

Aspect-Based Variability Model for Cross-Organizational Features in Service Networks

Stefan Walraven Bert Lagaisse Eddy Truyen Wouter Joosen

stefan.walraven@cs.kuleuven.be

Composition & Variability, 2010



Why Adaptability & Variability?

- ⇒ Different users of a service have different and varying requirements

Recent trend

- ▶ Feature-based + service-based ¹
- ▶ Selecting different features ⇒ different service variants

¹S. Apel, C. Kaestner, Christian, C. Lengauer. *Research Challenges in the Tension Between Features and Services*, 2008

Cross-Organizational Service Composition

- ▶ Composition of services from different organizations
- ▶ Each company has its own IT administration and trust domain
 - ▶ Services == black boxes
 - ▶ Independently developed & maintained
 - ▶ External parties not allowed to add or update features

Cross-Organizational Service Composition

- ▶ Composition of services from different organizations
- ▶ Each company has its own IT administration and trust domain
 - ▶ Services == black boxes
 - ▶ Independently developed & maintained
 - ▶ External parties not allowed to add or update features
- ▶ What does it mean when the implementation of a feature is scattered **across multiple organizations**?

Cross-Organizational Service Composition

- ▶ Composition of services from different organizations
- ▶ Each company has its own IT administration and trust domain
 - ▶ Services == black boxes
 - ▶ Independently developed & maintained
 - ▶ External parties not allowed to add or update features
- ▶ What does it mean when the implementation of a feature is scattered **across multiple organizations**?
 - ⇒ No single module (or aspect) per feature

Cross-Organizational Service Composition

- ▶ Composition of services from different organizations
- ▶ Each company has its own IT administration and trust domain
 - ▶ Services == black boxes
 - ▶ Independently developed & maintained
 - ▶ External parties not allowed to add or update features
- ▶ What does it mean when the implementation of a feature is scattered **across multiple organizations**?
 - ⇒ No single module (or aspect) per feature
 - ⇒ Division of features into client-side and server-side parts

