



Business Integration Technologies

Architectural Decisions and Patterns for Transactional Workflows in SOA

ICSOC 2007
September 18, 2007

Olaf Zimmermann
Jonas Grundler
Stefan Tai
Frank Leymann

Agenda

- SOA Decision Modeling overview
 - Meta model
 - Levels and layers
 - Decision scoping and dependencies
- Decision Model for transactional workflows in SOA
 - Step (a): Identification of conceptual decisions and patterns
 - Step (b): Decomposition by layer
 - Step (c): Refinement from conceptual to technology to vendor/asset level
- Conclusions and future work

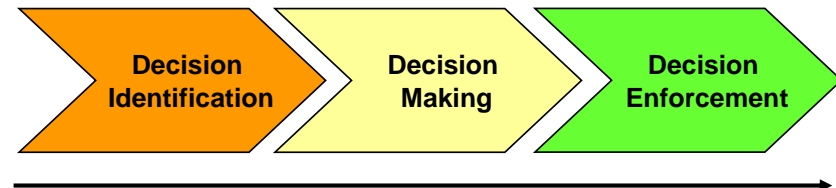
Agenda

- **SOA Decision Modeling overview**
 - **Meta model**
 - **Levels and layers**
 - **Decision scoping and dependencies**
- **Decision Model for transactional workflows in SOA**
 - Step (a): Identification of conceptual decisions and patterns
 - Step (b): Decomposition by layer
 - Step (c): Refinement from conceptual to technology to vendor/asset level
- **Conclusions and future work**

SOA Decision Modeling (SOAD) in a Nutshell

- Technical decision *identification* and *making* often disconnected from *enforcement* (cost, quality, risk impact!)
 - One-of-a-kind design of transaction, security, and reliability concerns
 - Gaps between logical and physical models, LOBs, presales and post sales

- SOAD provides method and tools for *end-to-end* decision maker collaboration and best practices exchange:
 - SOA decisions, patterns, and policies continuum
 - Reusable and machine-readable decision models
 - Web 2.0 tool support



SOA Decision Modeling

SOAD Project | WikiPage | User Settings | Help | View | Mash up | Collaborate | Debug

Ead-04 Transaction Management Pattern

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" by 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

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

Alternatives	DTP Transaction Islands (default)
1 DTP Transaction Islands (de	
2 DTP Transaction Bridge	
3 QTP Asynchronous Stills	

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

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

Functional domain

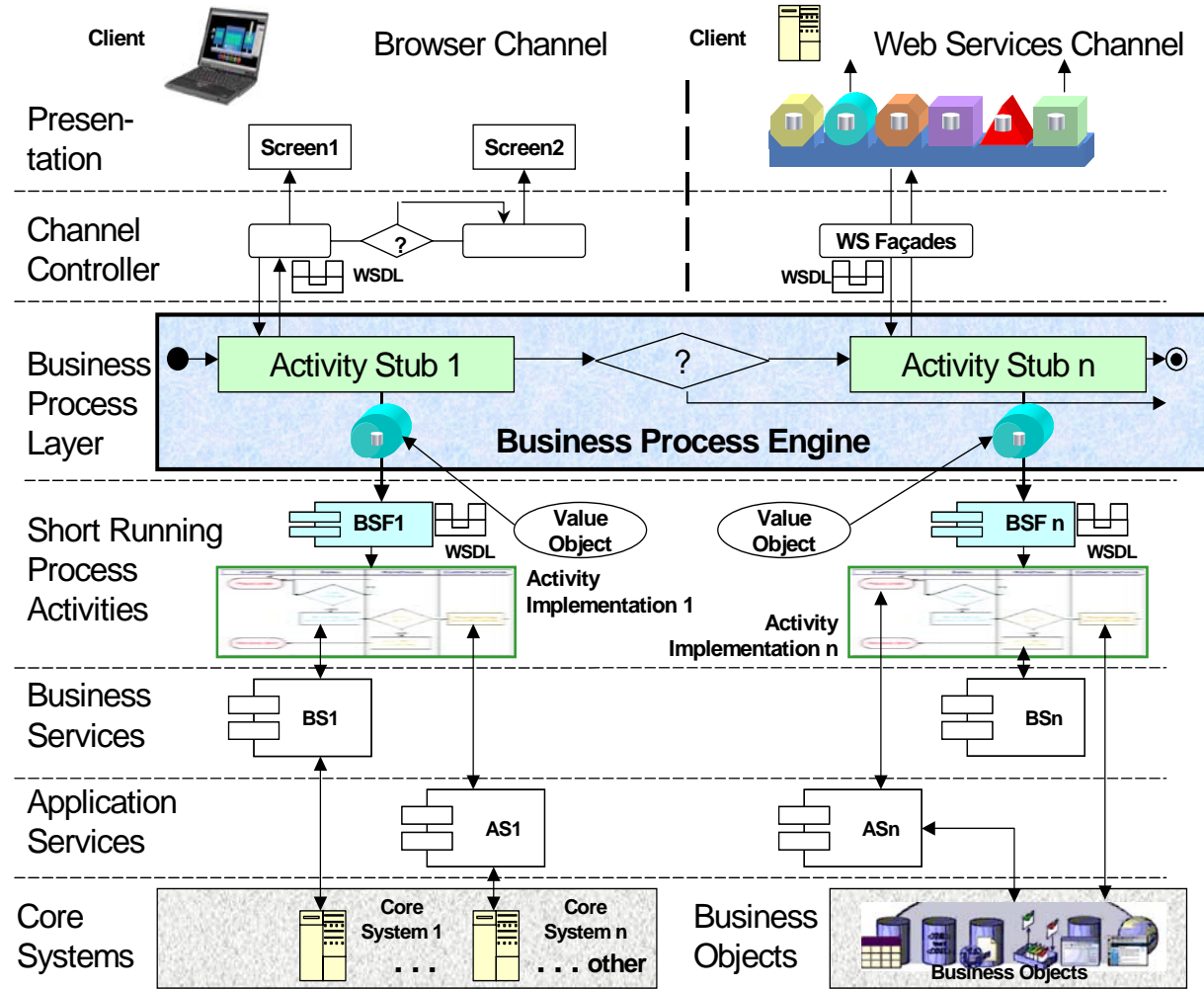
- Order entry management
- Two business processes: new customer, relocation

