



Agenda



- Introduction
- Flexible Variant Configuration (FVC)
- Feature Unweaving
- Conclusions

Introduction



Modeling requirements for single systems

- Numerous techniques
- Various tools

Introduction II



Modeling requirements for product lines

- Similar to modeling for a single system
 - ▣ Commonality
- Different from modeling for a single system
 - ▣ Variability

Introduction III

- Central Question:

*How can we model the **variability** of a software product line?*

- ▣ Ad hoc solutions (e.g. AND/OR tables)
- ▣ Flexible Variant Configuration
- ▣ Feature Unweaving



FLEXIBLE VARIANT CONFIGURATION

FVC - Overview

- Extend conventional modeling languages, e.g. UML, to support variability
- Core Elements
 - ▣ Domain defaults
Characterize a typical system in the domain
 - ▣ Customization scripts
Create variants with respect to domain defaults
- Flexible Variant Configuration Tool
Interprets customization scripts to produce customized domain models

FVC - Procedure

- **Start with a conventional model**

The user selects a meeting room.

- **Create domain defaults**

.REPLACE FACILITY by "meeting room"

The user selects a "FACILITY".

- **Create the customization scripts**

.REPLACE FACILITY BY "hotel room".

FVC - Further Remarks

- Graphical Models
 - ▣ Extend them to support modeling variability
 - ▣ Convert them to XML
 - ▣ Change the XML representation
 - ▣ Convert it back to the graphical model
- Drawbacks
 - ▣ No systematic handling of dependencies
 - ▣ Difficulties that arise from the use of UML

FEATURE UNWEAVING



Feature Unweaving

What it is

- Supports the extraction of variability from a given reference model
- Semi-automatic
 - ▣ Manual: The product manager identifies the elements of the model that vary
 - ▣ Automatic: extraction and refactoring of these elements into features

Core elements

Integrated requirements modeling

- ADORA
- To select elements in a single view

Aspect-oriented modeling

- To model variability
- Variable features are modeled with aspects
- Aspect weaving techniques can be used

Algorithm:

recursion:

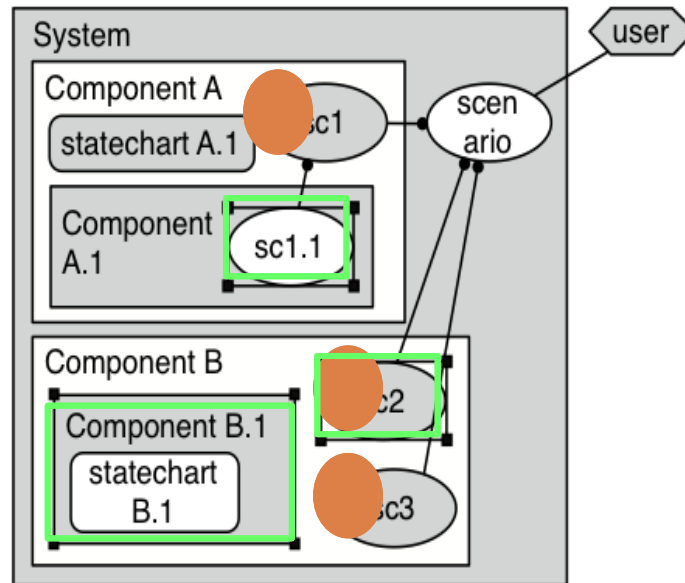
1. Create a base aspect container
 2. Call the recursive extraction function 
 3. Remove all redundant aspect containers in the base aspect container
 4. Fine-tune the graphic layout
 5. Set the base aspect container as a variability. Adds a unique decision item and annotate the relationships
- 2.1 Find the base parent container
 - 2.2 All selected elements within this container are extracted into the base aspect container
 - 2.3 The necessary join relationships are computed and created
 - 2.4 If further nested elements exist, a new aspect container is created inside the base aspect container and the next recursive call is performed



Feature Unweaving Illustration

Reference model

- Selection of elements that constitute a variable feature
- Sequence number of the scenario
- Containment hierarchy

