



Business Integration Technologies

# Reusable Architectural Decision Models for Enterprise Application Development

QoSA 2007  
July 13, 2007

**Olaf Zimmermann**  
Executive IT Architect  
IBM Zurich Research Lab  
[olz@zurich.ibm.com](mailto:olz@zurich.ibm.com)

## QoSA Abstract

In enterprise application development and other software construction projects, a critical success factor is to make sound architectural decisions. Text templates and tool support for capturing architectural decisions exist, but have failed to reach broad adoption so far. One of the inhibitors we perceived on large-scale industry projects is that architectural decision capturing is regarded as a retrospective and therefore unwelcome documentation task which does not provide any benefit during the original design work. A major problem of such a retrospective approach is that the decision rationale is not available to decision makers when they identify, make, and enforce decisions. Often a large, possibly distributed, community of decision makers is involved in these three steps.

In this paper, we propose a new conceptual framework for proactive decision identification, decision maker collaboration, and decision enforcement. Leveraging a meta model we extended to support reuse and collaboration, our framework instantiates decision models from requirements models and reusable decision templates. These templates capture knowledge gained on previous projects employing the same architectural style. As an exemplary application of these concepts to service-oriented architecture shows, reusable architectural decision models can speed up the decision identification and improve the quality of the decision making. Reusable architectural decision models can also simplify the exchange of architecture design rationale within and between project teams, and expose decision outcome as model transformation parameters in model-driven software development.

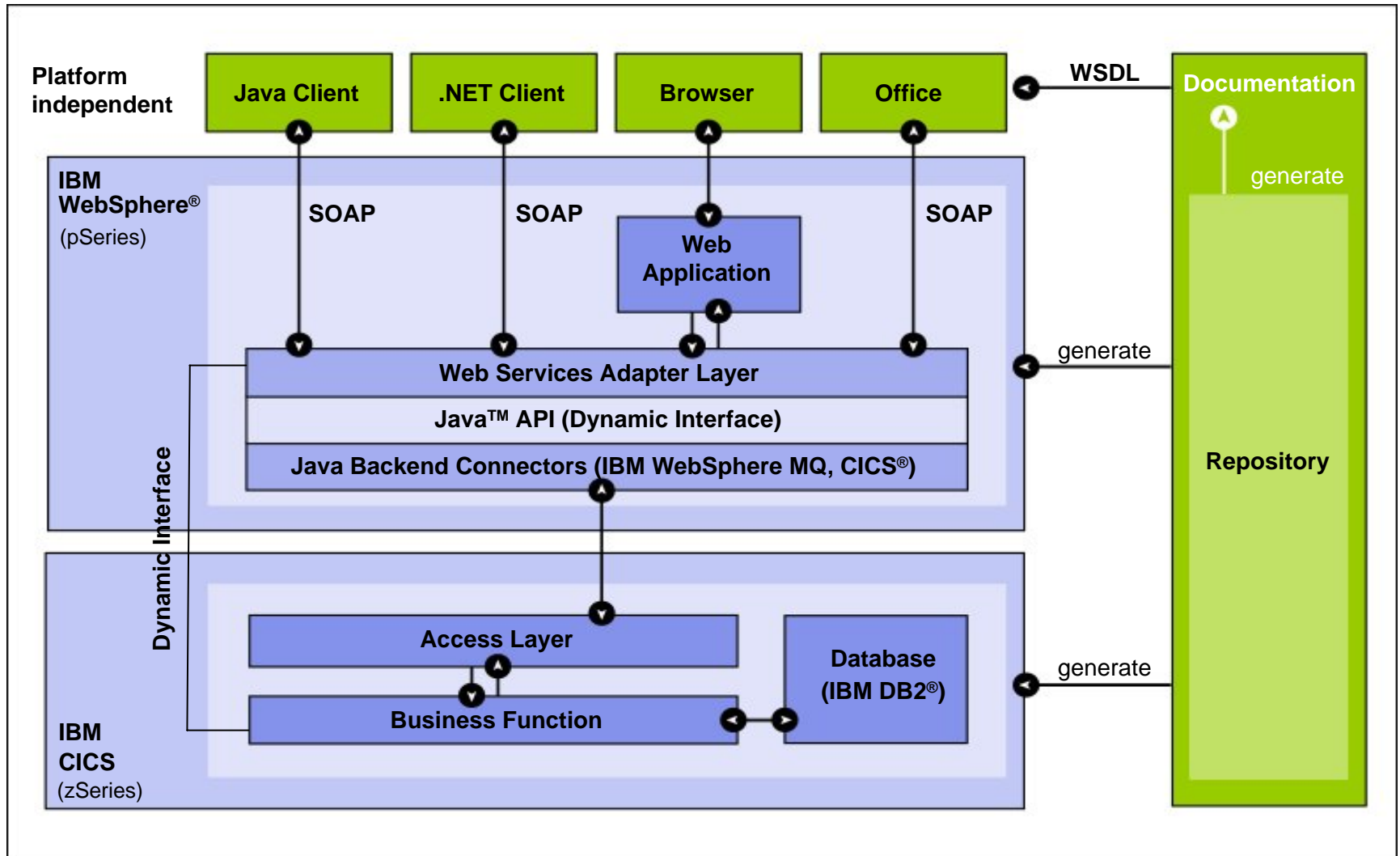
# Agenda

1. Motivating case studies
2. Extended meta model for architectural decision capturing
3. Decision modeling steps: identification, making, and enforcement
4. Example: transactional workflows and messaging in SOA
5. Tool support and practical demonstration
6. Conclusions and outlook