Main SOA drivers

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

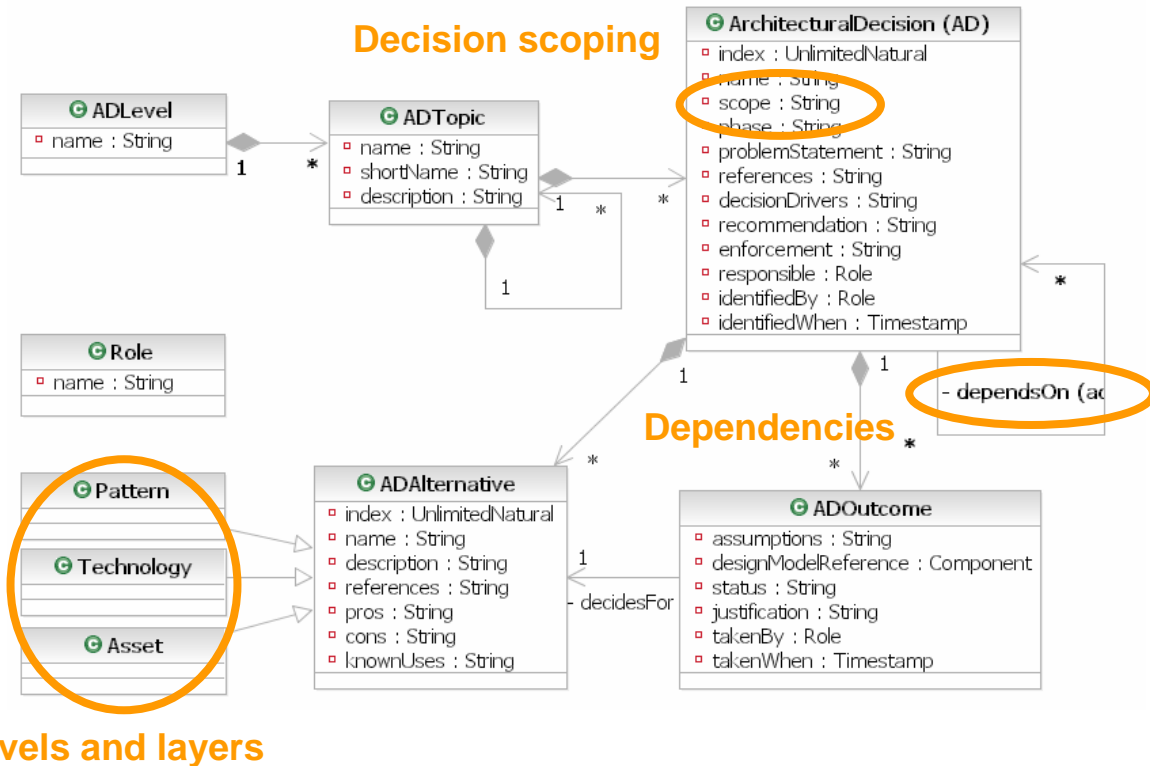
Service composition

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



Architectural Decision Modeling for Reuse [QOSA]

