

# An Architectural Decision Modeling Framework for SOA and Cloud Design

Dr. Olaf Zimmermann IBM Research – Zurich

# Session Abstract

In this presentation, we demonstrate how reusable architectural decision models can support Service-Oriented Architecture (SOA) and cloud design: We present an architectural decision modeling framework called SOA Decision Modeling (SOAD) which treats recurring architectural decision as first-class architecture design artifacts. SOAD provides a technique to systematically identify such recurring decisions.

We also present a reusable architectural decision model for SOA that was created with SOAD. This model separates long lasting platform-independent decisions from rapidly changing platform-specific ones; the alternatives in a conceptual model level reference architectural patterns. SOAD has its roots in our industry projects conducted since 2001; it has been leveraged successfully by practitioners since 2006.

SOAD is not only applicable to enterprise application, SOA, and cloud design, but also to other application genres and architectural styles. It supports use cases such as education, knowledge exchange, design method, review technique, governance instrument, and architecture change management.

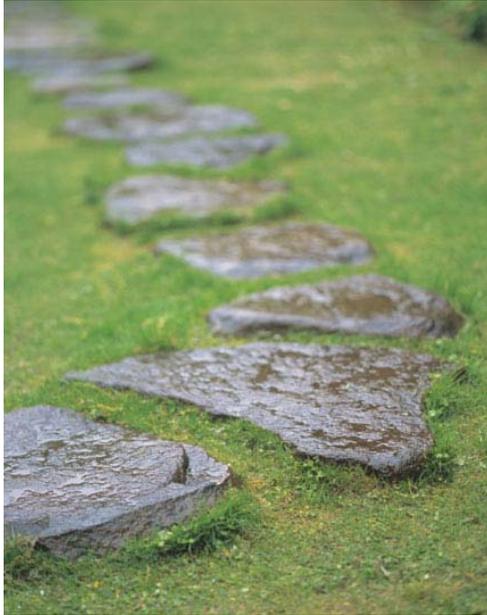
# Agenda

- Motivation and case study
- Usage scenarios for architectural decision modeling
- Scenario 1: After-the-Fact Decision Capturing
- Scenario 2: Active Method Guidance
- Scenario 3: Cross-Role Collaboration and Tool Integration
- Emerging tool support (demo)
- Discussion and summary

# Agenda

- **Motivation and case study**
- Usage scenarios for architectural decision modeling
- Scenario 1: After-the-Fact Decision Capturing
- Scenario 2: Active Method Guidance
- Scenario 3: Cross-Role Collaboration and Tool Integration
- Emerging tool support (demo)
- Discussion and summary

# Architectural Decisions: Current Trends



- Decision capturing support soon to be mandatory in architecture descriptions conforming to standard

<http://www.viewpoints-and-perspectives.info>

<http://www.architecting.co.uk/index.php>

**ISO/IEC WD4 42010**  
**IEEE P42010/D6**

- Popular text books and mature methods promote the concept
- Practitioner demand
- No tools yet

# What are Architectural Decisions? Why Bother?

- **“The design decisions that are costly to change” (Grady Booch, 2009)**

- **Our definition (from <http://soadecisions.org/soad.htm#wicsa>):**

*“Architectural decisions capture key design issues and the rationale behind chosen solutions. They are conscious design decisions concerning a software system as a whole, or one or more of its core components, with impact on non-functional characteristics such as software quality attributes.”*

- **From IBM UMF work product description ART 0513 (previous name: ARC 100):**

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

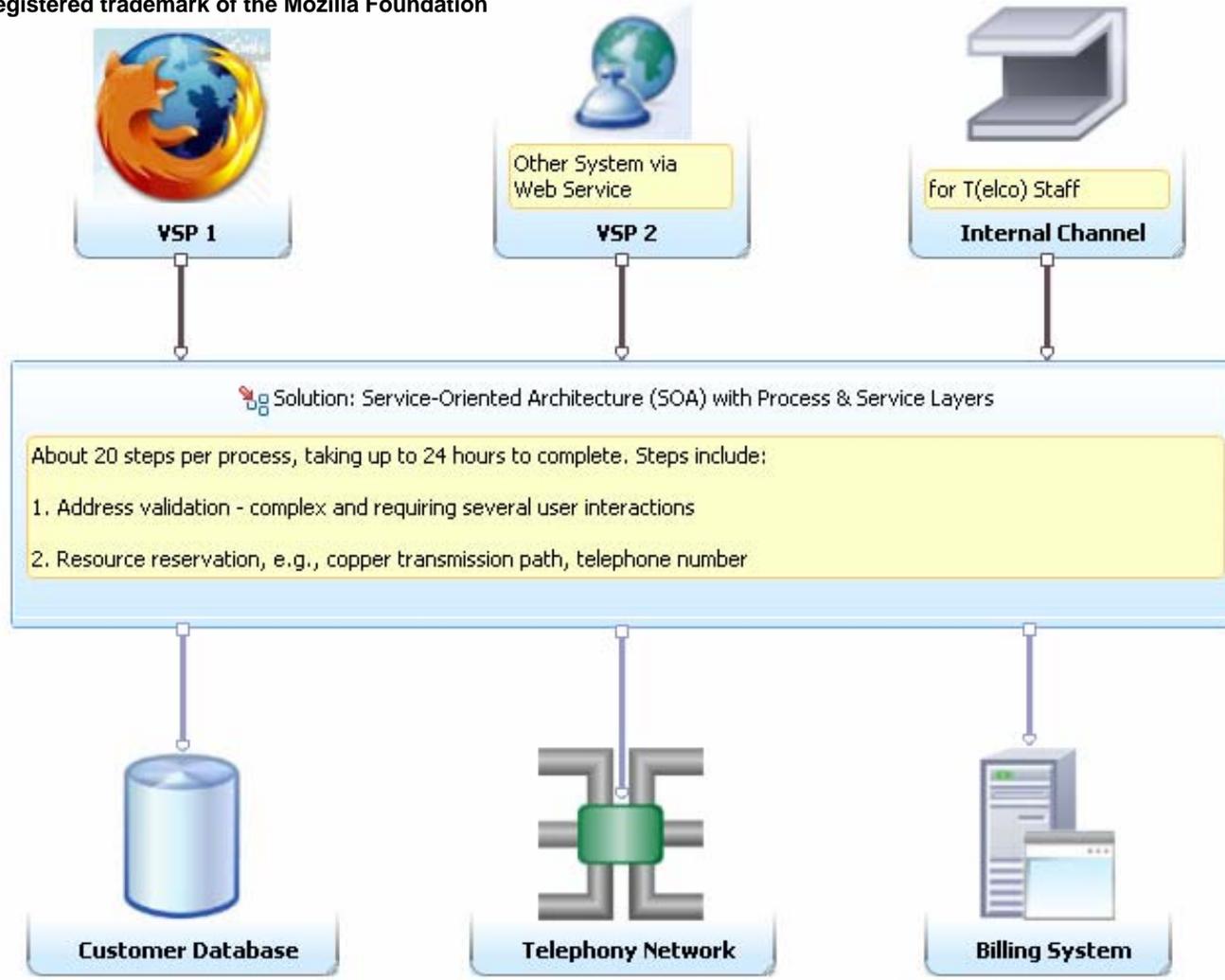
- Provide a single place to find important architectural decisions
- Make explicit the rationale and justification of architectural decisions
- Preserve design integrity in the provision of functionality and its allocation to system components
- Ensure that the architecture is extensible and can support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

Avoid unnecessary reconsideration of the same issues”