# Agenda

- 1. Motivating case studies**
2. Extended meta model for architectural decision capturing
3. Decision modeling steps: identification, making, and enforcement
4. Example: transactional workflows and messaging in SOA
5. Tool support and practical demonstration
6. Conclusions and outlook

# Core Banking SOA with Web Services [OOPSLA 2004]



# Multi-Channel Order Management SOA in the Telecommunications Industry [OOPSLA 2005]

## Functional domain

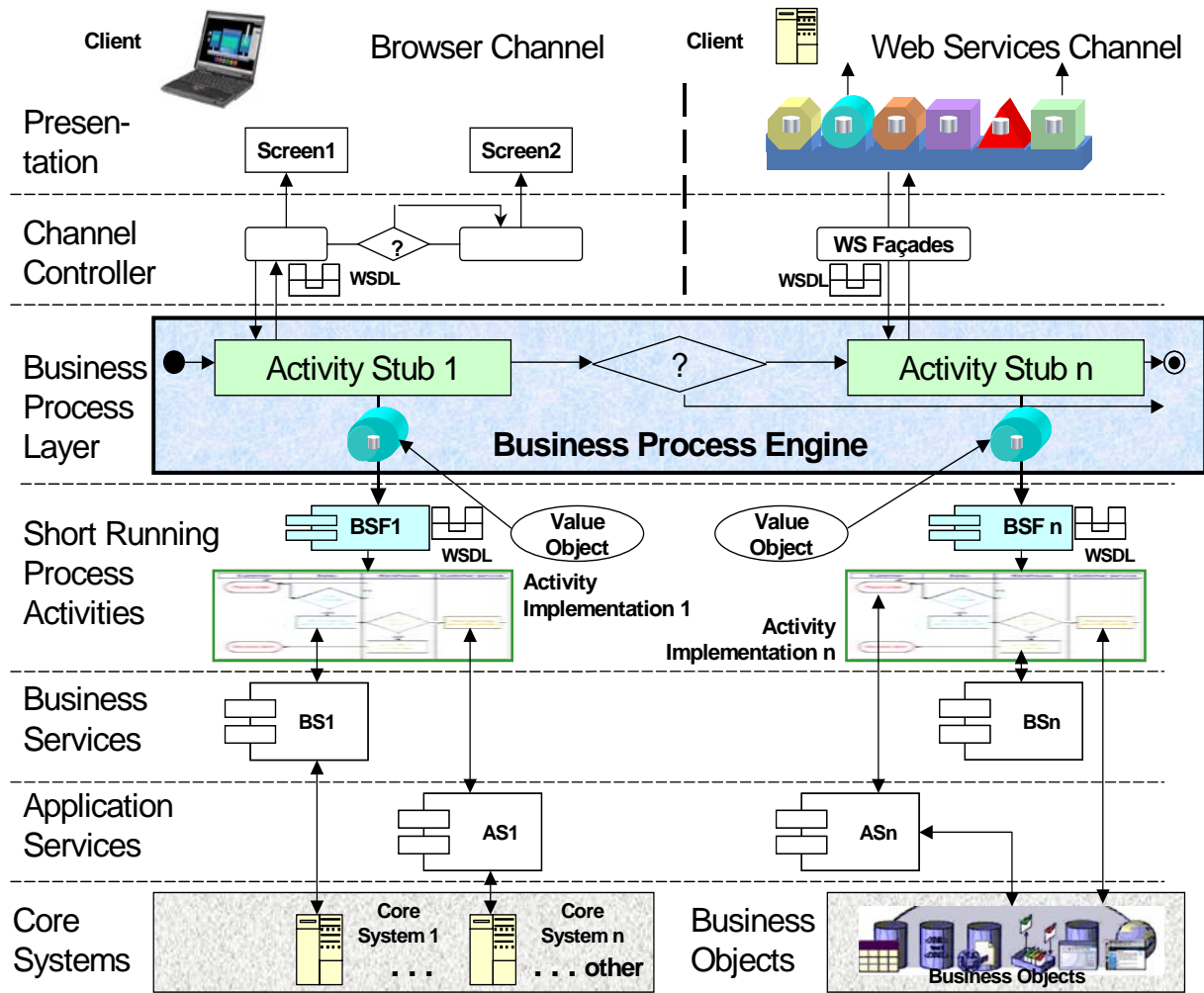
- Order entry management
- Two business processes: new customer, relocation

## Main SOA drivers

- Deeper automation grade (e.g. compensation)
- Services shared within and between domains

## Service composition

- Top-down from retailer interface and process
- Bottom-up from existing wholesaler systems



# Agenda

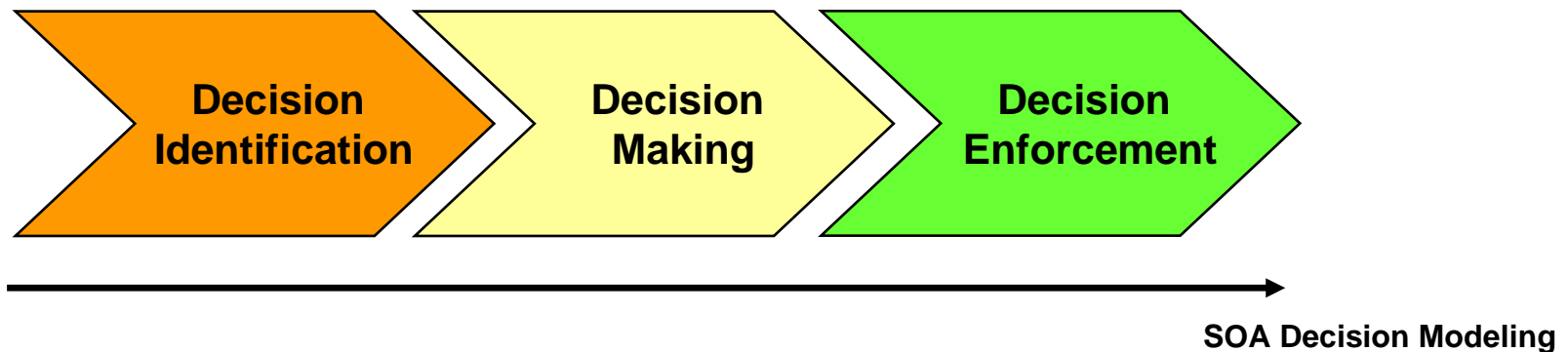
1. Motivating case studies
2. **Extended meta model for architectural decision capturing**
3. Decision modeling steps: identification, making, and enforcement
4. Example: transactional workflows and messaging
5. Tool support and practical demonstration
6. Conclusions and outlook