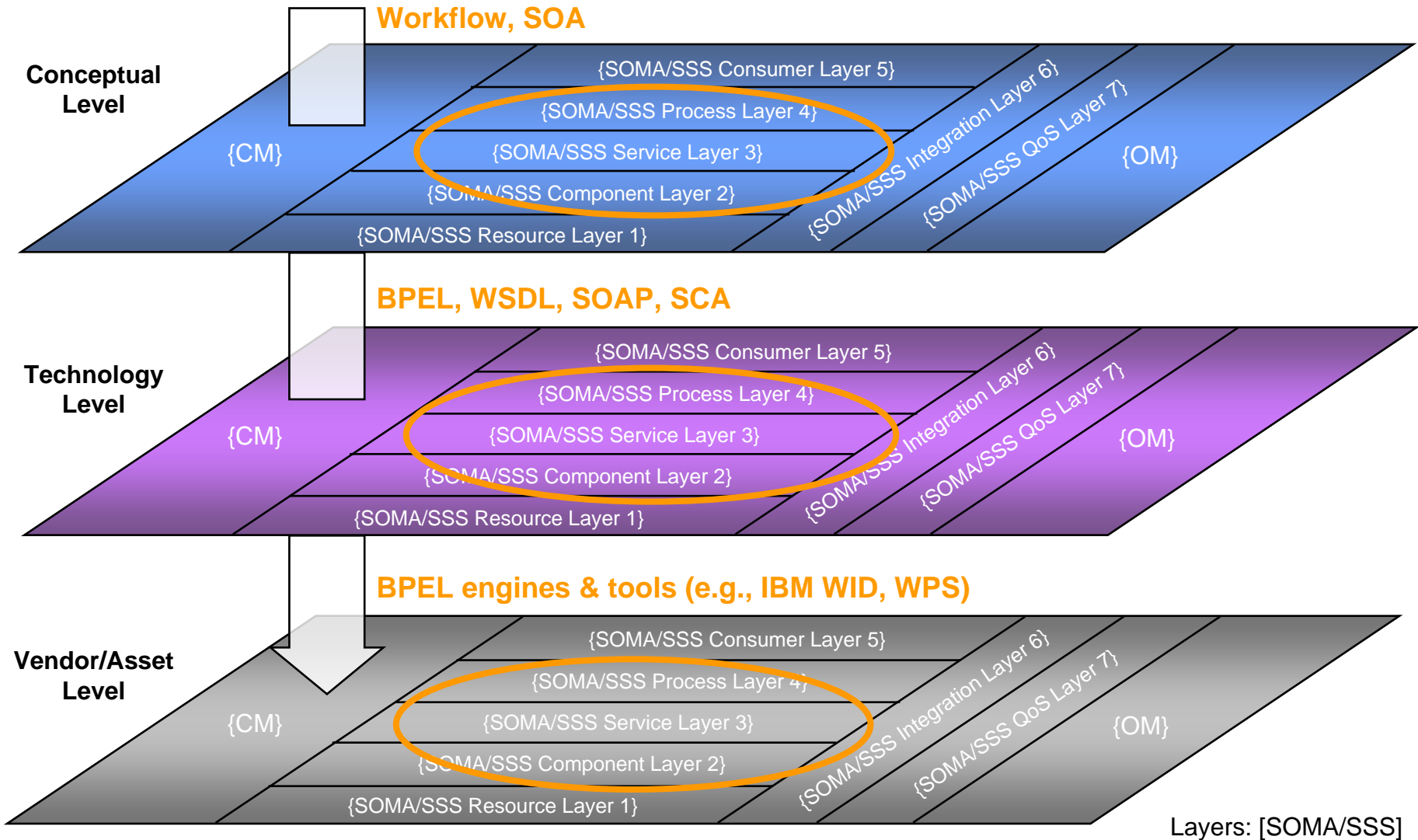
- Architectural decisions capture key design issues and the *rationale* behind a chosen solution:
 - Conscious design decisions concerning a software system as a whole, or one or more of its core components
 - With an impact on non-functional characteristics and quality factors



- Documenting architectural decisions *should* be state of the practice
- Using pre-populated AD *models* in an active, guiding role is a novel approach
 - Our work is based on existing work and ongoing research, e.g., [Kruchten 2006]