# Case Study: Telco Service Orders

VSP – Virtual Service Provider  
PSTN – Public Switched Telephone Network

Firefox is a registered trademark of the Mozilla Foundation

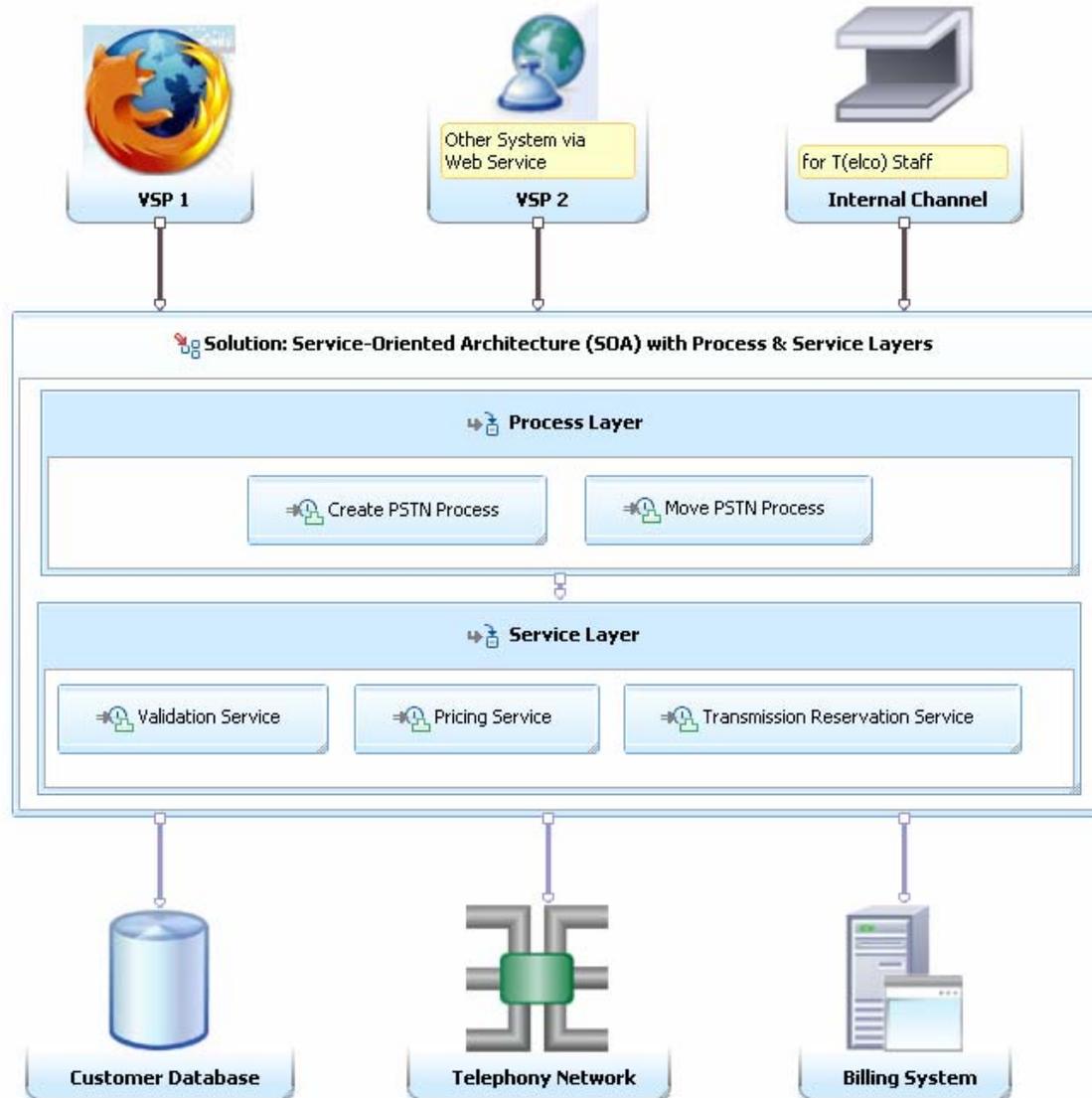


# Important Non-Functional Requirements (NFRs)

- 1. The software system supporting the two order management processes must be accessible both over a private industry-sponsored network and the Internet. The VSPs and the backend systems to be integrated (e.g., billing system) change over time.**
- 2. Business volumes are approximately 20,000 “Create PSTN service” requests and 15,000 “Move PSTN service” requests per month.**
  - Given up to 20 steps per process, and a peak hour load of 30% above average, this equates to a peak load of about 4,550 steps executed per hour (based on core business hours of ten hours per day, 20 days per month)
- 3. Initially, process instances must be able to persist from first step to last for three hours; however, this time will be extended to up to 24 hours in the future.**
- 4. VSPs are spread across a number of time zones, operating 23 hours per day and seven days per week.**
- 5. Average response time targets vary by process step, typically 3-5 seconds; 95% of the user interactions need to complete execution in 5-8 seconds.**
- 6. Domain-specific architecture design challenges include: 1. Address validation completeness and timeliness, 2. Releasing reserved resources (copper transmission path, telephone number) when a process instance fails or customer walks away.**
- 7. Communication patterns and protocols must support multiple platforms.**

# Architecture Overview Diagram (Informal)

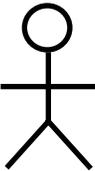
Firefox is a registered trademark of the Mozilla Foundation



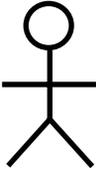
# Agenda

- Motivation and case study
- **Usage scenarios for architectural decision modeling**
- Scenario 1: After-the-Fact Decision Capturing
- Scenario 2: Active Method Guidance
- Scenario 3: Cross-Role Collaboration and Tool Integration
- Emerging tool support (demo)
- Discussion and summary

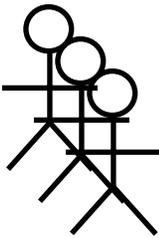
# Three Usage Scenarios for Architectural Decisions



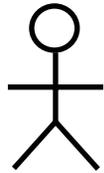
Knowledge Engineer  
(EA Specialty)



Solution Architect



EA Staff (e.g. CoE)  
Project Team (C/ALM)  
External Parties (e.g. Auditors)



Solution Architect

## **Scenario 1: After-the-Fact Decision Capturing**

1.1 Document decisions made and their rationale (supporting decision log report generation)

## **Scenario 2: Active Method Guidance**

- 2.1 Distinguish decisions required (issues) from decisions made (outcomes)
- 2.2 Share issues and related best practices via guidance models
- 2.3 Assign design work items (issues with open outcomes) to team members and track decision making progress (a.k.a. "backlog")

## **Scenario 3: Cross Role Collaboration**

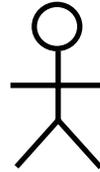
- 3.1 Identify issues in requirements artifacts and trace their resolution
- 3.2 Bind architectural decisions to enterprise-wide architectural principles and reusable assets such as pattern repositories
- 3.3 Enforce decisions in UML and topology models
- 3.4 Integrate with process tasks
- 3.5 Measure decision making practices (model analysis)

# Agenda

- Motivation and case study
- Usage scenarios for architectural decision modeling
- **Scenario 1: After-the-Fact Decision Capturing**
- Scenario 2: Active Method Guidance
- Scenario 3: Cross-Role Collaboration and Tool Integration
- Emerging tool support (demo)
- Discussion and summary

# Scenario 1 (Documentation): A Story from the Case Study (2004-2005)

“According to our **method**, I have to create a **work product** called ‘Architectural Decisions’. It is supposed to record the key **decisions made** on the order management SOA project, and the **rationale** behind them.”



**Solution Architect**

“This will help me survive the upcoming technical **quality assurance review** requested by the world-wide SOA subject matter expert they will be flying in shortly. And hopefully it will **stop** these **endless and pointless discussions** ‘why SOA’ that our developers have been raising since the project start.”

“So let’s create an architectural decisions work product:

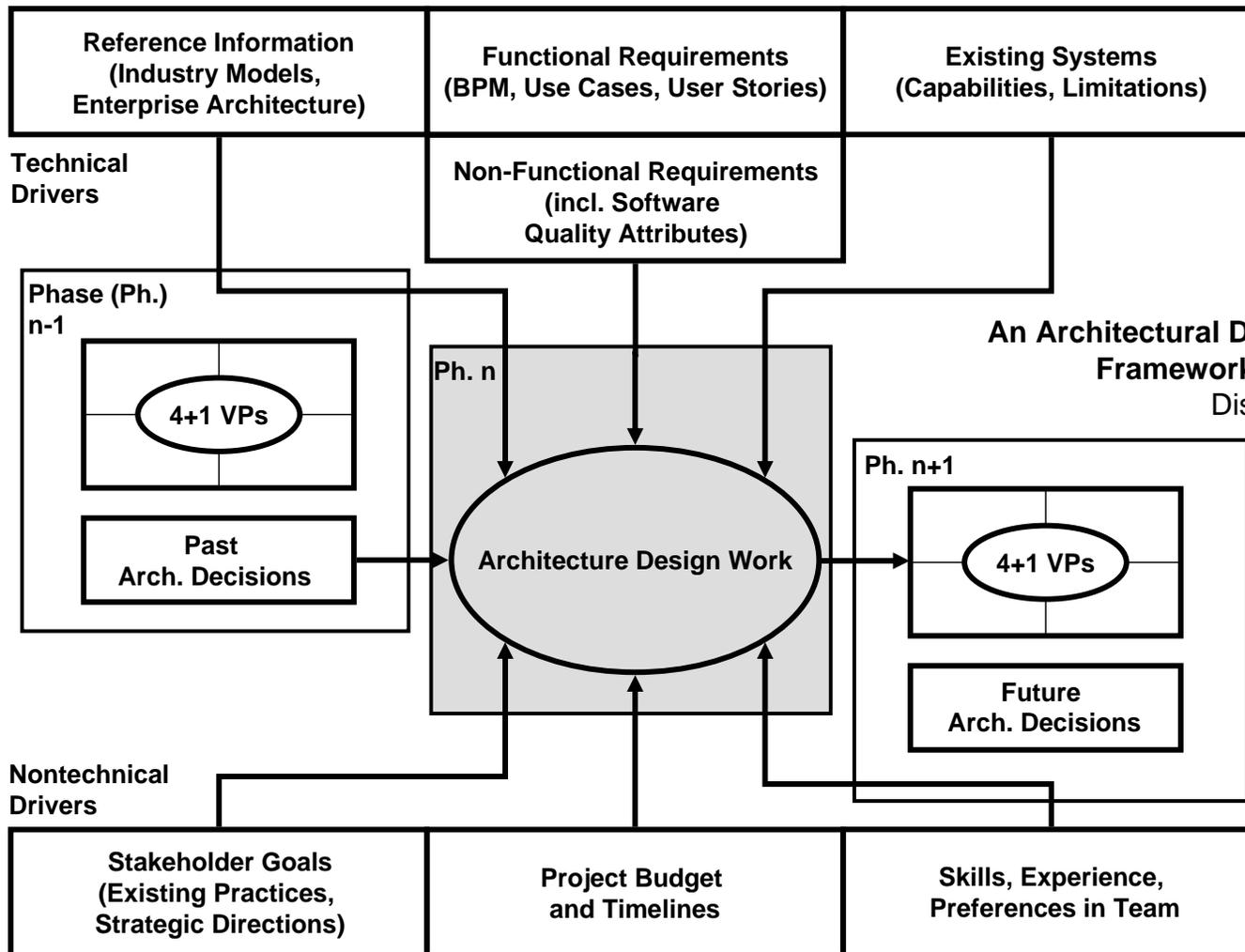
1. We **decided for process-enabled SOA** because the business scenario is a **long running, multi channel, multi actor scenario** with **heavy resource coordination requirements**. There are 20 backend systems, only a few of which are transactional.
2. We decided for **BPEL** because **it is a standardized workflow language** with emerging tool and runtime support. The out-of-the-box support for compensation will help us meet the coordination requirements.
3. We decided for **IBM WebSphere products** because this **the preferred vendor** of the client. The IBM BPEL engine can easily handle the required load. BPEL is new (July 2004), so lab advocacy will be needed.

This is getting **tedious in MS Word** although I have all insight I need. Will capture the next few ones later. I really **need a tool** that supports decision capturing and sharing.”