# Agenda

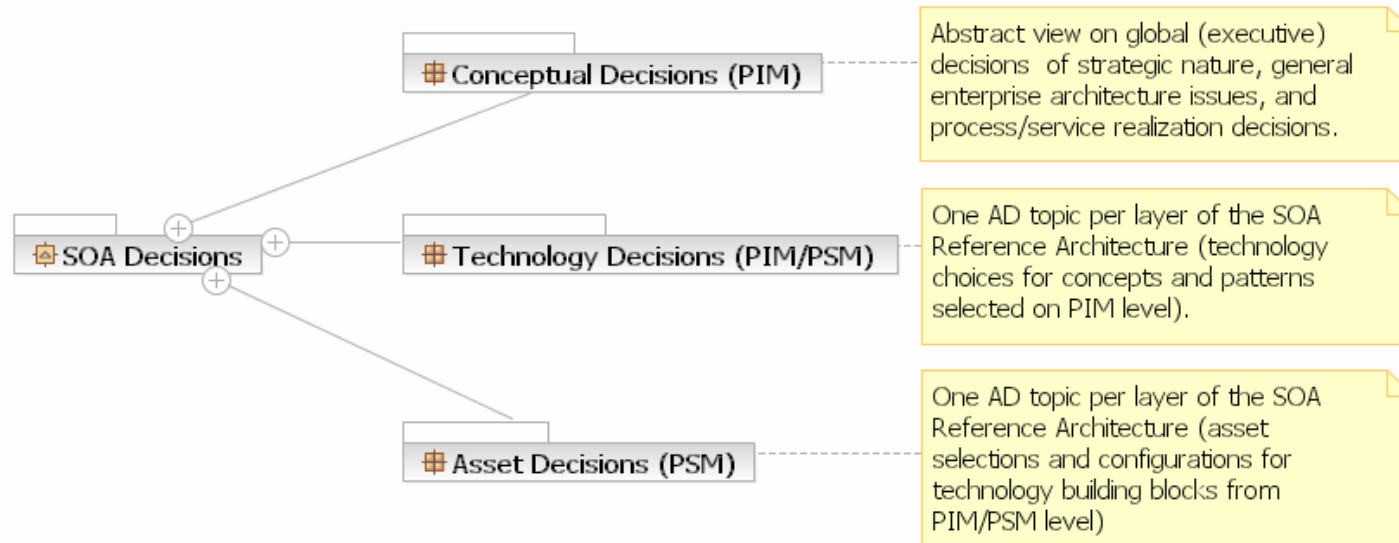
1. Motivating case studies
2. Extended meta model for architectural decision capturing
3. **Decision modeling steps: identification, making, and enforcement**
4. Example: transactional workflows in SOA
5. Tool support and practical demonstration
6. Conclusions and outlook

## Three Conceptual Decision Making Steps...



- Decision *identification* – driven by requirements, higher-level decisions
  - “To know what you don’t know (yet)”
- Decision *making* – selection of architecture alternatives based on decision drivers (the real art)
  - “To know what your client and developers expect you to know”
- Decision *enforcement* – stick vs. carrot (the latter is preferred obviously)
  - “To let others know what you now know”

## ... for 100+ Decision Types on 3 Levels of Abstraction.



### ■ Some of the AD Topics in the three levels:

- PIM: service message/operation design decisions, interface design decisions
- PIM/PSM: abstract BPEL design (vendor specifics come in on PSM layer)
- PSM: Web services stack selection and configuration, SCA deployment decisions

# Decision Identification: State of the Art and the Practice

## ■ State of the art

- Pattern languages
- Domain-specific extensions of general purpose methods
- Study of similar projects and the Web