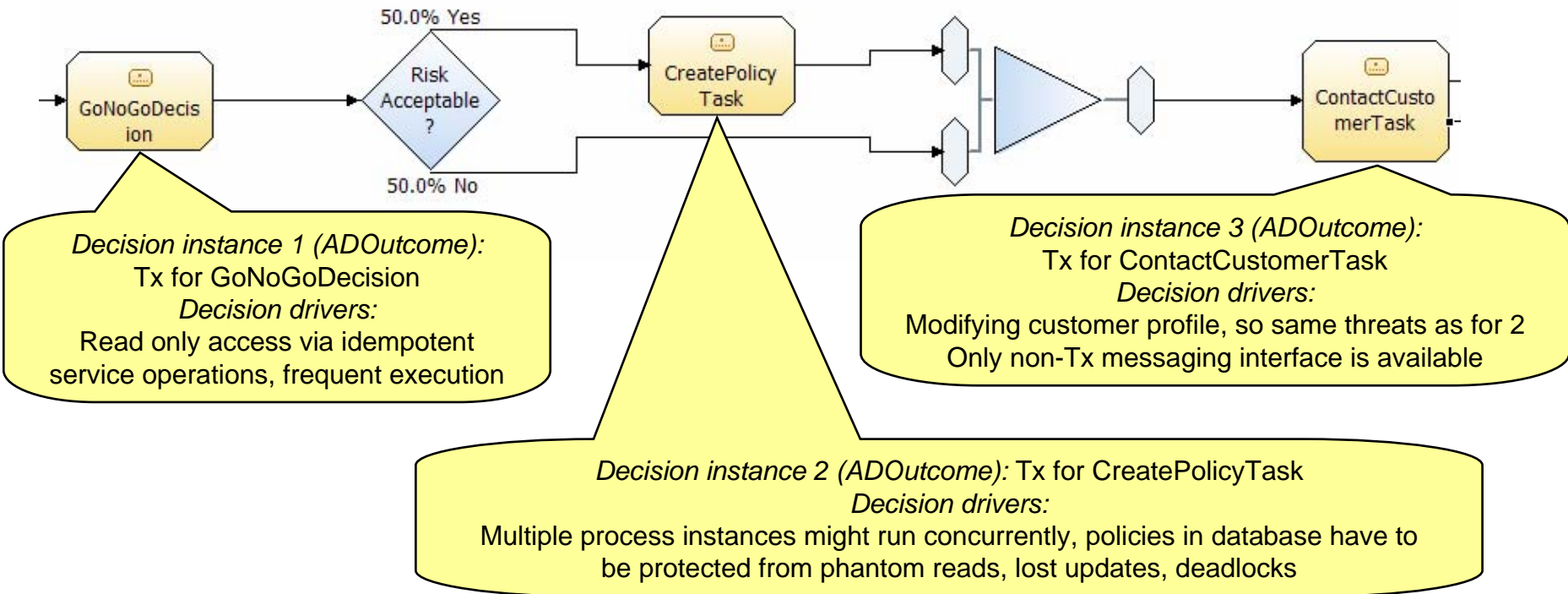
3*(1+7+1) Top-Level ADTopic Groups

{CM} Component Modeling
 {OM} Operational Modeling Decisions



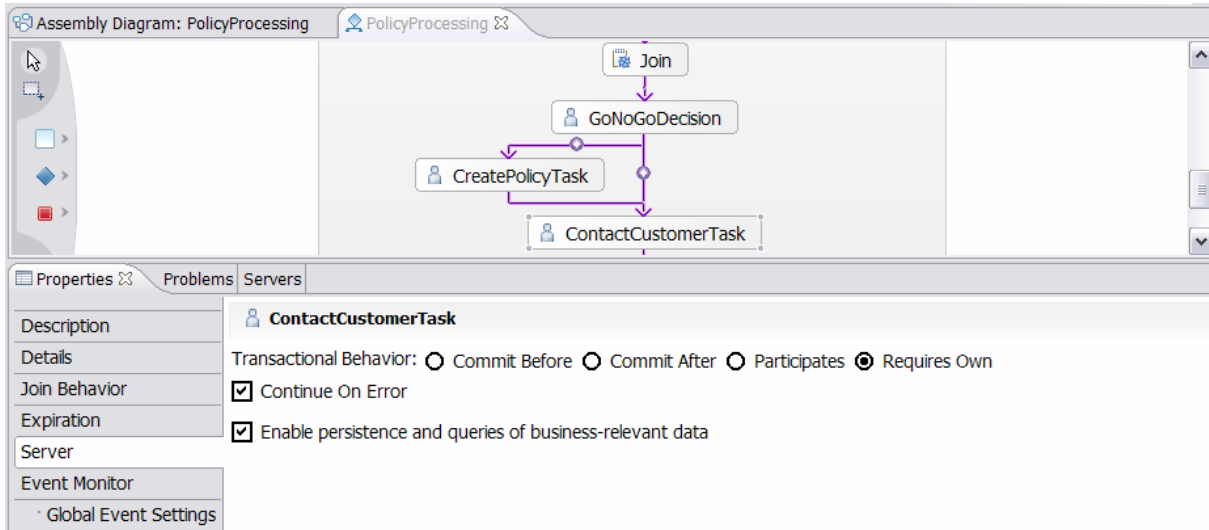
Key Decision on Layer 2, 3, 4: Activity Transactionality