# Architectural Decision (AD) about Integration Style – Documented according to IBM Unified Method Framework (UMF)

| Subject Area         | Process and service layer design   | Topic | <i>Integration</i> |
|----------------------|--|-------|--------------------|
| Name                 | Integration Style  | AD ID | <b>3</b>           |
| 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 requirements NFR 1 to NFR 7 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 T staff expect immediate responses to their requests (NFR 5). Messaging will give us guaranteed delivery (NFR 3, NFR 6).                     |       |                    |
| Implications         | Need to select, install, and configure a message-oriented middleware.  |       |                    |
| 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.                                 |       |                    |

# Architectural Decision Making in Context – Decision Drivers



Zimmermann O.,  
An Architectural Decision Modeling  
Framework for SOA Design.  
Dissertation.de, 2009

# Valid Justifications... and Counter Examples

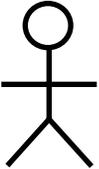
- **Convincing rationale:**
  - Direct link to (non-)functional requirements, quality attributes in particular
  - Positive experience on previous project
  - Existing skills, license agreements
- **Poor justifications:**
  - Market momentum (technology or vendor push)
  - Only one alternative known/considered
  - Keep CVs of team members current
- More examples are given in this IBM developer works article:  
<http://www.ibm.com/developerworks/architecture/library/ar-knowwiki1/>

# Agenda

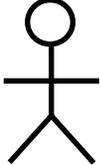
- Motivation and case study
- Usage scenarios for architectural decision modeling
- Scenario 1: After-the-Fact Decision Capturing
- **Scenario 2: Active Method Guidance**
- Scenario 3: Cross-role Collaboration and Tool Integration
- Emerging tool support (demo)
- Discussion and summary

# Scenario 2 (Guidance): The Case Study Continues

“This is going to be an interesting assignment. According to the PPT charts that outline the architecture, the team decided for SOA but does not know much about the **design issues** that it has to resolve now. These issues include **service granularity, transaction boundaries, message exchange patterns.**”



Knowledge  
Engineer  
(SOA SME)



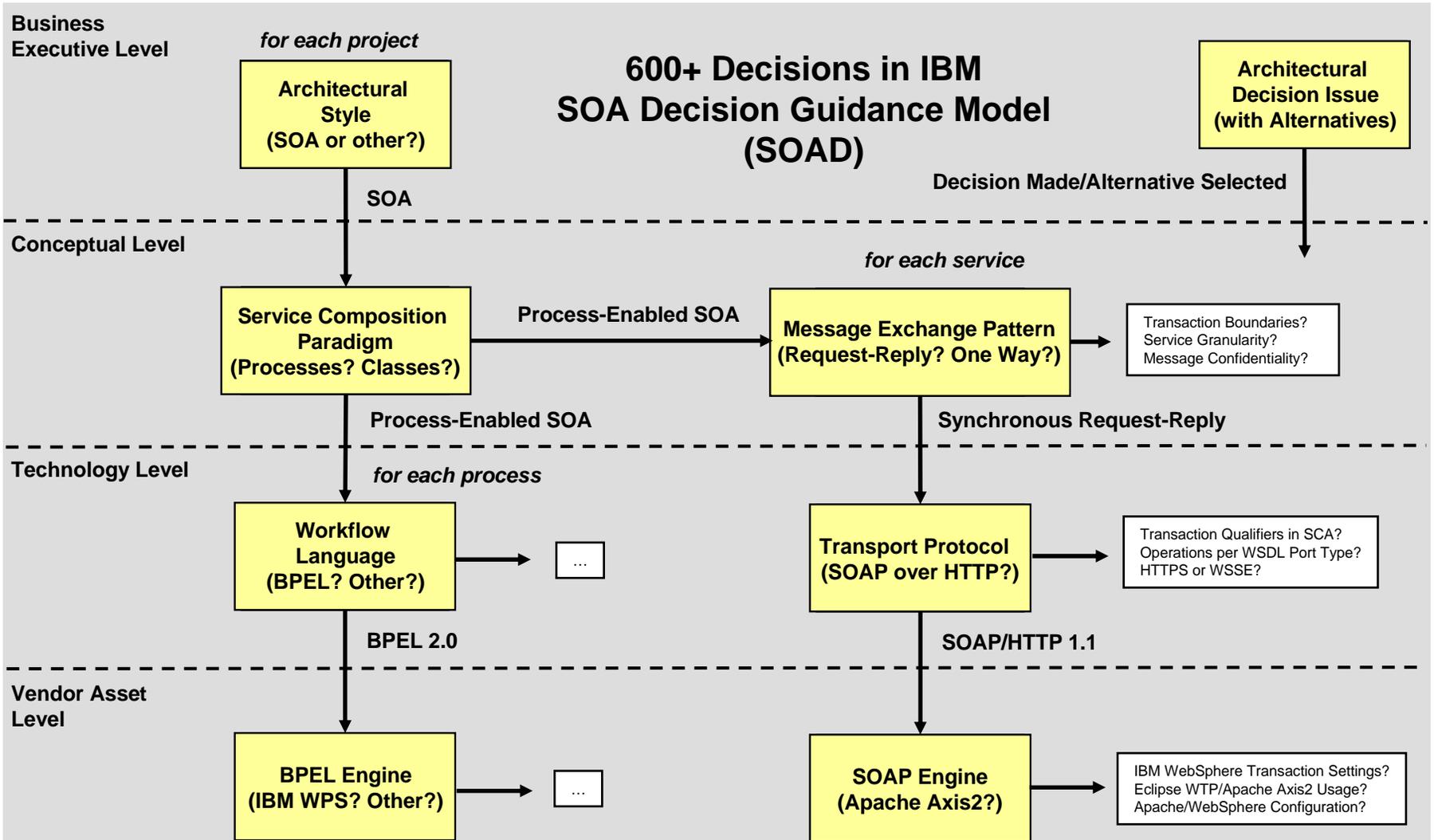
Solution Architect

“**Are we the first** IBM client that implements a process-based SOA with BPEL? If not, I’d love to have a look at the architecture design documentation from the previous projects. Which detailed **design issues** did the team encounter? Which **patterns and technology alternatives** did they consider? Why did they decide for the ones they eventually picked? Did they work? Do they have other **best practices** to share?”

“This **decision-centric approach** to **knowledge sharing, architecture design, and reviewing** worked really well. In each workshop, we looked at selected issues to be resolved, which apparently came from some reusable asset (called **guidance model** if I remember correctly). We based the selection (called **tailoring**) on various factors such as project scope, risk and cost, and amount of experience in the local team. The **issues come with alternatives that are known to have worked** on previous projects, **and with links to additional issues** to be investigated. Without this knowledge base, I would have had no idea that I have to worry about the **system transaction boundaries** inside the business processes and the underlying Web services, let alone where in the IBM BPEL engine to configure these settings. Same for my developers by the way.”

“If we succeed with this project, I will **harvest and contribute** to the guidance model the additional issues that we encountered, the alternatives we chose, and the rationale for these decisions.”

# From AD Documentation to Active Method Guidance



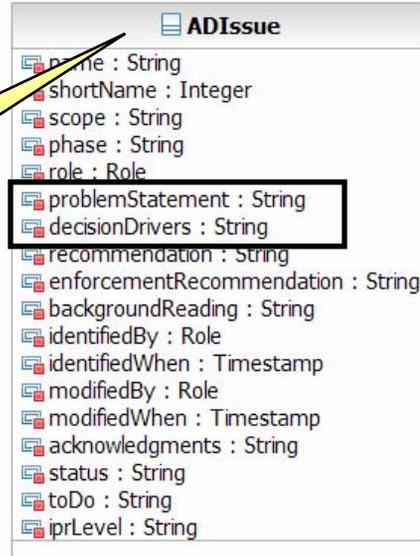
# Entity Types and Associations in UML Metamodel

## Guidance Model Decisions Required and Candidate Solutions

“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.”

### Problem and criteria

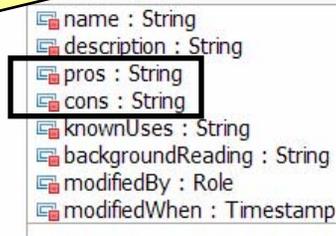


1

- isSolvedBy

\*

### ADAlternative



## Decision Model Decisions Actually Made on Projects

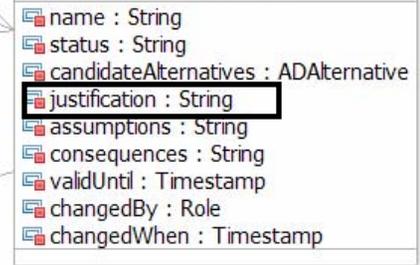
“We decided for the MVC alternative to resolve the web page flow issue because we gained positive experience with it on many similar projects.”