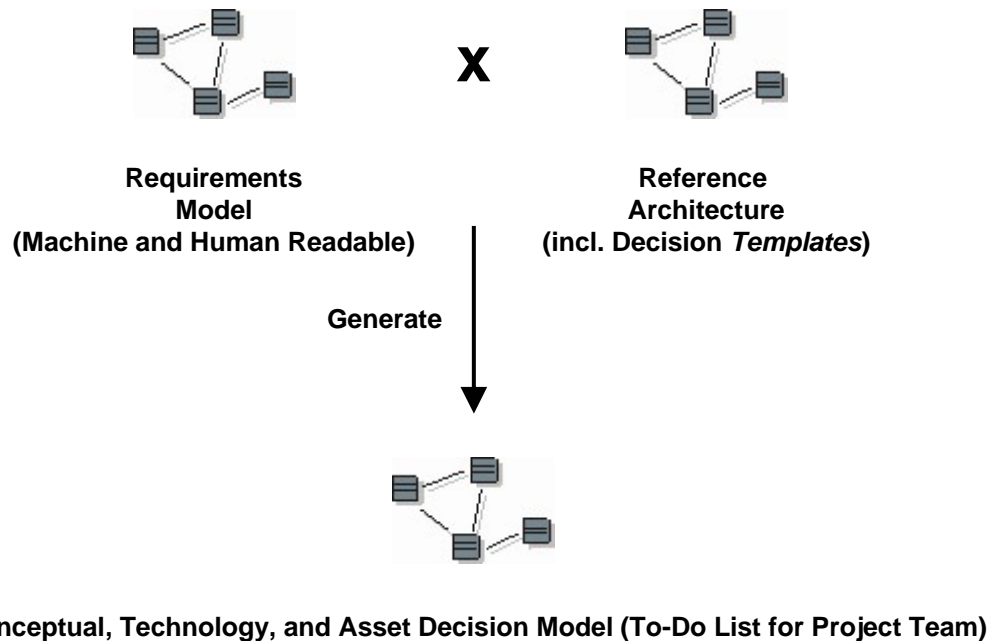
## ■ Project reality

- Ad hoc decision identification
- Triggered by requirements analysis and high level design activities, or external forces such as vendor push (or “strategic decisions”, “future needs”, “synergy potential”)

## ■ Consequences

- Too much time spent in early project phases for educational activities, which would better be invested studying the client’s business problem and designing the solution (in the decision making, that is)

## Decision Identification: Our Approach



- Decision model for project instantiated based on community decision catalog (reusable asset) and (non-)functional requirements
  - Push philosophy, from asset to project (rather than pull model as in state of the practice)
- Organizing principles: MDA levels, topic groups, cross-cutting dependencies

# Decision Making: State of the Art and the Practice

- **State of the art**

- SEI ATAM and similar approaches
- Decision support systems

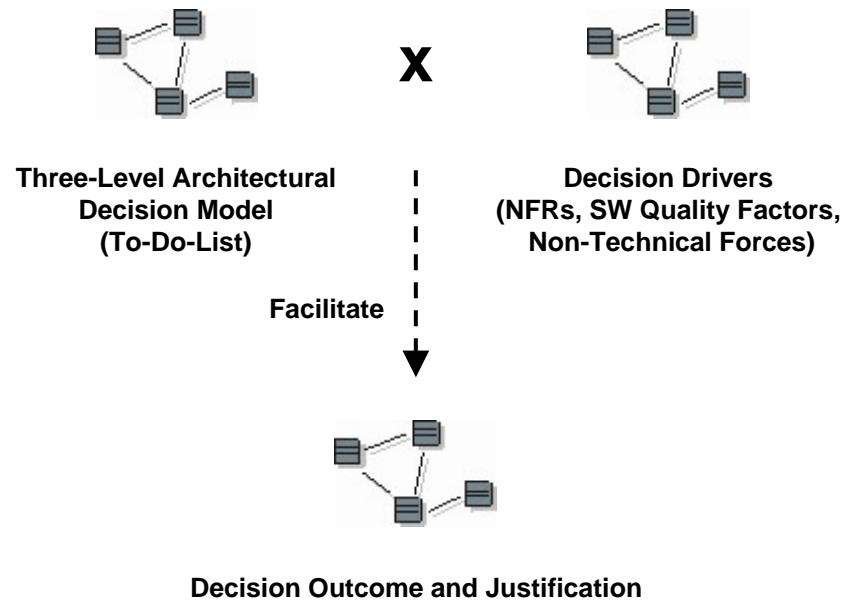
- **Project reality**

- Uninformed decision making (e.g. IGS competitor uses brain/heart/gut model)
- Personal experience and preferences as singular drivers

- **Consequences**

- Decisions based on gut feel
- At best local optima can be achieved
- Troubled projects often begin with ill-fated strategic decision making!

# Decision Making: Our Approach



- Collection of proven decision making techniques used for SOA decision making (using decision models created during identification step)
  - Continuum from basic “good practice” recommendations to slightly more structured SWOT tables to QOC diagrams to hands-on evaluations and formal alternative scoring algorithms (decision support system)
  - NFRs and general quality factors are natural decision drivers; pattern forces are QOC criteria; non-technical forces include politics, environmental constraints, team dynamics

# Decision Enforcement: State of the Art and the Practice

## ■ State of the art

- CMMI and/or heavyweight or agile processes
- Expressive, self-configuring and -healing runtimes – aspects, policies, etc.

