**
  - An Architectural Decision Modeling Framework for SOA and Cloud Design
- **Five presentations and one tutorial in 2011 (AD one of seven themes)**
  - Architectural Implications of Cloud Computing
  - Guidance Models and Decision-Making Tooling for SOA, Cloud, and Outsourcing Solution Design
  - Dealing with the Complexities of a Global Service-Oriented Architecture
  - Evaluating a Partial Architecture in a ULS Context
  - Themes for Architecture Success
- **Continued coverage in 2012**
  - Y-Template introduced in Making Architectural Knowledge Sustainable
- **AD concept embedded in many presentations in 2013**
  - The Design Space of Modern HTML5/JavaScript Web Applications
  - 8 more presentations mentioning decisions in abstract

General Architectural Knowledge Sources

http://www.ifs.hsr.ch/Architectural-Refactoring-for.12044.0.html?&L=4

- **InfoQ, Stack Overflow, TheServerSide.com**
- **IBM developerWorks, MSDN and similar vendor-sponsored sites**
  - E.g. Google Developers, Amazon developer forums
- **Blogs and websites**
  - Peter Eeles, Peter Cripps
  - Gregor Hohpe's ramblings on eaipatterns.com
  - Martin Fowler's bliki
  - Philippe Kruchten's weblog and articles on architecture
  - Michael Stal on blogspot
- **Books and magazines**
  - IEEE Software magazine – free multimedia content online
  - Patterns books
  - SEI Technical Reports (TRs)