1

- hasOutcome

\*

### ADOutcome



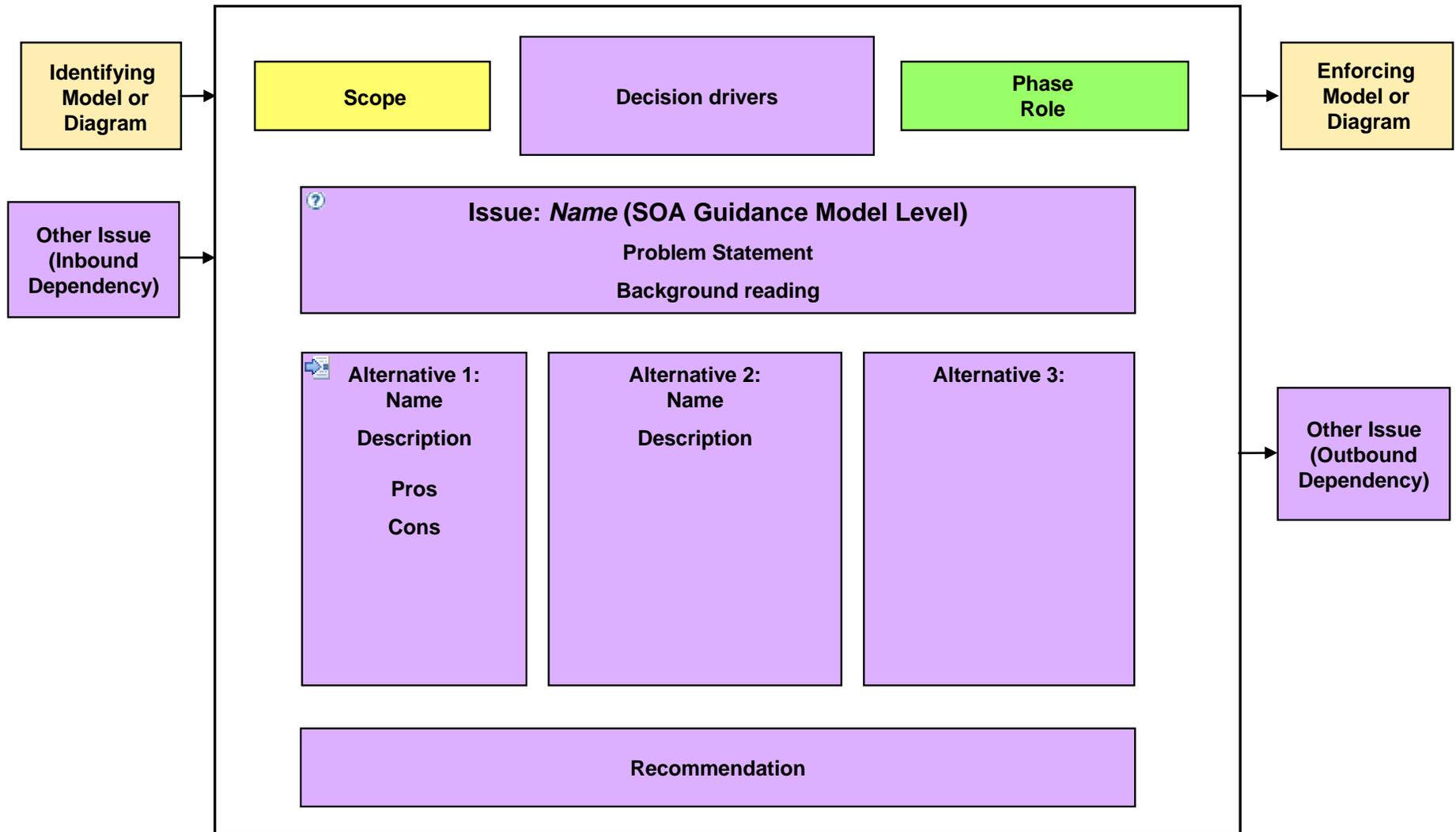
0..1

- chosenAlternative

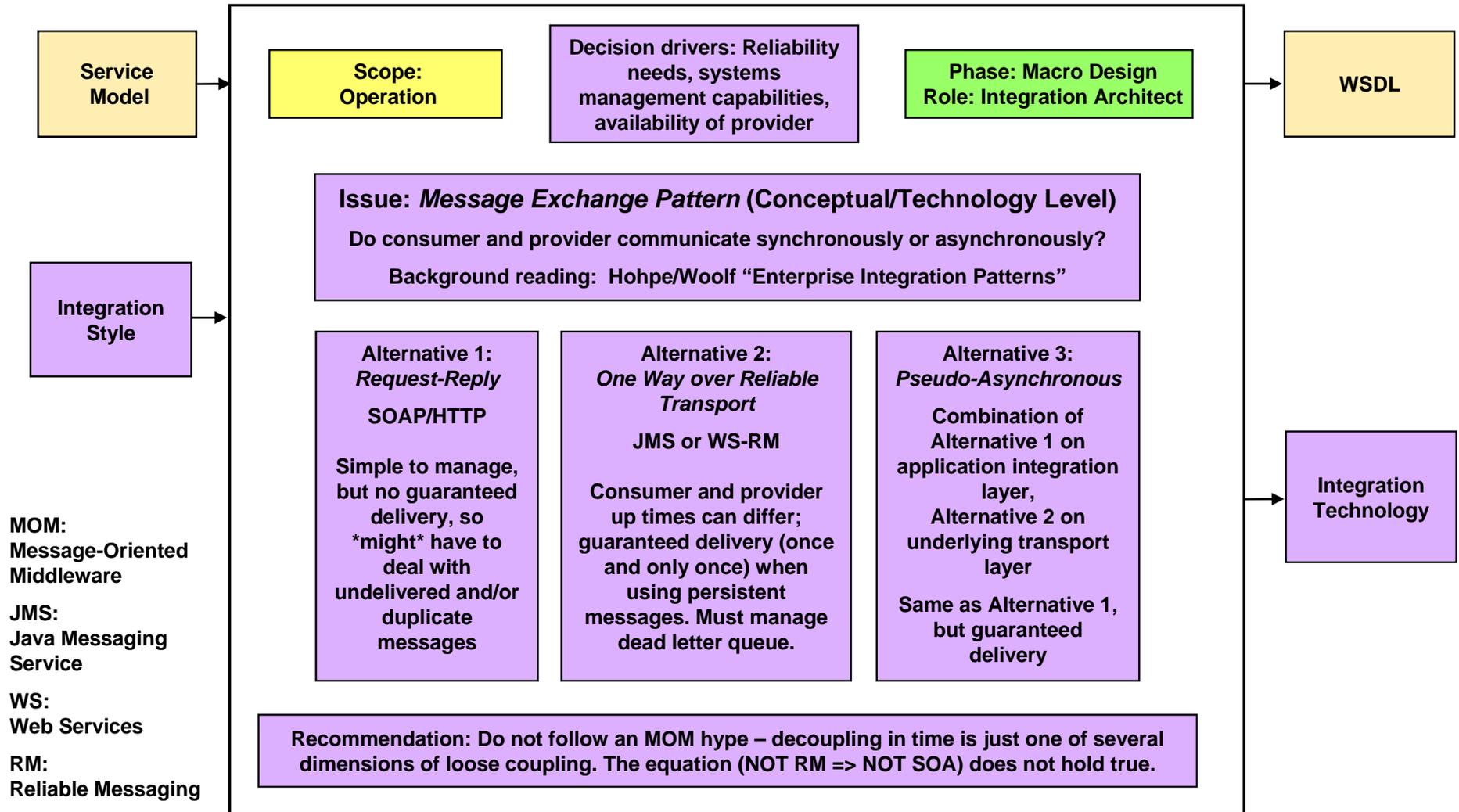
\*

**Chosen solution  
and justification**

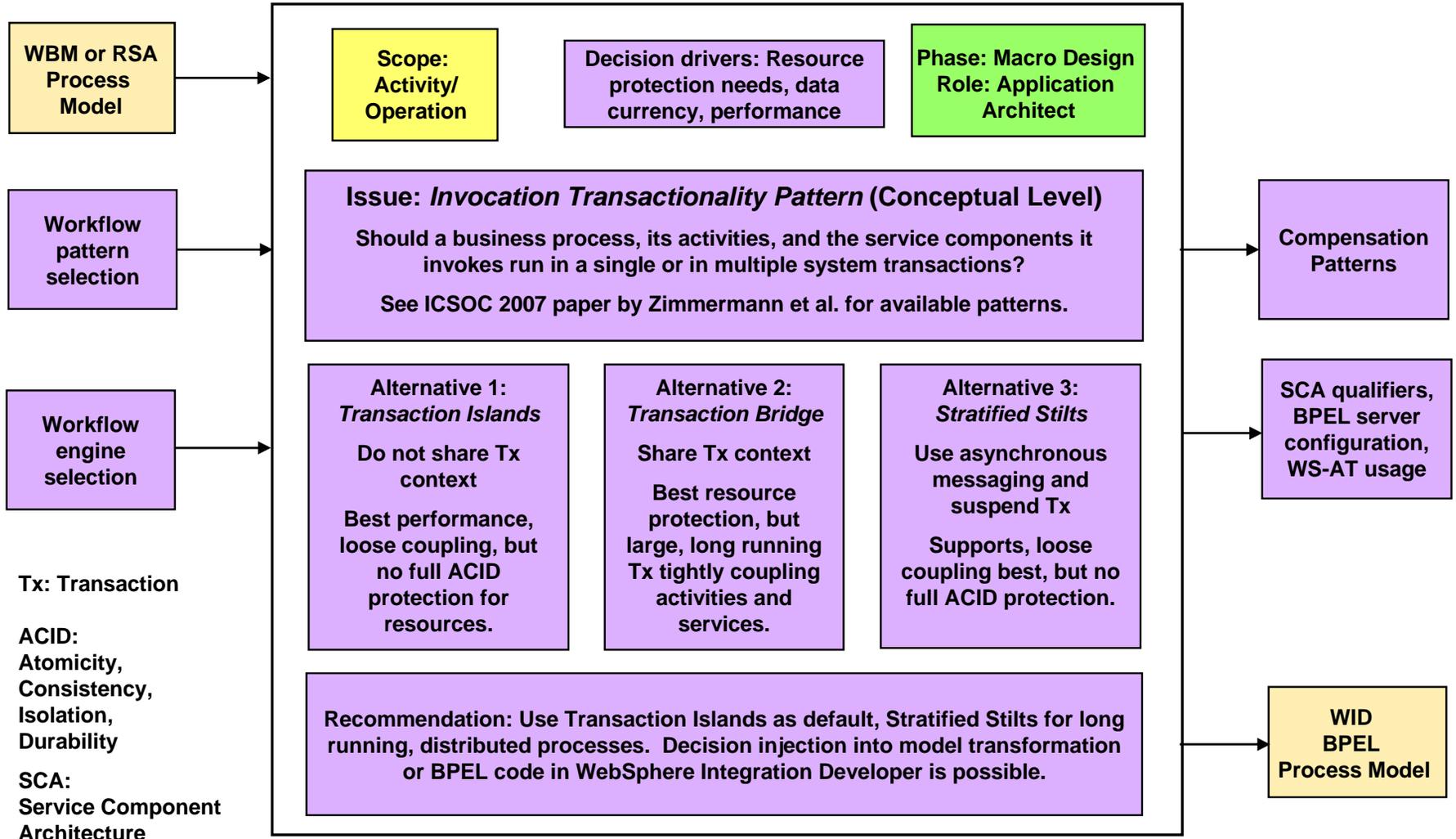
# Template Used to Present Issues and Alternatives