Motivating Example

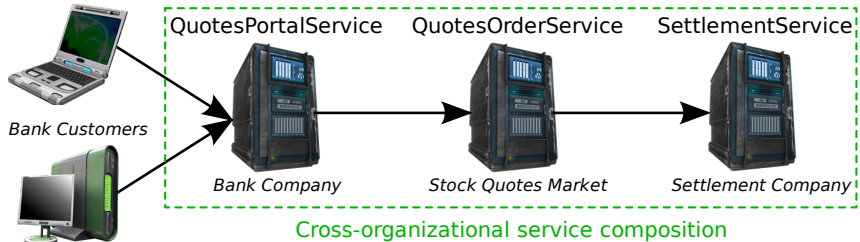
Goal

Approach

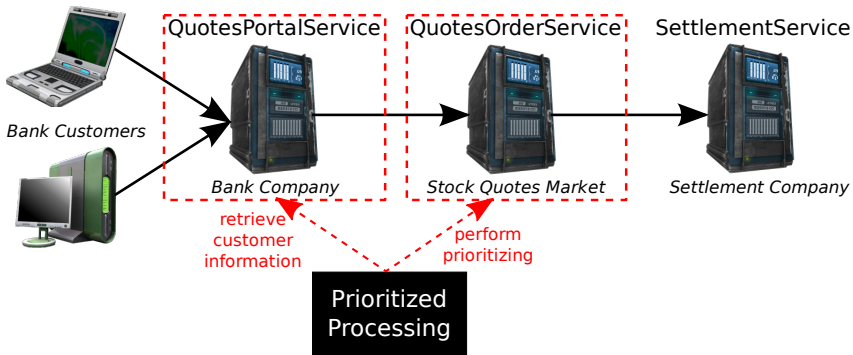
Cross-Org Service Customization

Summary

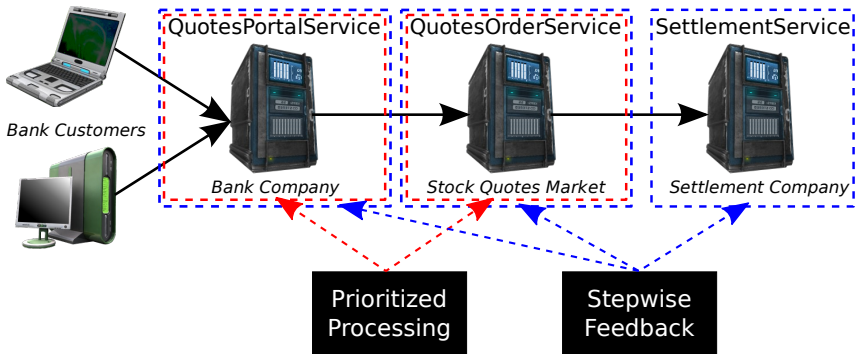
Example: Stock trading service



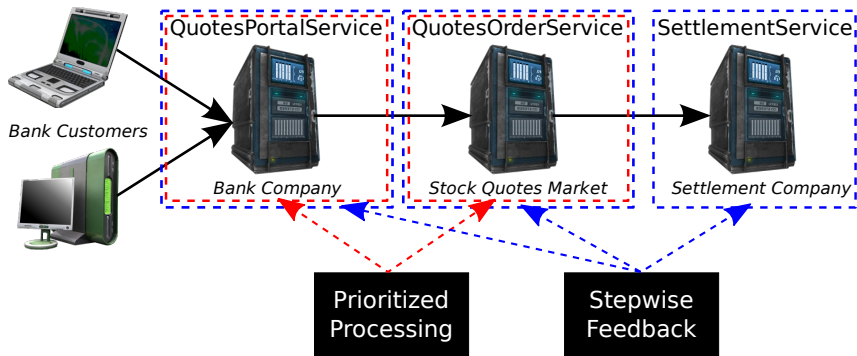
Example: Stock trading service



Example: Stock trading service

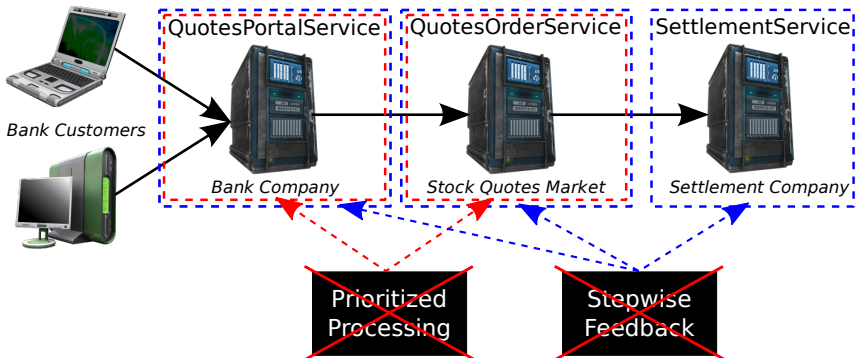


Example: Stock trading service



- ▶ Single module encapsulating feature functionality for \neq services?

Example: Stock trading service



- ▶ Single module encapsulating feature functionality for \neq services?
- ▶ Not desired in a cross-organizational context!

- ▶ How to ensure interoperability?
- ▶ How to share semantically compatible features?

- ▶ How to ensure interoperability?
- ▶ How to share semantically compatible features?

Cross-organizational service customization

- ▶ High-level contract language
- ▶ Coordinated deployment of cross-org features

Overview of Approach

Motivating Example

Goal

Approach

- Multi-Layered Architecture

- High-Level Feature Ontology

- Mapping to Aspect-Based Implementation

Cross-Org Service Customization

Summary

Multi-Layered Architecture

- ▶ Based on cross-organizational coordination architectures ²
- ▶ Language for describing agreements between services:
 1. **Conceptual model**
 - ▶ modeling concepts describing high-level agreements
 - ▶ independent from computational model
 2. **Computational model**
 - ▶ behavioral concepts
 - ▶ mappable to implementable actions in underlying software system

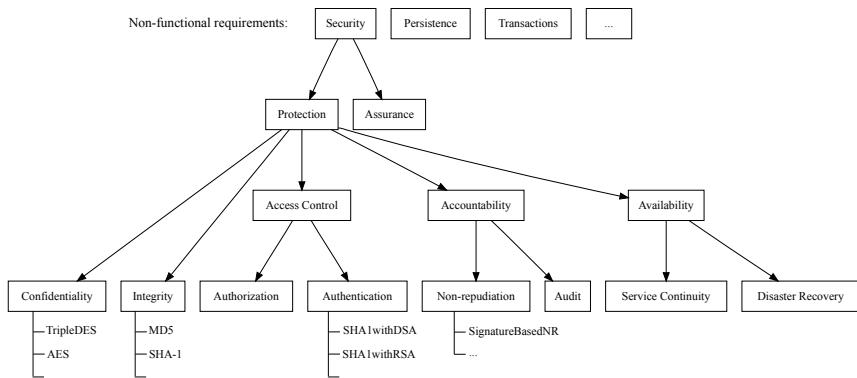
²E. Truyen, W. Joosen. *A Reference Model for Cross-Organizational Coordination Architectures*, 2008