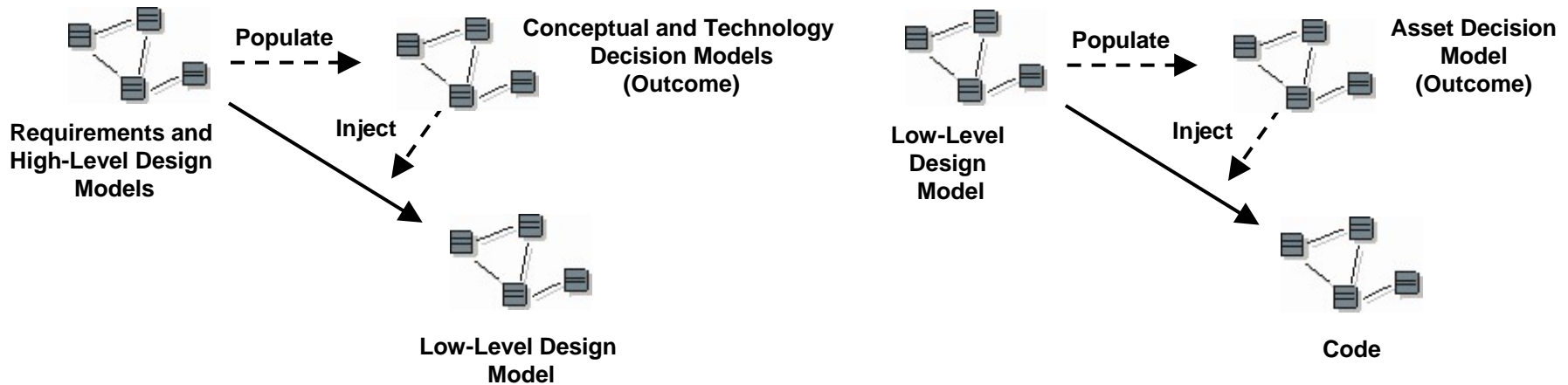
## ■ Project reality

- Coaching, architectural templates, code reviews – fine, but time consuming
- Hard coded model transformations not respecting architectural decisions
  - Negative example: technical attributes view in WBM 6.x (exposes some irrelevant decisions, e.g. naming, hard codes the relevant ones using inappropriate defaults)

## ■ Consequences

- Resources are wasted for correcting design errors detected late in the game
- Architectural knowledge lost during maintenance phase

## Decision Enforcement: Our Approach

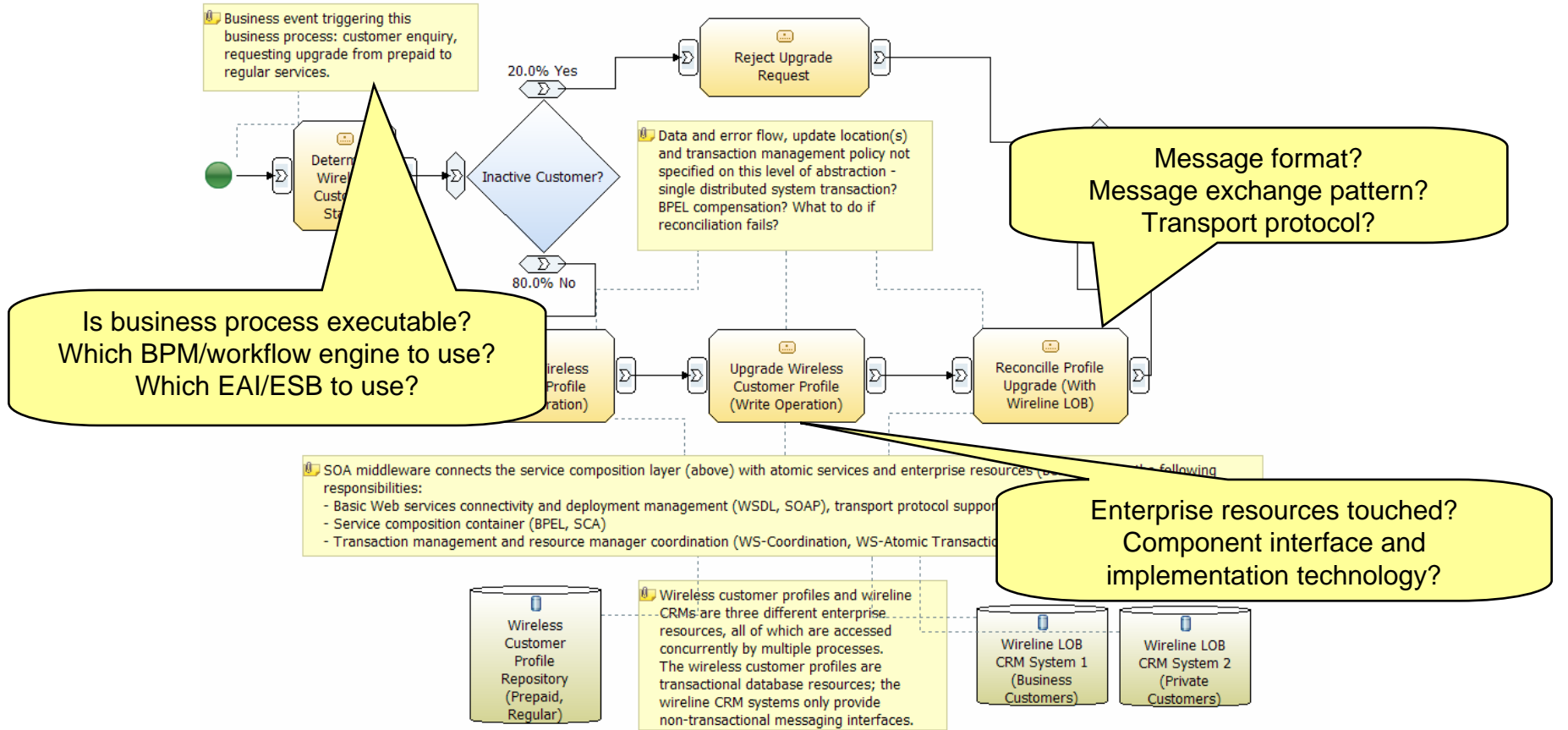


- Existing practices are fine for *some* decisions, typically lower-level ones
  - Decision injection into MDA model transformations as an additional option in case expressive runtimes are not available yet
- Machine-readable models can be interpreted by model transformations and code generators
  - Positive example: DPTK/JET (e.g. transactionality decision injection into WID)

# Agenda

1. Motivating case studies
2. Extended meta model for architectural decision capturing
3. Decision modeling steps: identification, making, and enforcement
4. **Example: transactional workflows and messaging in SOA**
5. Tool support and practical demonstration
6. Conclusions and outlook