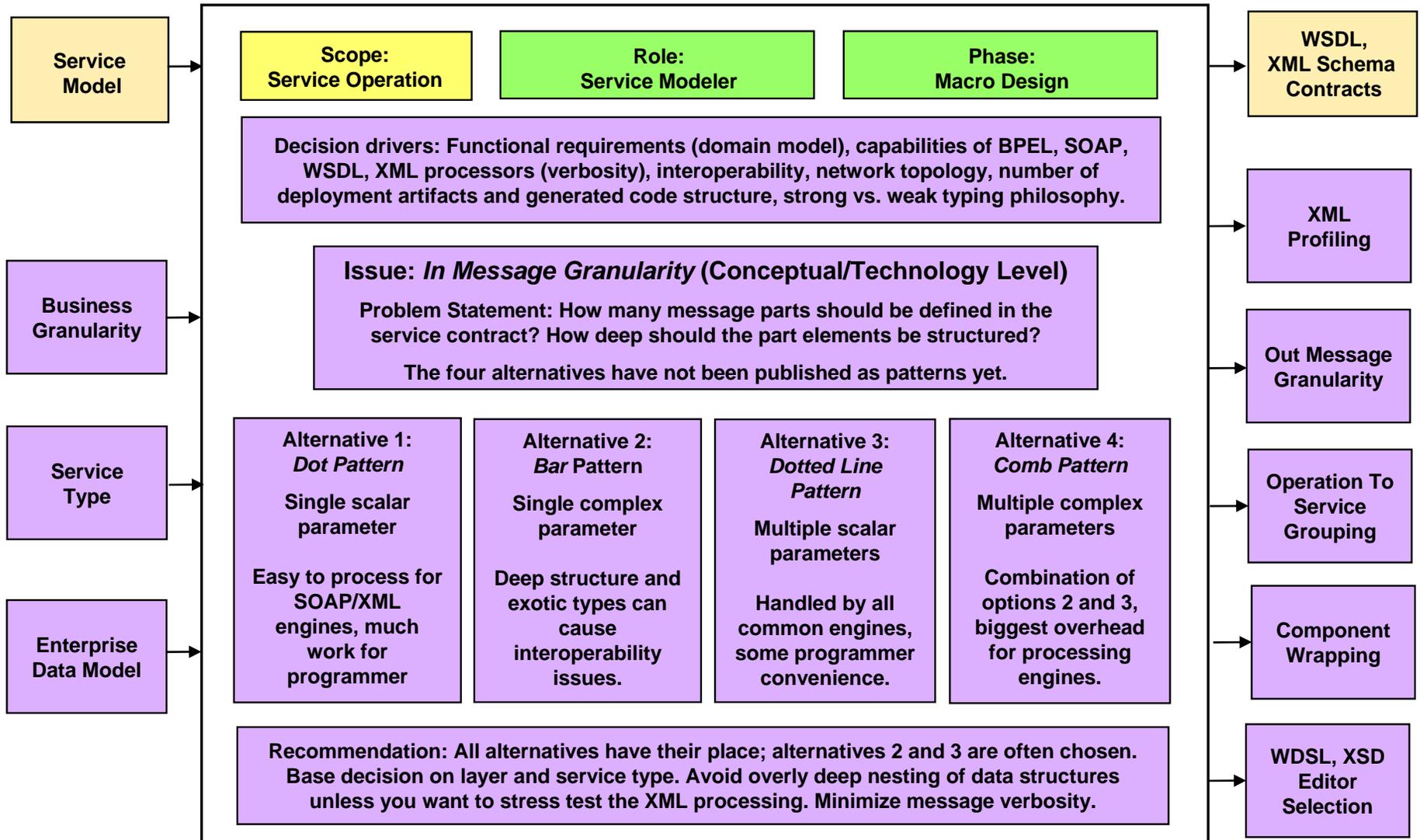
# Integration Layer Design Issue: *Message Exchange Pattern*



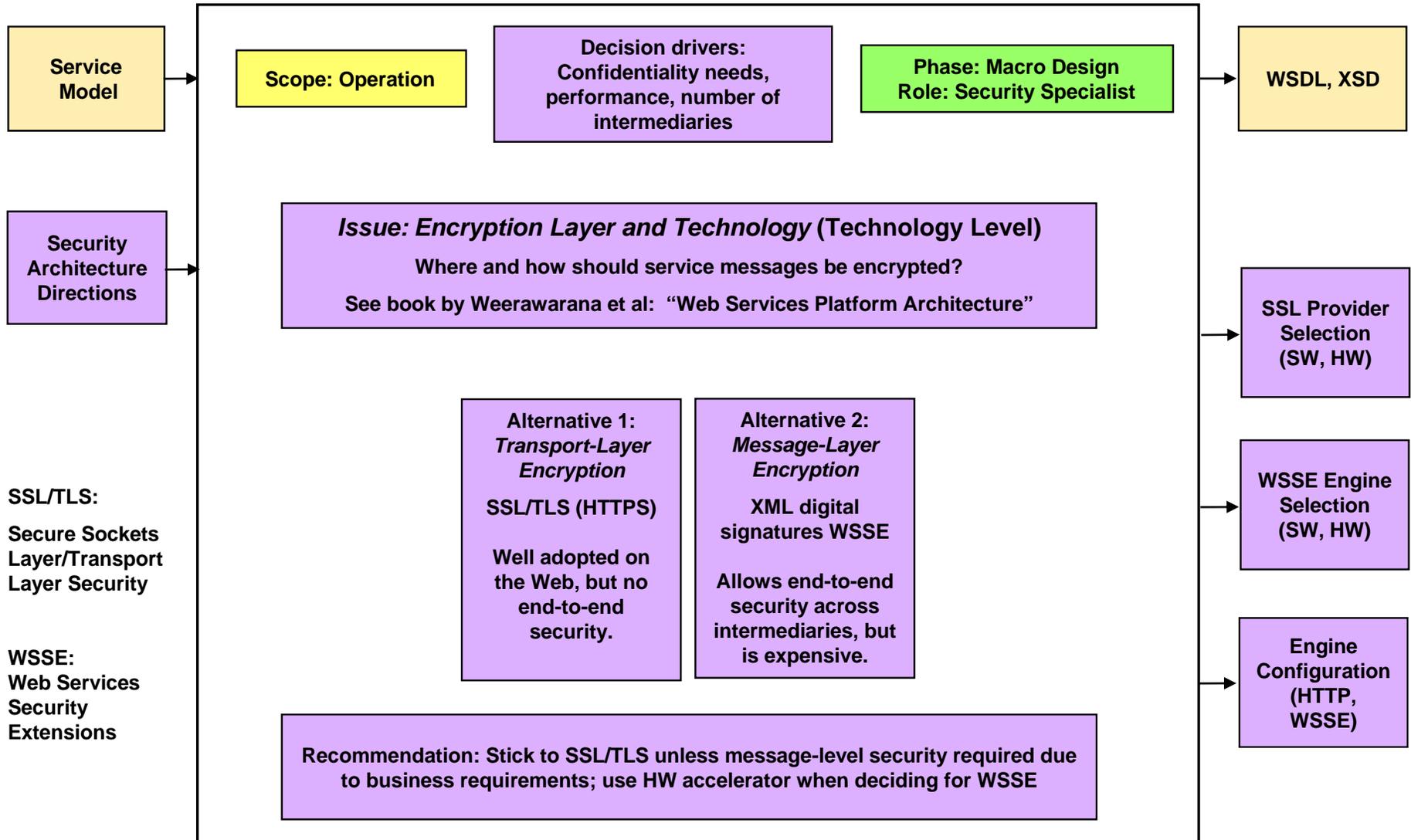
# Issues Dealing with *System Transaction Boundaries* Topic



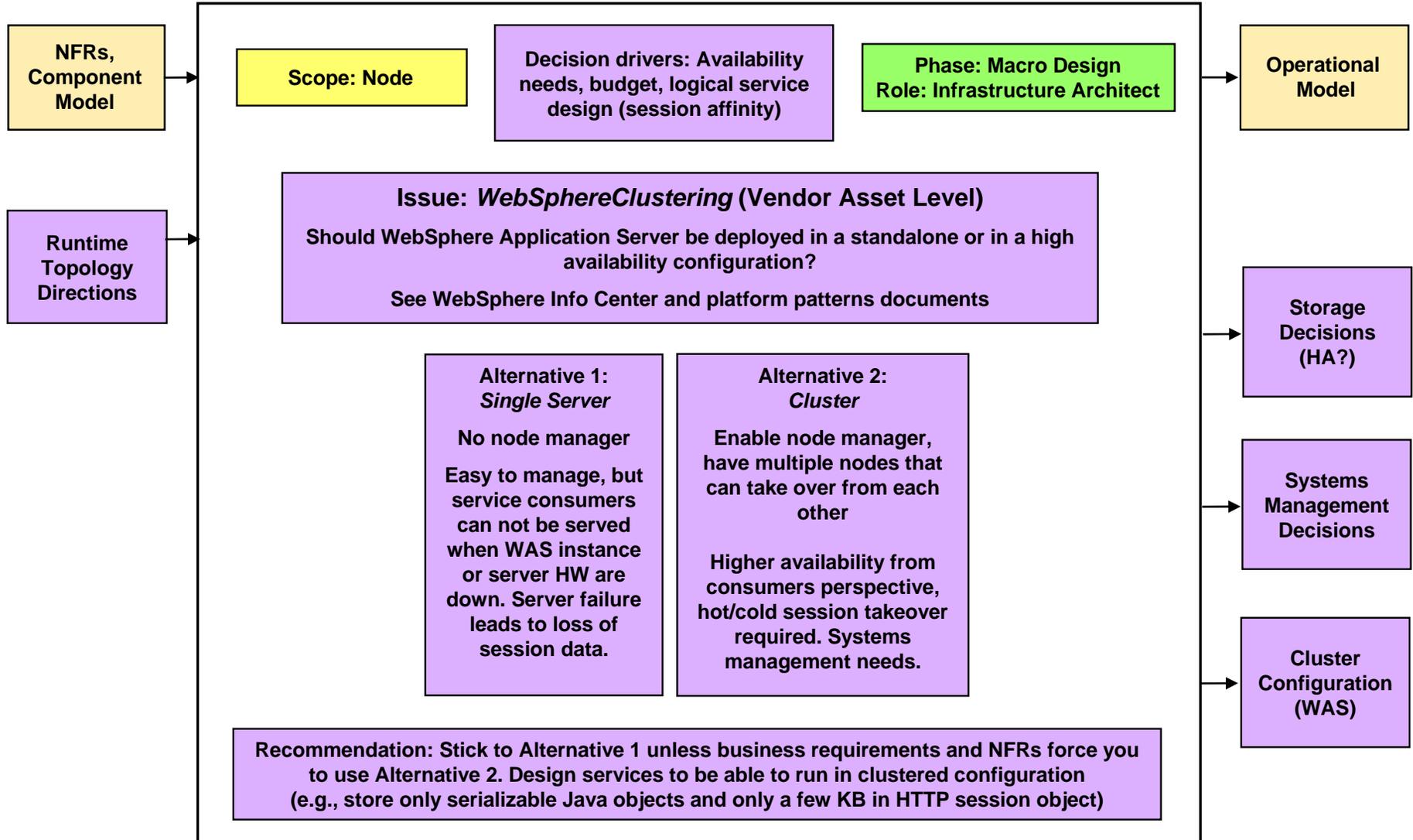
# Issues Dealing with *Service Granularity* Topic