Decision type (AD): Transactionality (Tx), *Decision scope:* Task/Service Operation
Decision drivers: Resource protection needs, data currency, performance, legacy interfaces



On the IT design level, the process modeler defines the transaction boundaries for all invoke activities within a process, as well as the underlying architectural layers

Commercial Tools: Transactionality on BPEL/SCA Level



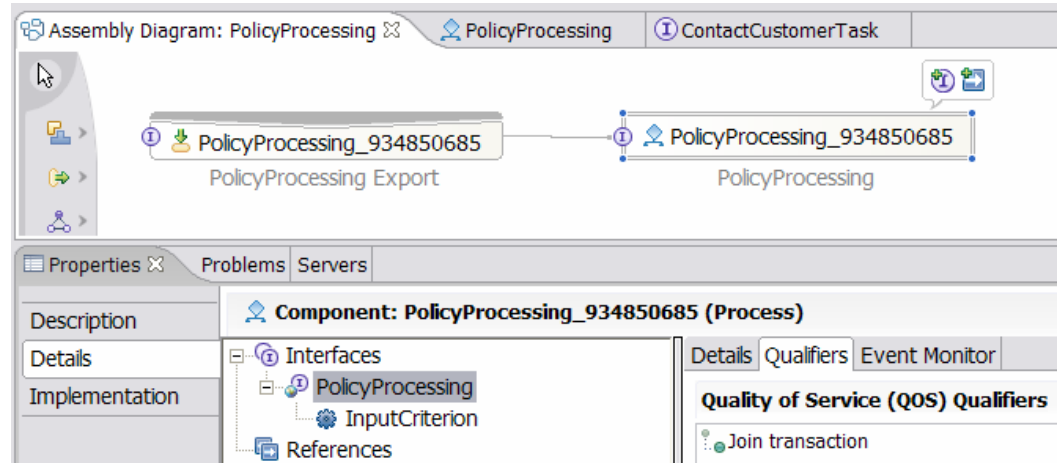
E.g., BPEL editor in IBM WebSphere Integration Developer (WID):

Proprietary radio button per invoke activity: “TransactionalBehavior”

Assembly editor in WID:

Proprietary SCA qualifiers for SCA reference, import interface, implementation, such as “Join transaction”