# Decision Identification Example: Process Transactionality



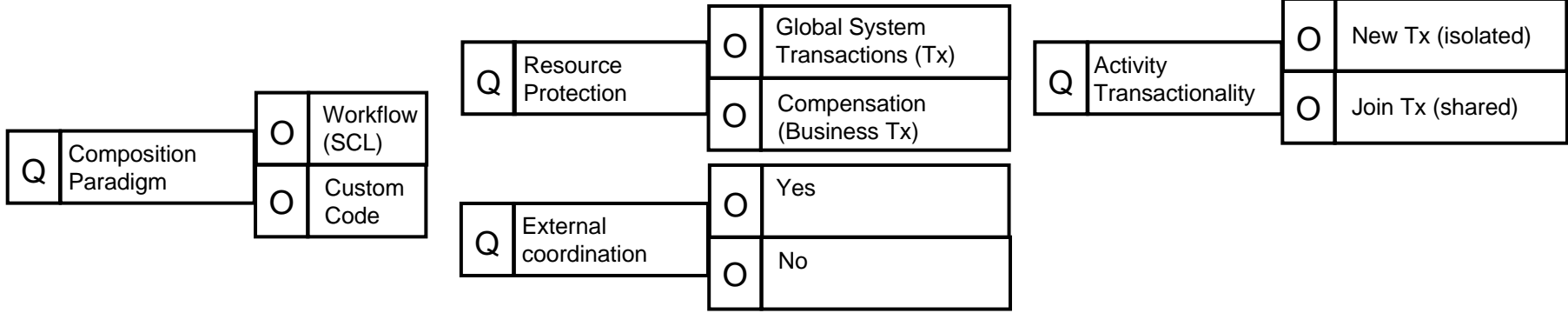
- Service architects concerned with many decisions, e.g. process transactionality, protocols, product selection/configuration, operational model