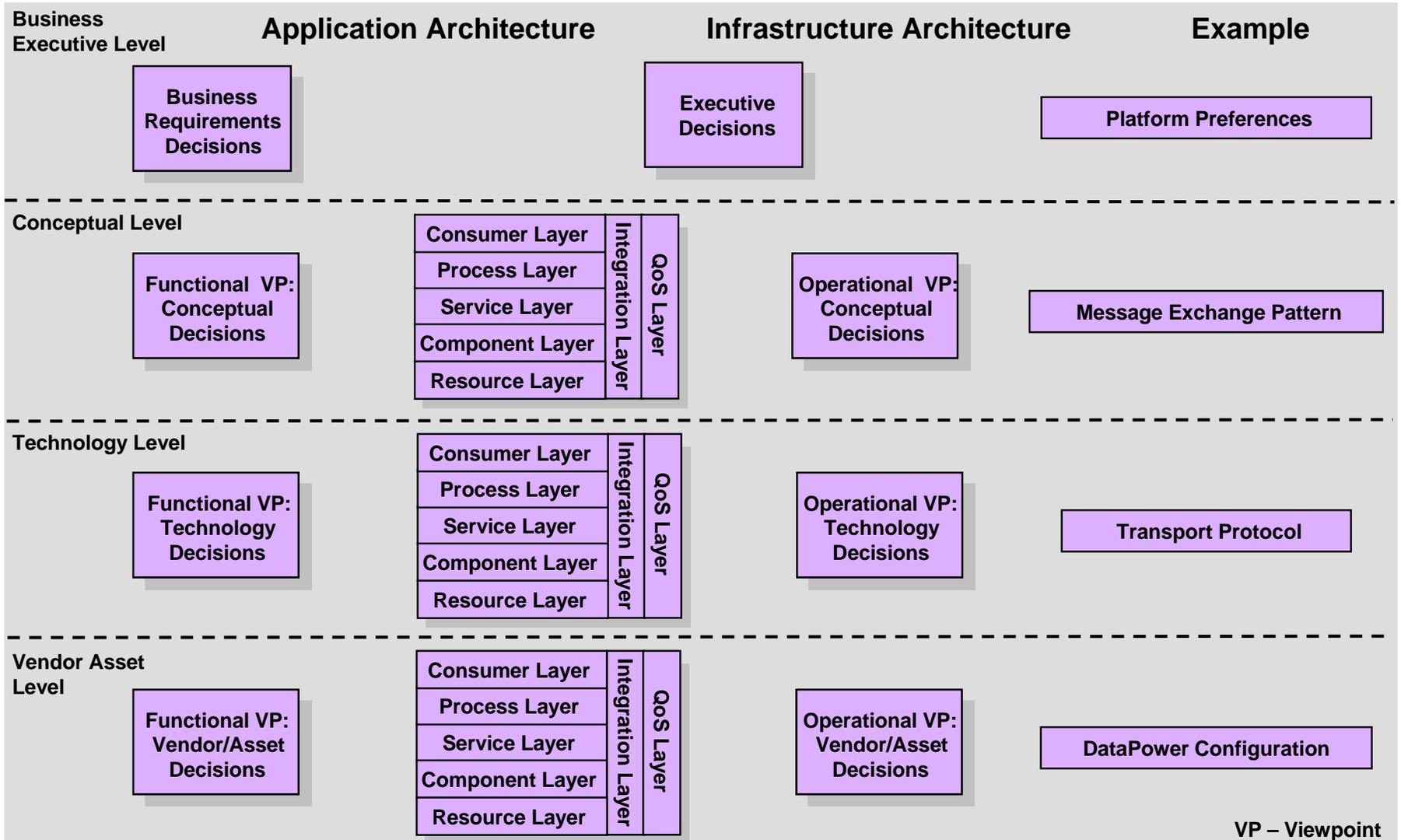
# Conceptual/Technology Issues about *Encryption*



# Vendor Asset Level *Clustering* Issues



# SOA Design Issues Organized by Levels and Layers

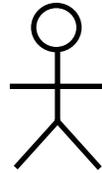


# Agenda

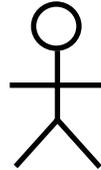
- Motivation
- Usage scenarios for architectural decision modeling
- Scenario 1: After-the-Fact Decision Capturing
- Scenario 2: Active Method Guidance
- **Scenario 3: Cross-Role Collaboration and Tool Integration**
- Emerging tool support (demo)
- Discussion and summary

# Scenario 3 (Cross Role): Desired State in the Case Study

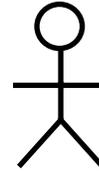
“I am not supposed to worry about technical details of the SOA solution too much, but quite a few **architecturally significant requirements** such as confidentiality and 24x7 availability have been stated. I will mark them as such. I’ll be interested to **trace** whether and how the architecture satisfies them.”



**Business Analyst**



**Solution Architect**



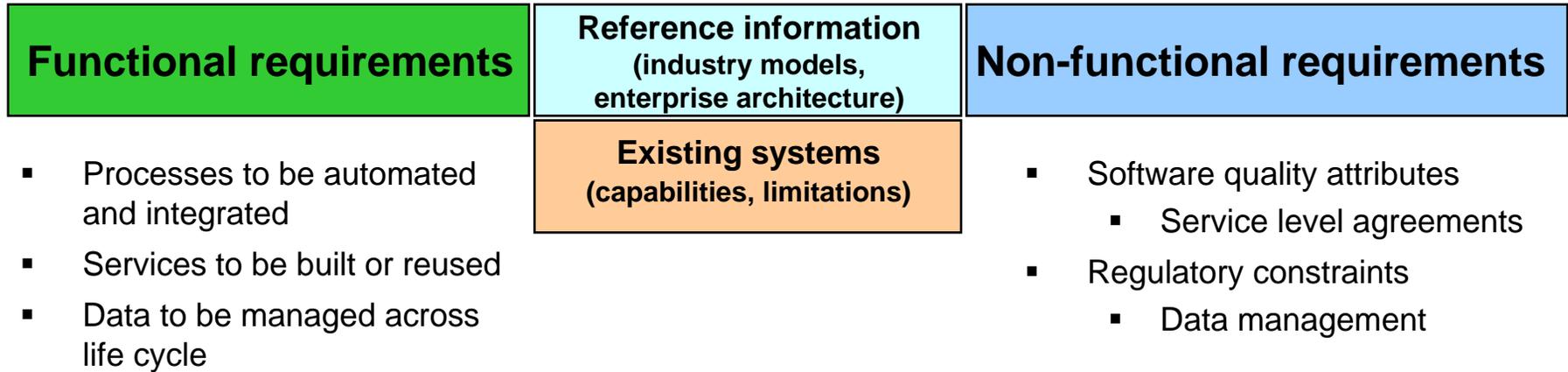
**Developer**

“The requirements people are using this new tool which supports **decision model tailoring and outcome instance creation**. So I already know that **two business processes** with about 100 Web services activities have to be realized. We’ll have to use **HTTPS** to connect Web channels to the system. We’ll also pick a **clustered topology**. This really accelerates the architecture design work.”

“In recent times, developers have criticized my architecture design as too high level and too difficult to implement. I am not always sure how to **enforce my architectural decisions** so that the implementation reflects them properly. With the new collaboration tooling, I can track them as **design work items** and link them to development tasks. Some of the enforcement can even be **automated** by code generators.”

“The new collaboration tooling has its goods and its bads from my perspective. On the positive side, I can always **find somebody** who knows about the design goals and decisions made so far. And we all are informed about **project health**. However, the architects are much closer involved now; I can no longer create and hide my own designs. And some of the routine **coding and configuration steps** were **taken over by model transformations**.”