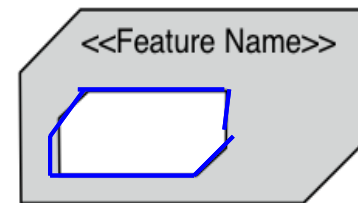
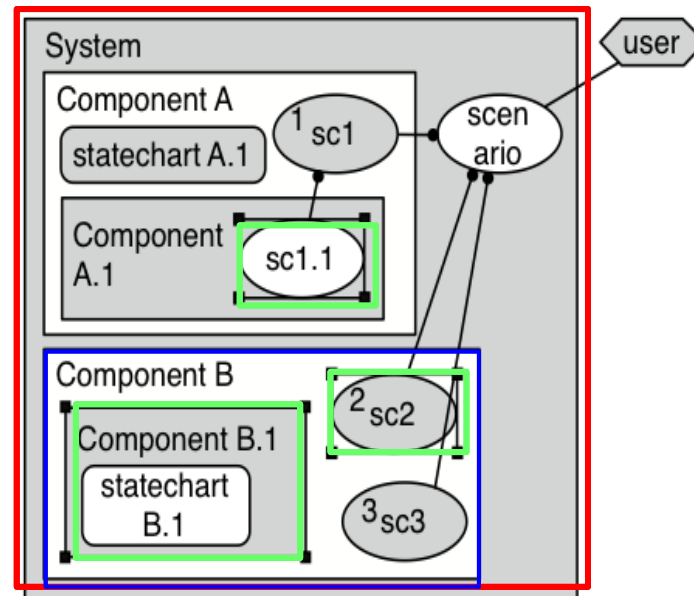


Location where the extracted feature shall be positioned

■ 2.1 PC : System object

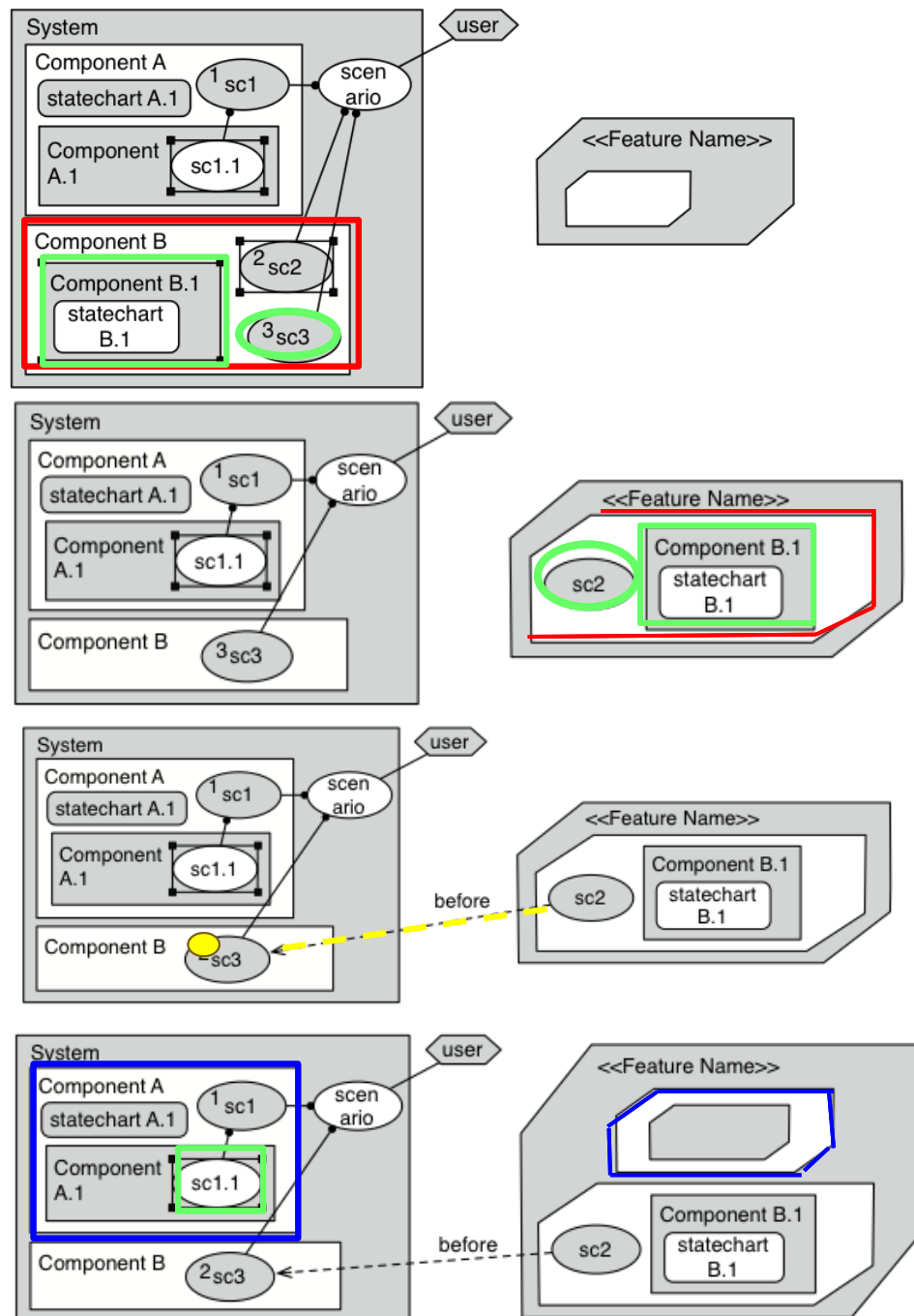
2.2 No extraction (no directly nested selected elements)