**Project Scope** →

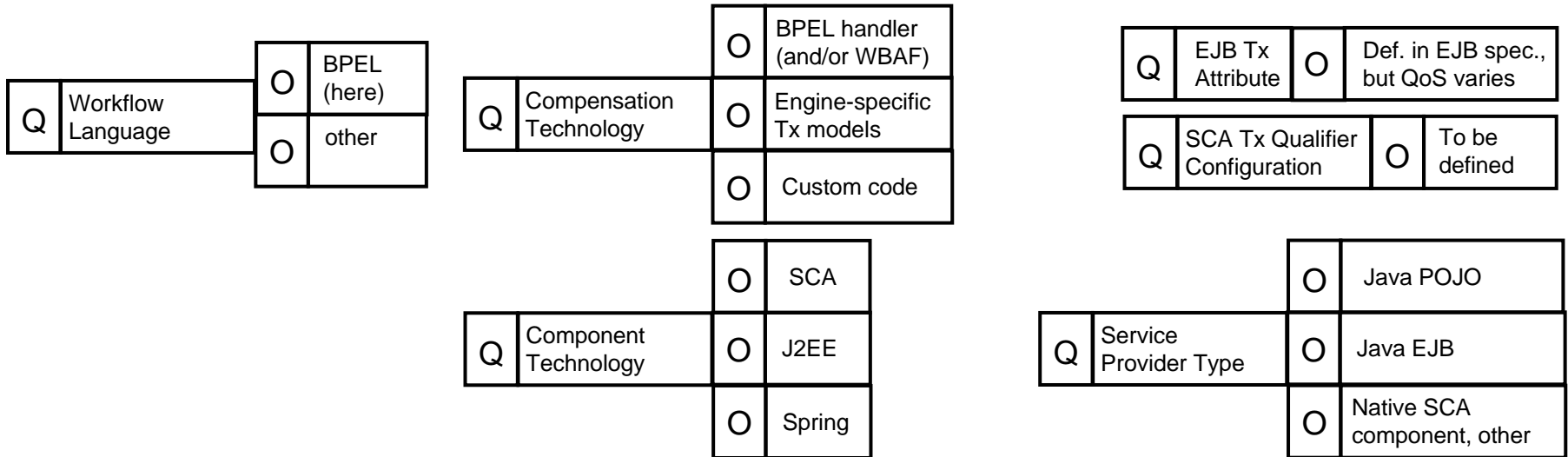
**Process Scope** →

**Service / Operation Scope**

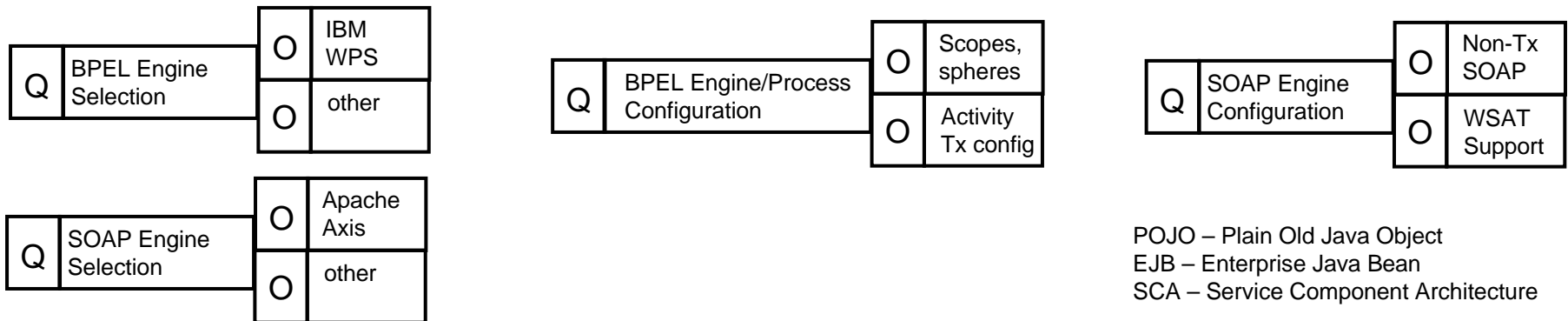
**Conceptual Level**



**Technology Level**



**Asset Level**



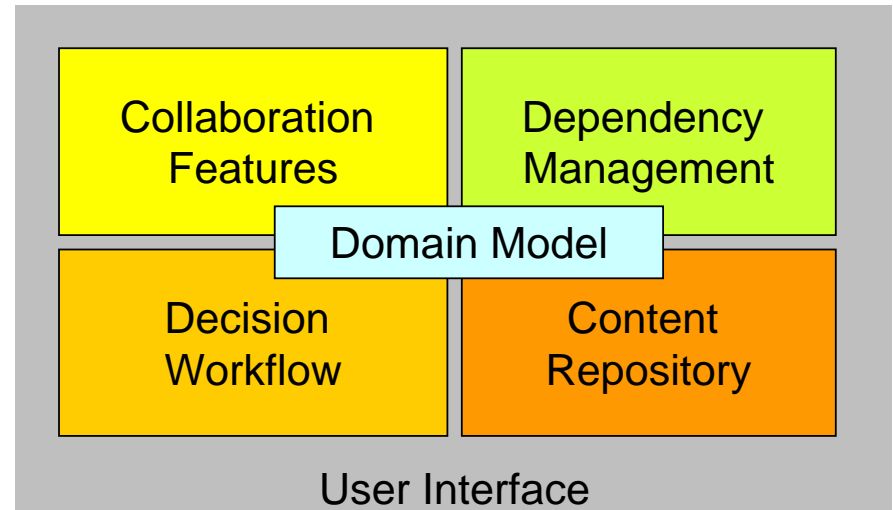
POJO – Plain Old Java Object  
 EJB – Enterprise Java Bean  
 SCA – Service Component Architecture

# Agenda

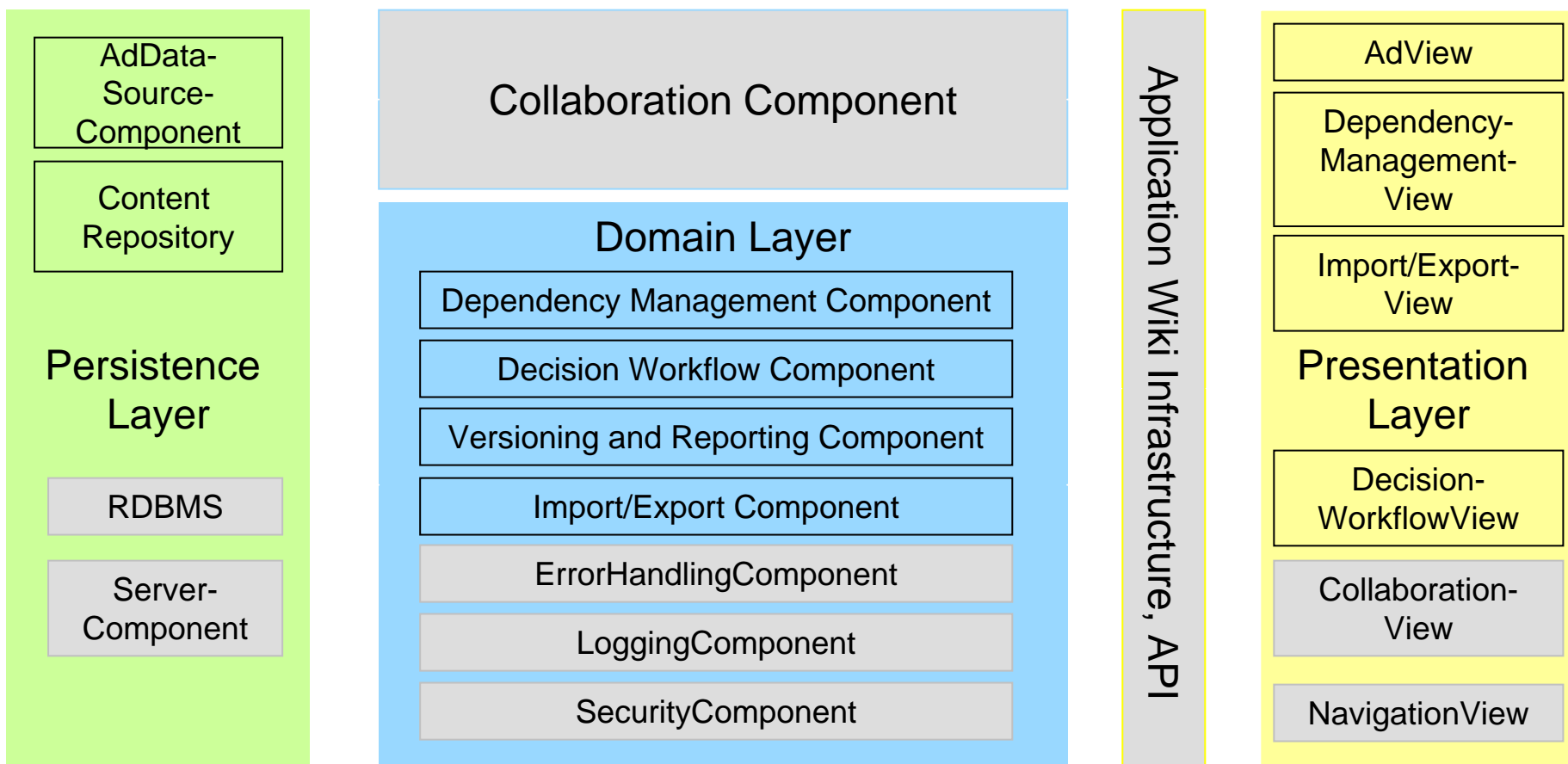
1. Motivating case studies
2. Extended meta model for architectural decision capturing
3. Decision modeling steps: identification, making, and enforcement
4. Example: transactional workflows and messaging in SOA
5. **Tool support and practical demonstration**
6. Conclusions and outlook