# From Requirements to Architectural Decisions

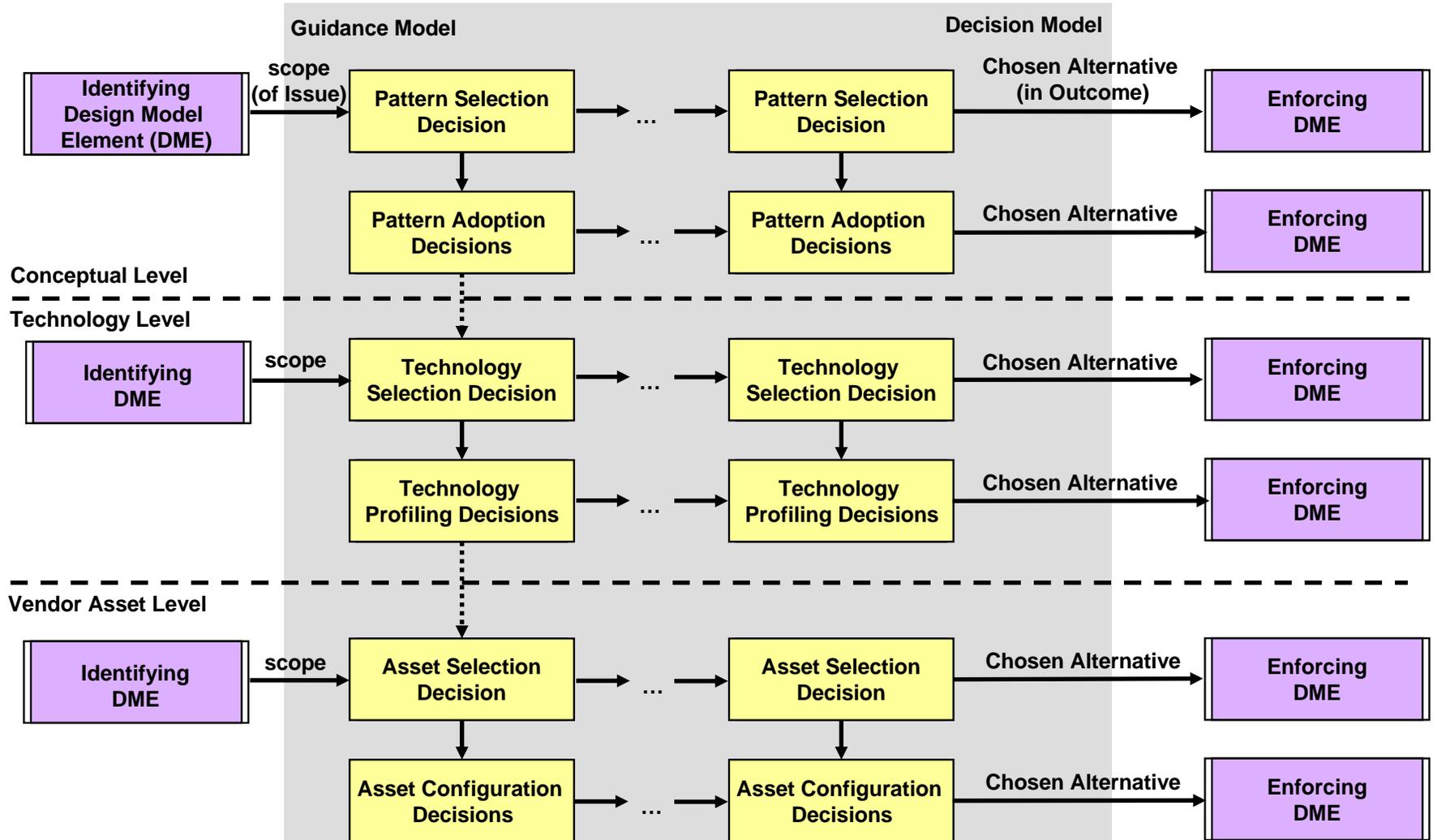


***How to build the processes/services and satisfy their quality requirements and constraints?***

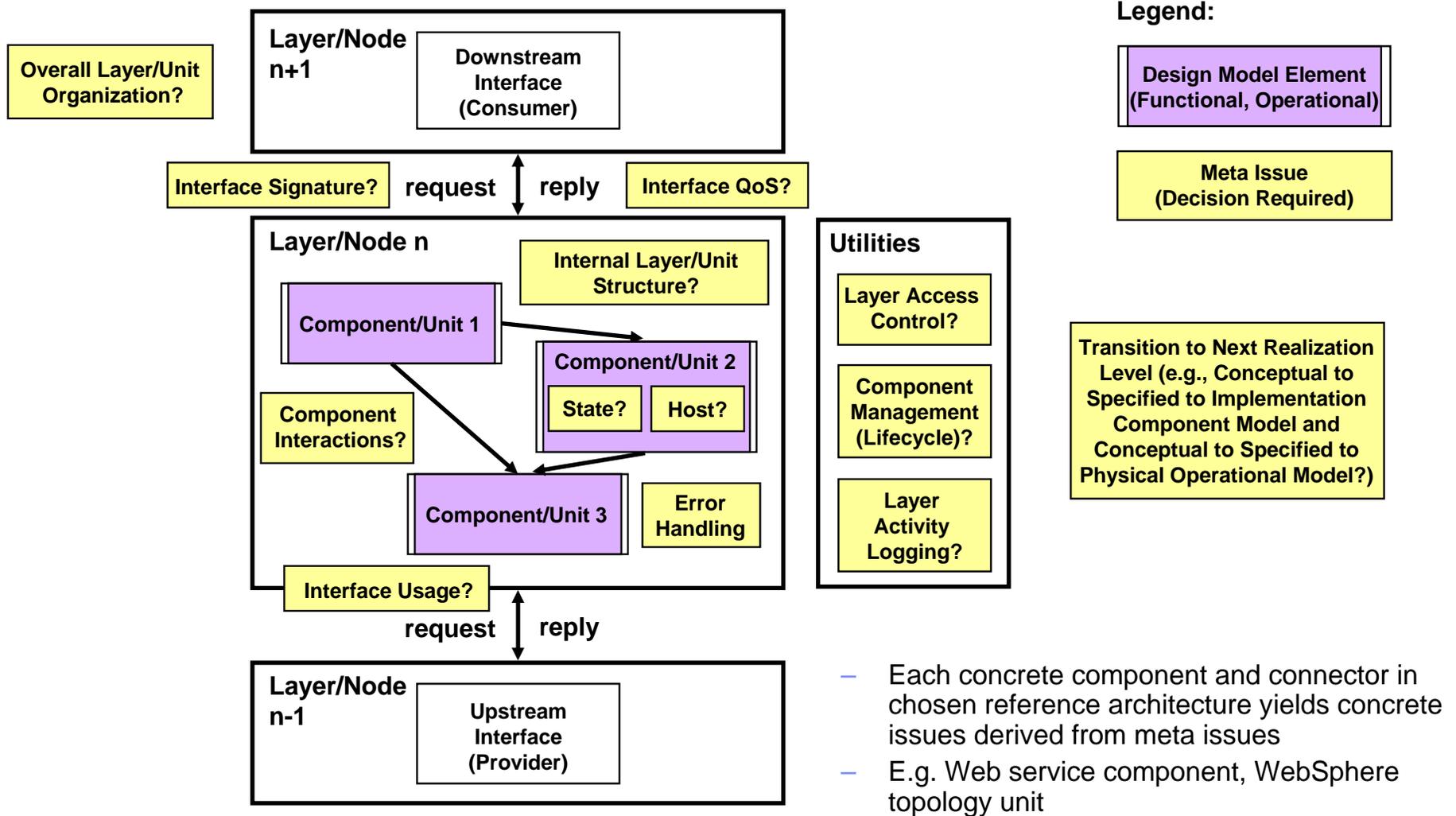
## **Architectural Decisions**

Recurring decisions carry *best practices* regarding *pattern adoption*

# Active Method Guidance (Meta Issue Level): Handshakes between Design Models and Decision Models



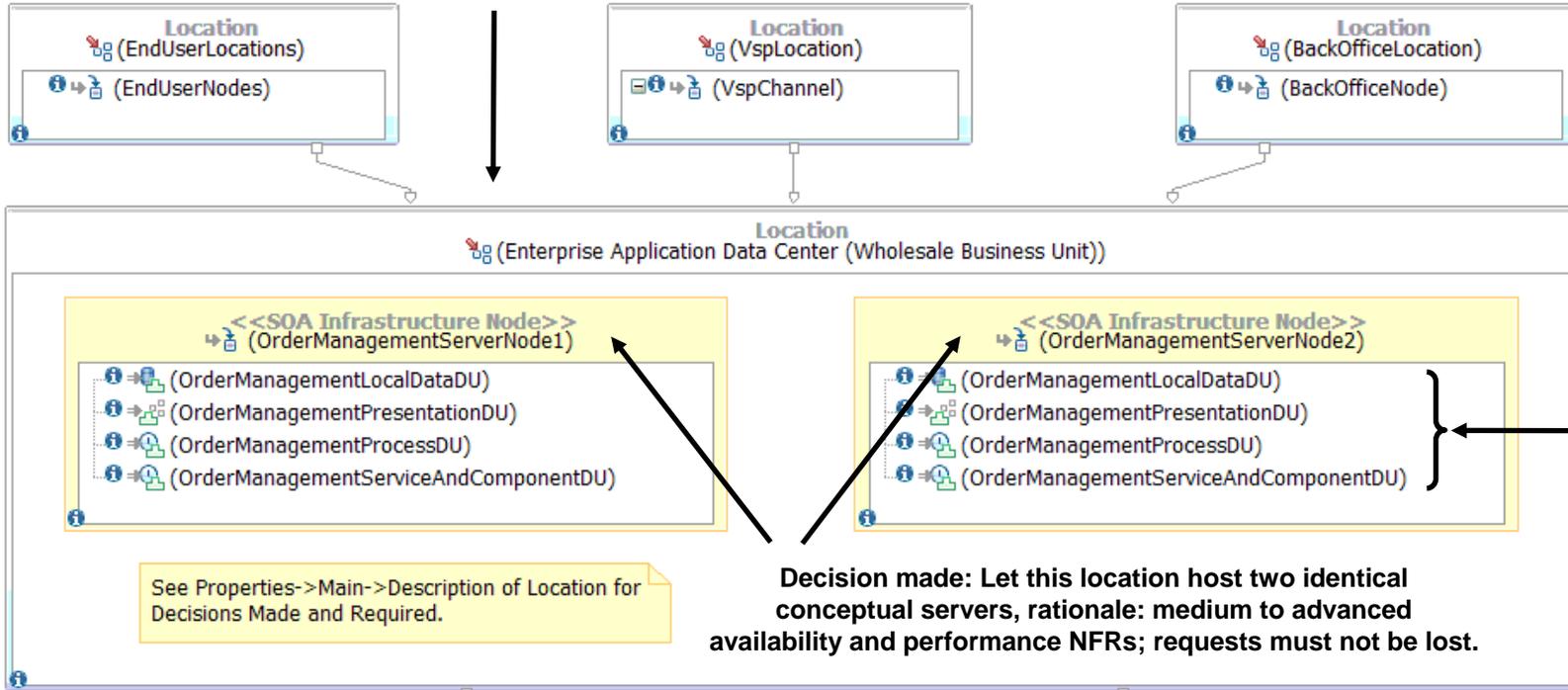
# (Meta) Issue Identification in High-Level Component Models