Plus SOAP engine, WSAT, JMS configuration

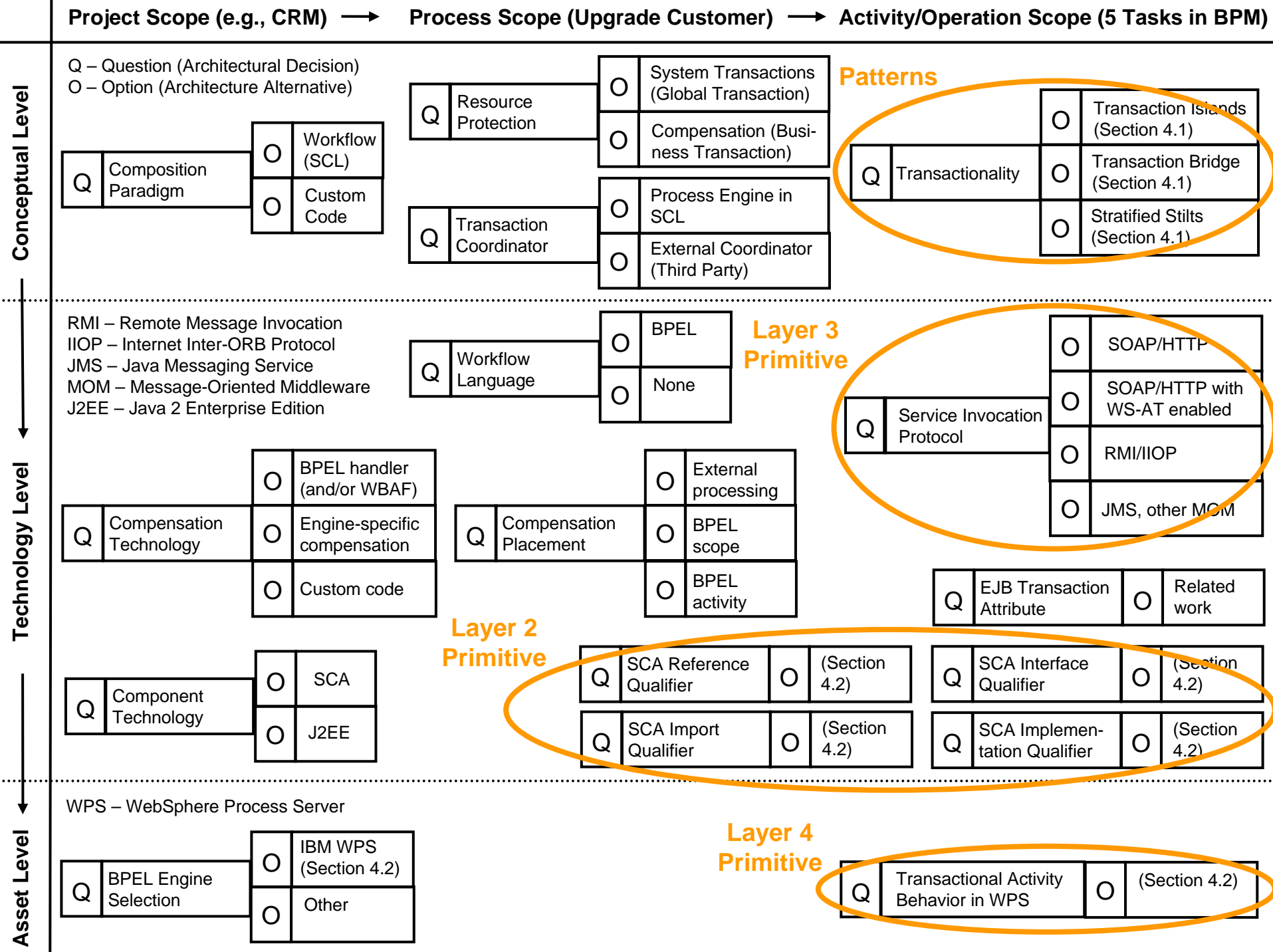


Life without Architectural Patterns and Decisions...

- No platform-independent solution descriptions or best practices
 - In industry example, WID-specific articles and an online help exist on WWW
 - Given advice is platform-specific and assumes deep platform expertise
- BPM to BPEL export tools set transactionality to inappropriate default values
 - In industry example, one WID setting per activity, plus four SCA qualifiers involved
- Role problem
 - Solution architect (responsible, but no awareness) vs. platform specialist (skills, tool)
- Reuse problem
 - Many platform-specific settings: BPEL, SCA, WSAT, etc.
- Reconciliation problem
 - Changes to WID overwritten each time BPEL is regenerated; BPEL vs. SCA consistency

