A Dynamic Software Product Line Approach using Aspect Models at Runtime

Tom Dinkelaker, Ralf Mitschke, Karin Fetzer*, Mira Mezini

Technische Universität Darmstadt, Germany
* SAP Research Dresden, Germany
Talk in a nutshell

1. Validation of dynamic configurations

2. Configuration using dynamic execution context

3. Realization of 1. & 2. with dynamic AOP
Validation of dynamic configuration

SPL Definition (Feature Model)

Static Configuration (Deployment)

validate

Dynamic Configuration (Runtime)

validate
Single model for dynamic and static configuration

Static Configuration (Deployment)

Dynamic Configuration (Runtime)

validate
Extending Feature Models to validate dynamic configurations

Validated Constraints:

- **implies:** $A \rightarrow B$  (require activation of targeted feature)
- **excludes:** $\neg (A \land B)$  (require deactivation of targeted feature)
- **precedes:** $\text{priority}(A) > \text{priority}(B)$  
  (executes A before B resolves interaction)
Dynamic Variations

- Approving orders manually or automatically

Variation Point

**Variant A**
- receive
- manual approve
- pack
- ship

**Variant B**
- receive
- auto approve
- pack
- ship
Combining Variations

→ Requires resolution strategy

- Expressible in our notation:

![Diagram showing ManualApproval precedes AutomaticApproval with a receive state and subsequent states labeled with question marks.]}
Combining Variations

→ Requires resolution strategy
  - Expressible in our notation:

![Diagram showing Manual Approval and Automatic Approval with an excludes relationship]
2. Configuration using dynamic execution context
Context dependent Variation

- Resolve interaction by quantity of orders

:Order
quantity = 748

:Order
quantity = 31
Validate dynamic configurations

Dynamic Aspect Language & Runtime

Model

AutomaticApproval

ManualApproval

Realize

... AutomaticApproval

{ // quantity < 100
   ...
}

\[ \cap \]

Runime Constraint Model

... ManualApproval

{ // quantity >= 100
   ...
}

= \emptyset
3. Realization with dynamic AOP
for Validation (1.) & Dynamic Context (2.)
Encode Dynamic Features as Dynamic Aspects

- Dynamic aspects provide de-/activation
- Feature constraints are defined on aspects
- Aspects expose runtime context
- General advantages of aspects
  - flexible modularization
  - enable crosscutting features (e.g. security)
Constraints with dynamic AOP

aspect (name: "AutomaticApproval", deployed: true) { ... }

aspect (name: "ManualApproval", deployed: true) {

declare_exclusion "ManualApproval","AutomaticApproval";
}

dynamically deployable

ManualApproval excludes AutomaticApproval
Realization with dynamic AOP

```java
aspect (name:"ManualApproval") {

after( receive_order() ) {
    boolean trusted = UI.approvalDialog(order).isCustomerTrustable();
    if (! trusted ) // disapprove order
}
}

Context at variation point.
resolved via:
order = thisJoinPoint.context.get("order")
```
aspect(name:"ManualApproval") { 
  // quantity >= 100
  after( receive_order(100, *) ) { ... }
}

aspect(name:"AutomaticApproval") { 
  // quantity < 100
  after( receive_order(0, 100) ) { ... }
}
Case Study
Sales scenario approach

- Business-driven end-user features
- User-driven feature adaptation (feature management console)
Contributions

- Dynamic feature models with novel dynamic constraints
- Realization with dynamic aspect runtime model
  - constraint enforcement
  - context dependent interactions
- Case study as basis for evaluation
Discussion

- For details on aspect runtime visit research talk:

  “An Architecture for Composing Embedded DSLs”

- Other ways of managing dynamic variation?