Multi-Layered Architecture

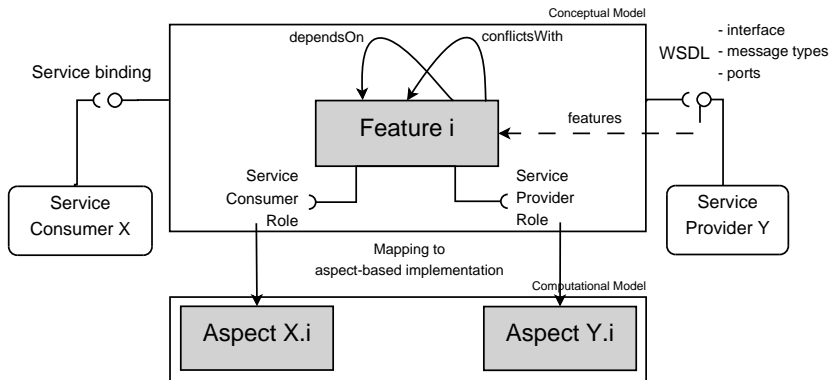
- ▶ Based on cross-organizational coordination architectures ²
- ▶ Language for describing agreements between services:
 1. **Conceptual model** → **Feature ontology**
 - ▶ modeling concepts describing high-level agreements
 - ▶ independent from computational model
 2. **Computational model** → **AO-based implementation mapping**
 - ▶ behavioral concepts
 - ▶ mappable to implementable actions in underlying software system

²E. Truyen, W. Joosen. *A Reference Model for Cross-Organizational Coordination Architectures*, 2008

High-Level Feature Ontology



Overview Architecture



Feature Contract

- ▶ Agreement between service consumer and service provider
- ▶ High-level, technology-independent
- ▶ Clear scope (particular application domain, service network. . .)
- ▶ Intended behavior described in feature contracts:
 - ▶ feature identifier
 - ▶ roles (e.g. `ServiceConsumer`)
 - ▶ responsibilities of a role (constraints, interfaces. . .)
 - ▶ composition rules (dependencies)

Feature Contract

```
feature PrioritizedProcessing {
  dependsOn: Billing;
  role ServiceConsumer {
    responsibility retrieveCustomerAccount {
      provides: CustomerAccount;
    }
  }
  role ServiceProvider {
    responsibility performPrioritizing {
      requires: CustomerAccount;
      provides: AccountableItem;
    }
  }
}
```

Aspect-Based Implementation Mapping

- ▶ Mapping feature ontology → AO-implementations
- ▶ Independently defined by each organization
 - ▶ On the level of internal processes and data
 - ▶ Hiding implementation details for external parties
 - ▶ Feature implementation using AOP technology of their choice⇒ heterogeneity
- ▶ Use of AOSD:
 - ▶ Clean separation of concerns
 - ▶ Feature == set of aspect-components
 - ▶ Dynamically composed

Mapping to Aspect-Based Implementation

```
featureImplMapping PrioritizedProcessingImpl {
  implements: PrioritizedProcessing;
  role: ServiceProvider;
  ao-composition {
    id: PrioritizedOrderProcessing;
    pointcut {
      kind: execution;
      componenttype: *;
      componentinstance: *;
      interface: IOrderProcessing;
      method: process;
    }
    advice {
      comptype: PrioritizedOrderProcessing;
      interface: IPrioritizedOrderProcessing;
      method: prioritize;
    }
  }
}
```

Mapping to Aspect-Based Implementation

```
featureImplMapping PrioritizedProcessingImpl {  
  implements: PrioritizedProcessing;  
  role: ServiceProvider;  
  ao-component: PPAOComponent;  
}
```

Expressing user-specific preferences

- ▶ Select desired set of features from feature ontology
- ▶ Declarative specification for configuration across service network

Cross-Org Service Customization

Expressing user-specific preferences

- ▶ Select desired set of features from feature ontology
- ▶ Declarative specification for configuration across service network

Example: Web Services

- ▶ WS interface specifies available features
- ▶ Configuration by defining *service bindings*

```
servicebinding {  
  URI: http://www.stocktradingexample.be;  
  port: StockTradingServiceSoapEndpoint;  
  features: PrioritizedProcessing , Billing;  
}
```

GlueQoS³

- ▶ Middleware-based resolution mechanism
- ▶ Searching satisfiable set of QoS features during service composition
 - ▶ Decentralized selection
 - ▶ Per-collaboration basis
- ▶ Fixed ontology for classifying features
- ▶ No support for:
 - ▶ User-specific customization
 - ▶ Consistent processing throughout cross-org service composition

³E. Wohlstadter et al. *GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions*, 2004

Conclusion

- ▶ Need for high-level contract language to describe cross-organizational features
- ▶ Coordinated cross-org deployment
- ▶ Proposed approach:
 1. Technology-independent feature ontology
 2. Aspect-based feature implementation mapping

- ▶ Outlook:
 - ▶ Further development & improvement of our approach
 - ▶ Thorough validation and evaluation of coordination architecture

Questions?