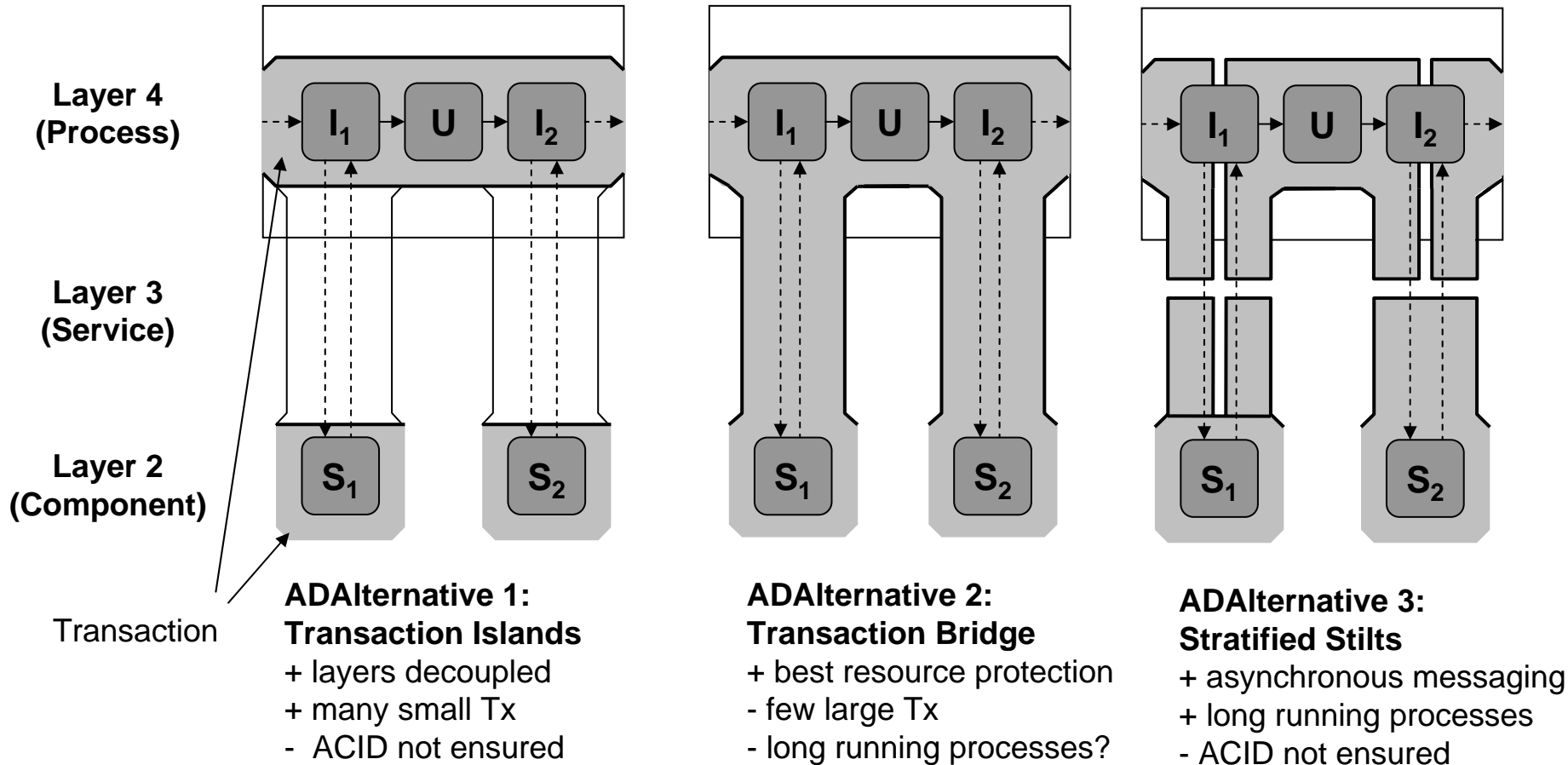
Agenda

- SOA Decision Modeling overview
 - Meta model
 - Levels and layers
 - Decision scoping and dependencies
- **Decision Model for transactional workflows in SOA**
 - **Step (a): Identification of conceptual decisions and patterns**
 - **Step (b): Decomposition by layer**
 - **Step (c): Refinement from conceptual to technology to vendor/asset level**
- Conclusions and future work



Patterns as ADAalternatives on Conceptual Level

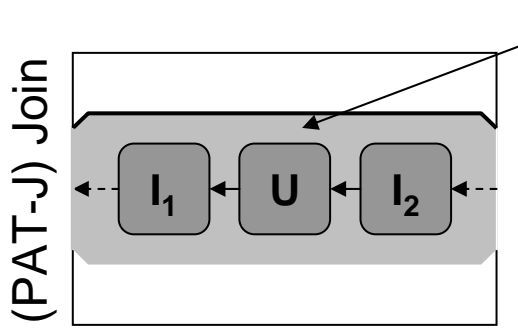
{Conceptual Decision Model}



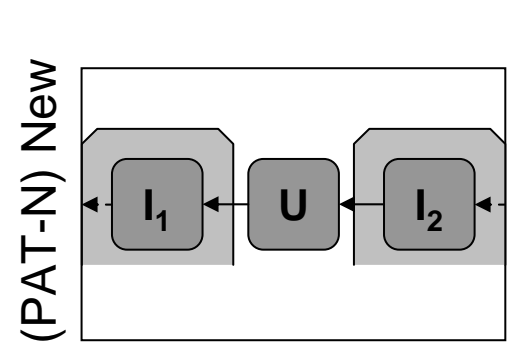
Process Activity Transactionality Primitives (Layer 4)



Layer 4
(Process)