## Decision Knowledge Engineering Use Cases [SEKE]

- Regulatory compliance
  - E.g., maturity models
- Collaboration
  - In geographically distributed teams
- Reuse
  - Of already gained knowledge
- Resulting use cases:
  - *Obtain* architectural knowledge, captured in decision models
  - *Tailor* imported decision models according to project-specific needs, e.g., filtering content.
  - *Involve* experts from other projects and CoP leaders when looking for advice.
  - *Manage* dependencies between correlated decisions automatically to guide the user.
  - *Share* gained architectural knowledge with other projects and CoP (after sanitization).



# Conceptual Architecture of AD<sub>kwik</sub> [SEKE]



# Decision Making Example: Process Transactionality

SOAD Project WikiPage User Settings Help
View Mash up Collaborate Debug

## Ead-04 Transaction Management Pattern

**Scope:** CustomerProfileUpgrade process  
**Phase:** solution outline

**Problem Statement**  
Transaction management ensuring ACID characteristics is a system-level response to resource integrity requirements. In a business process execution environment, both the process and all invoked services have to decide for their transaction management settings, many of which of vendor-specific nature. With this decision, we raise the discussion to the platform-independent level and suggest selecting between 3 proven patterns.

**Forces (Decision Drivers)**  
Forces include resource protection needs, parallelism, concurrency, number and size of transactions. And usual NFRs.

**Alternatives**

1 DTP Transaction Islands (de)

2 DTP Transaction Bridge

3 QTP Asynchronous Stilts

**DTP Transaction Islands (default)**

**Description**  
This pattern comprises of the primitives PAT-N, PT-SNT, CWT-CS, ST-N

**Pros**  
Reasonable size of transaction

**Contras**  
Quite a few transactions. Switch to DTP Transaction Bridge if needed.

**Known Uses**

**References**  
See draft transaction management decisions and patterns paper for more information (attached to this Wiki page).

Modified by SoadAdmin on 2007-01-24 16:48:15

status: open (Decide now)  
[ask community](#)

**Go to**  
[Problem Statement](#)  
[Forces \(Decision Drivers\)](#)  
[Alternatives](#)  
[Recommendation](#)  
[Enforcement](#)

**References**  
 If you need a quick reminder what transaction management is about, take a look at the following article:  
<http://www-128.ibm.com/developerworks/java/libr>  
 If you need a thorough introduction to the topic at hand, in a workflow/BPM context, we recommend Chapter 7 of the following book: "Production Workflow" bz F. Leymann and D. Roller (see our [\[\[References\]\]](#) page).

---

**Relationships**

- implies [suspendTx Qualifier \(Reference\)](#)
- implies [Process Activity Transactionality \(PAT\)](#)
- implies [Compensable attribute](#)
- implies [Protocol Transactionality \(PT\)](#)

[relationship editor](#)

**Information**  
 Responsible person for this AD: [lead architect](#)

This AD was identified by [AwbUser](#) on 2007-01-24 16:18:54

# Agenda

1. Motivating case studies
2. Extended meta model for architectural decision capturing
3. Decision modeling steps: identification, making, and enforcement
4. Example: transactional workflows and messaging in SOA
5. Tool support and practical demonstration
6. **Conclusions and outlook**

## Preliminary Validation Results for SOAD

- Implemented prototypical tool support for concepts and content
  - Eclipse-based Architects' Workbench
  - Web 2.0 collaboration front end
- Applied approach retrospectively to own SOA case studies
  - 26 reusable decisions captured in text book "Perspectives on Web Services", Springer-Verlag
- Coached IT architects with SOA and other expertise to use SOAD
  - SOA coach in professional services engagement: 13/15 required decisions anticipated
  - Information integration architects: appreciated level and relationship concepts

## Summary of Key Concepts in SOAD

- **Pre-population** of decision space
  - SOAD contains best practices from SOA early adoption projects
- **Decision models** replacing text templates
  - Three levels of refinement from concepts to technology to assets
  - Decision consistency checking and propagation through relationships
  - Decision scoping providing link to design models
- Decision maker **collaboration**
  - Existing meta models extended to support decision lifecycle
  - Web 2.0 front end
- **Decision outcome injection** into model transformations
  - Faithful to original vision of MDA

## Advanced Usage Scenarios and Future Research

- **Project planning and health checking**
  - Work breakdown structures can be created from the decision model, as open decisions correspond to required activities
  - If there are many, frequent changes, or many questions are still unresolved in late project phases, the project is likely to be troubled
- **Decision models as input to software configuration planning**
  - Product selection and operational modeling decisions define which software licenses are required, and on which hardware nodes the required software has to be installed
- **Enterprise architecture instrument**
  - Company-specific instance of the SOA Design Space, consisting of a subset of decisions and alternatives to give freedom of choice to individual project teams without sacrificing overall architectural integrity
- **Prescriptive micro-methodology for SOA construction**