- Each concrete component and connector in chosen reference architecture yields concrete issues derived from meta issues
- E.g. Web service component, WebSphere topology unit

# Conceptual Operational Model (COM) and Rational Software Architect Topology: Decisions Made/Required

**Decision made: Model a single location for entire order management system, rationale: physical separation/mirroring not required**



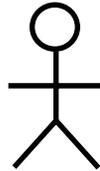
**Decision made: Host four Deployment Units (DUs), rationale: UML component layering, best practice stated in SOA Infrastructure Reference Architecture**

**Decisions now required (identified in this model): Transition to Specified OM, specification of realization links**

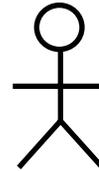
**More decisions required:**  
 Realization links for all nodes, DUs  
 Network protocols for all external channels  
 Security settings, systems management

## Scenario 3: Cross Role Collaboration (Continued)

“The order management project was a big success. The team learned a lot about the design of process-driven SOA. We plan to apply this architectural style on several new projects; hence, we would like to **establish architectural principles** such as loose coupling and **share best practices** throughout the company. To make this actionable, I will **upgrade the decision model from the project to a company-wide guidance model**. The issues in the guidance model have to be resolved explicitly on each and every solution development project. If any alternatives are chosen that are not in the guidance model, enterprise-level approval is required.”



**Enterprise  
Architect**



**Solution Architect**

“I was initially skeptic about this company-wide guidance model: We have enough **policies and principles** already. And in the end, the business requirements overrule everything. Finally, who will populate the model and keep it up to date? None of the knowledge management approaches we have tried worked in the long term.”

“I have to admit that they managed to find a **balance** between **freedom-of-choice** and **freedom-from-choice**. The issues and alternatives in the model are all relevant, easy to locate and to digest, but far away from being trivial. With the new tooling, it is really simple to submit desclets from the project to the knowledge harvesting process.”

# Vision: Enterprise-Wide Guidance Model

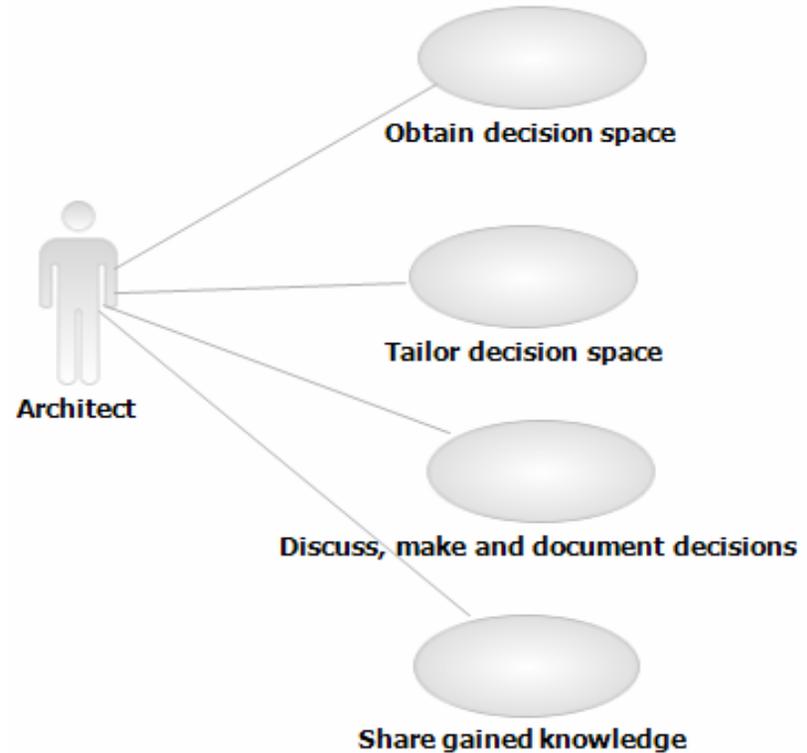
- Enterprise architecture group owns and maintains guidance model
  - Input comes from solution architects on development/integration projects
  - Quality assured, aligned with enterprise IT strategy
- Does not mandate a particular architecture, but frames design work
  - Recommend certain alternatives:
    - E.g. “always use document/literal SOAP”
  - Ban others:
    - E.g. “no open source assets can be used due to open legal issues”
  - Finding a balance between freedom-of-choice and freedom-from-choice
- Allows project teams to share lessons learned and best practices
  - Actionable enterprise architecture
  - Enterprise architects perceived as friends, not foes

# Agenda

- Motivation
- Usage scenarios for architectural decision modeling
- Scenario 1: After-the-Fact Decision Capturing
- Scenario 2: Active Method Guidance
- Scenario 3: Cross-Role Collaboration and Tool Integration
- **Emerging tool support (demo)**
- Discussion and summary

# Architectural Decisions Knowledge Tools Project (Rational/RES)

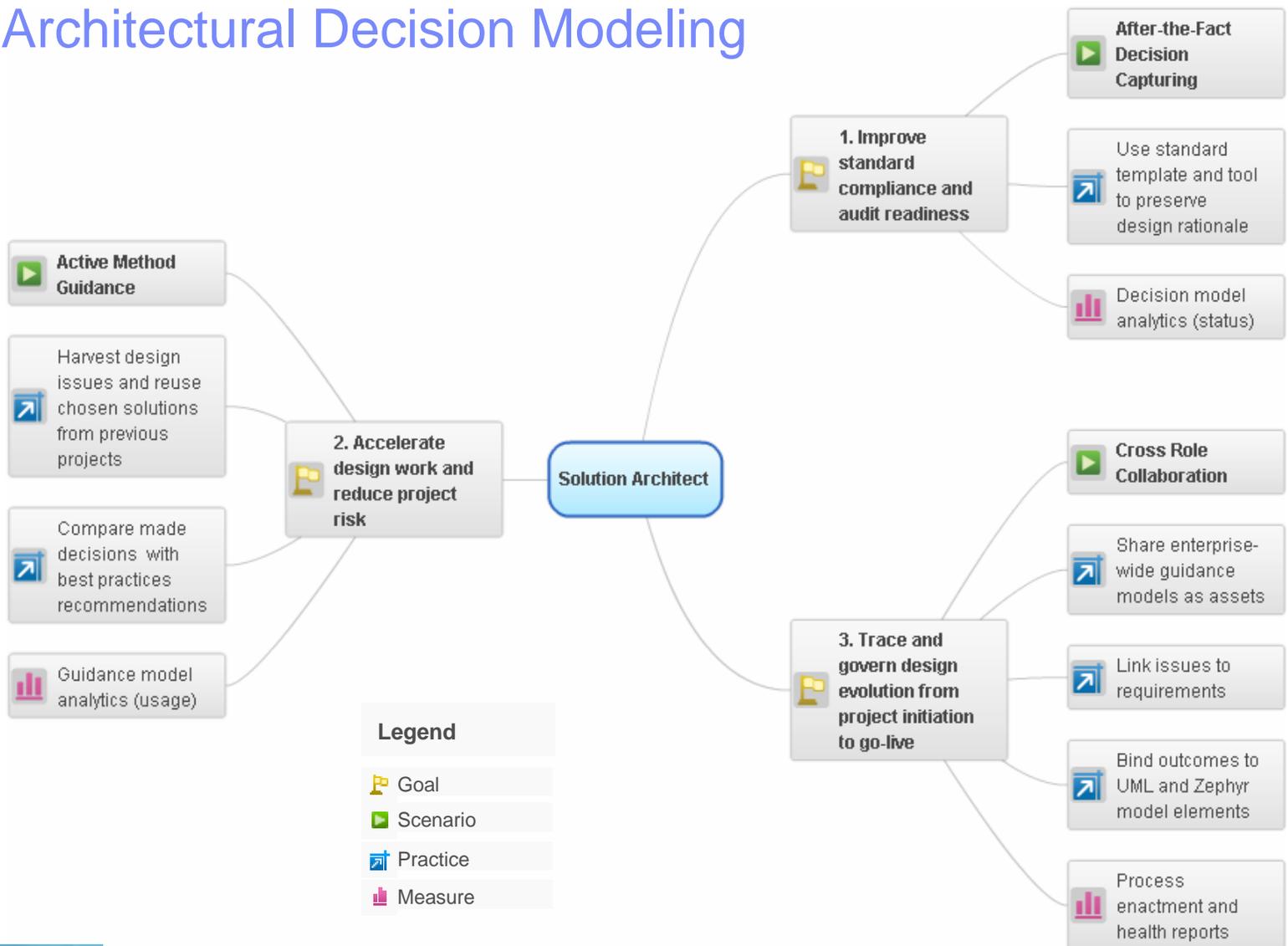
- Regulatory compliance
  - E.g., maturity models
- Collaboration
  - In geographically distributed teams
- Reuse
  - Of already gained knowledge
- Other required features:
  - Import and export
  - Searching and filtering
  - Dependency management
  - Report generation



# Agenda

- Motivation
- Usage scenarios for architectural decision modeling
- Scenario 1: After-the-Fact Decision Capturing
- Scenario 2: Active Method Guidance
- Scenario 3: Cross-Role Collaboration and Tool Integration
- Emerging tool support (demo)
- **Discussion and summary**

# Value of Architectural Decision Modeling



# Summary and Discussion

- Architectural decision making is a key responsibility of IT architects which is often underestimated and underrepresented in existing methods and tools.
- In SOA design, many decisions recur. This makes it possible to reduce the documentation effort and to share architectural decision knowledge including best practices (design acceleration and quality assurance).
- Prototypical tool support for decision modeling with reuse is available.
- We would like to hear from you now...
  - ... are the presented scenarios, concepts, and tool features useful and usable?
  - ... would you have additional requirements, e.g. collaboration and integration needs?
- Email: [olz@zurich.ibm.com](mailto:olz@zurich.ibm.com), website: [www.soadecisions.org](http://www.soadecisions.org)