Alternative PAT-J:
Invoke activity joins existing Tx



Alternative PAT-N:
Each invoke activity opens new Tx

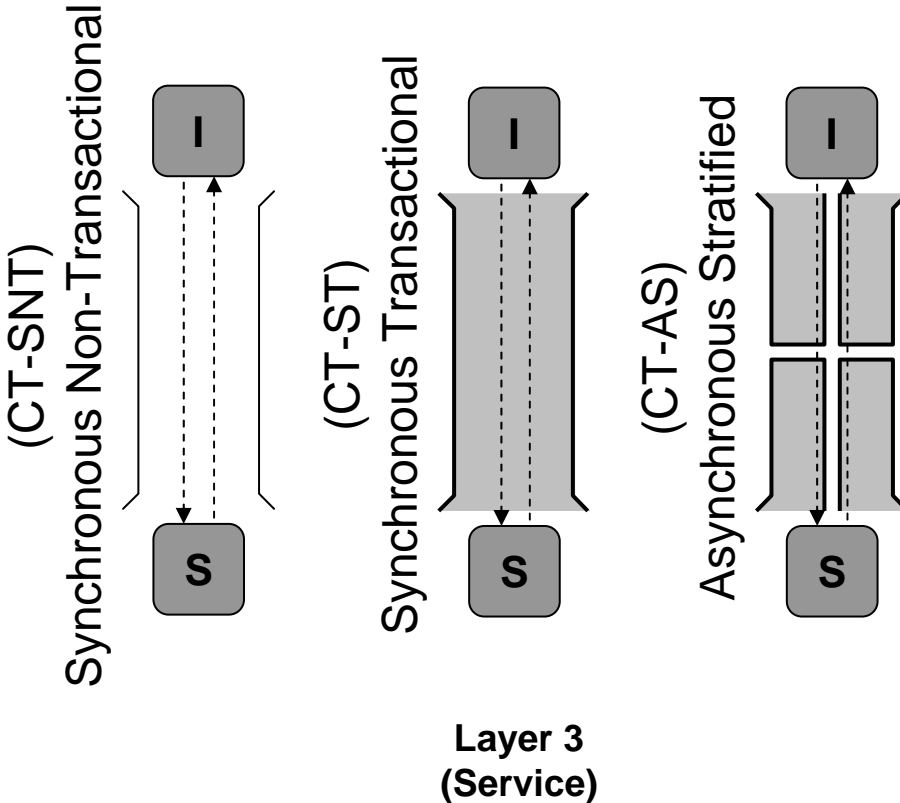


Refinement dependencies (“forces”):

No standard mapping to BPEL – many platform-specific extensions & limitations (e.g., WID can add Tx boundaries at any time, ACID scopes in BPEL4J)

In WID, PAT-J maps to “Participates”, PAT-N to “Requires Own”

Communication Infrastructure Primitives (Layer 3)



**CT-SNT: SOAP/HTTP, IIOP
(without Tx context sharing)**

**CT-ST: SOAP/HTTP with WS-AT,
IIOP (with Tx context sharing)**

CT-AS: WS-RM, JMS

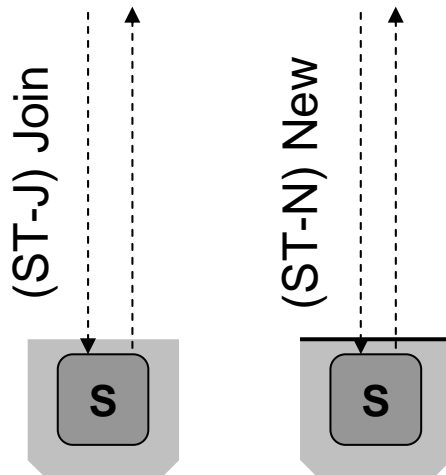
**Plus SCA Qualifiers in WID/WPS
(see slide with SP primitives)**

Service Provider Transaction Primitives (Layer 2)

{Conceptual Decision Model}

{Technology Decision Model}

{Asset Decision Model}



**Layer 2
(Component)**

SCA Qualifiers in WID/WPS

Primitive Qualifiers	CT SCA reference (BPEL process as component invoking others)	CT SCA import (reference to external service)	CT SCA interface (service provider component)	ST SCA implementation (service provider component)
TRANSACTION ISLANDS	CT-SNT <u>DeliverAsyncAt=n/a</u> <u>SuspendTx=true</u>	CT-SNT <u>JoinTx=false</u>	CT-SNT <u>JoinTx=false</u>	ST-N (or ST-J) Transaction= <u>local global any</u>
TRANSACTION BRIDGE	CT-ST <u>DeliverAsyncAt=n/a</u> <u>SuspendTx=false</u>	CT-ST <u>JoinTx=true</u>	CT-ST <u>JoinTx=true</u>	ST-J Transaction= <u>global</u>
STRATIFIED STILTS	CT-AS <u>DeliverAsyncAt=commit</u> <u>SuspendTx=false</u>	CT-AS <u>JoinTx=n/a</u>	CT-AS <u>JoinTx=n/a</u>	ST-J Transaction= <u>global</u>

Agenda

- SOA Decision Modeling overview
 - Meta model
 - Levels and layers
 - Decision scoping and dependencies
- Decision Model for transactional workflows in SOA
 - Step (a): Identification of conceptual decisions and patterns
 - Step (b): Decomposition by layer
 - Step (c): Refinement from conceptual to technology to vendor/asset level
- **Conclusions and future work**

Summary of Benefits and Novel Contributions

- Design method complementing general purpose methodologies
 - Anticipating required decisions, giving concrete advice (per role)
- Model-driven approach to decision capturing
 - Reusable, pre-populated ADMs replacing text tables (reuse)
 - Decision outcome not overwritten by code generation (reconciliation)
- Tool prototype
 - Creation of decision model from requirements modeling (BPM) tool
 - Decision injection into BPEL editor

- Time savings
- Risk mitigation
- Quality improvements

Future Work Regarding These and Other SOA Decisions

- Capture more variations and decision drivers for presented patterns
- Map primitives to additional platforms (technologies, assets)
- Capture patterns for other decision types
- Represent patterns as runtime policies

- Integrate SOAD into end-to-end tool chain
 - Get more information from BPM (roles, resources, technical attributes)
 - Design and decision model interlock
 - Collaboration platform
 - More case studies