2.3 noting to calculate



■ 2.4 Creation of a new nested AC

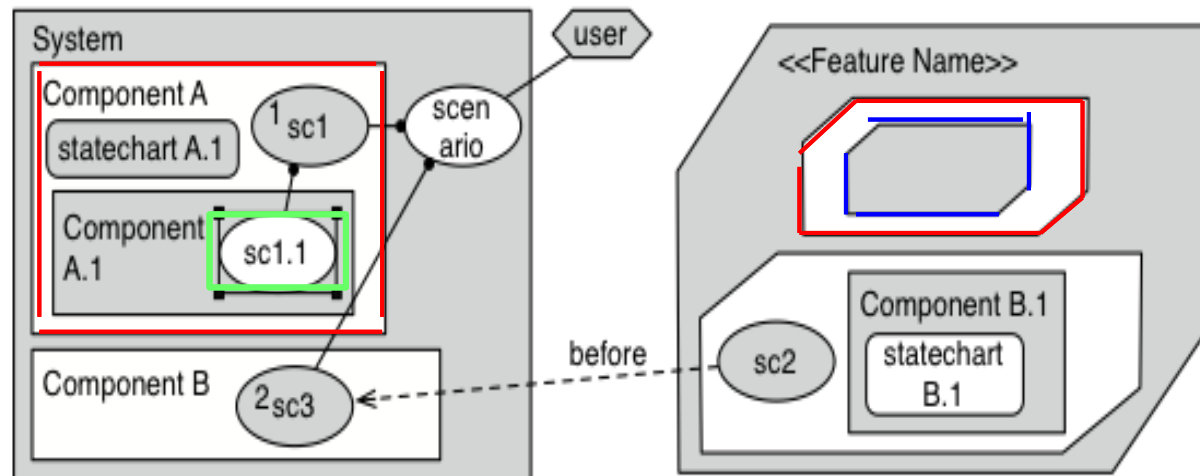
- 2.1 PC: Component B
- 2.2 Extraction of sc2 and component B.1 (Remove the connection to its parent scenario and relocate into the new AC)
- 2.3 Creation of a “before” relationship (keeps the core model consistent)
- 2.4 Creation of a new nested AC



■ 2.1 PC: Component A

2.2 No extraction
(no directly nested
selected elements)

2.3 noting to
calculate



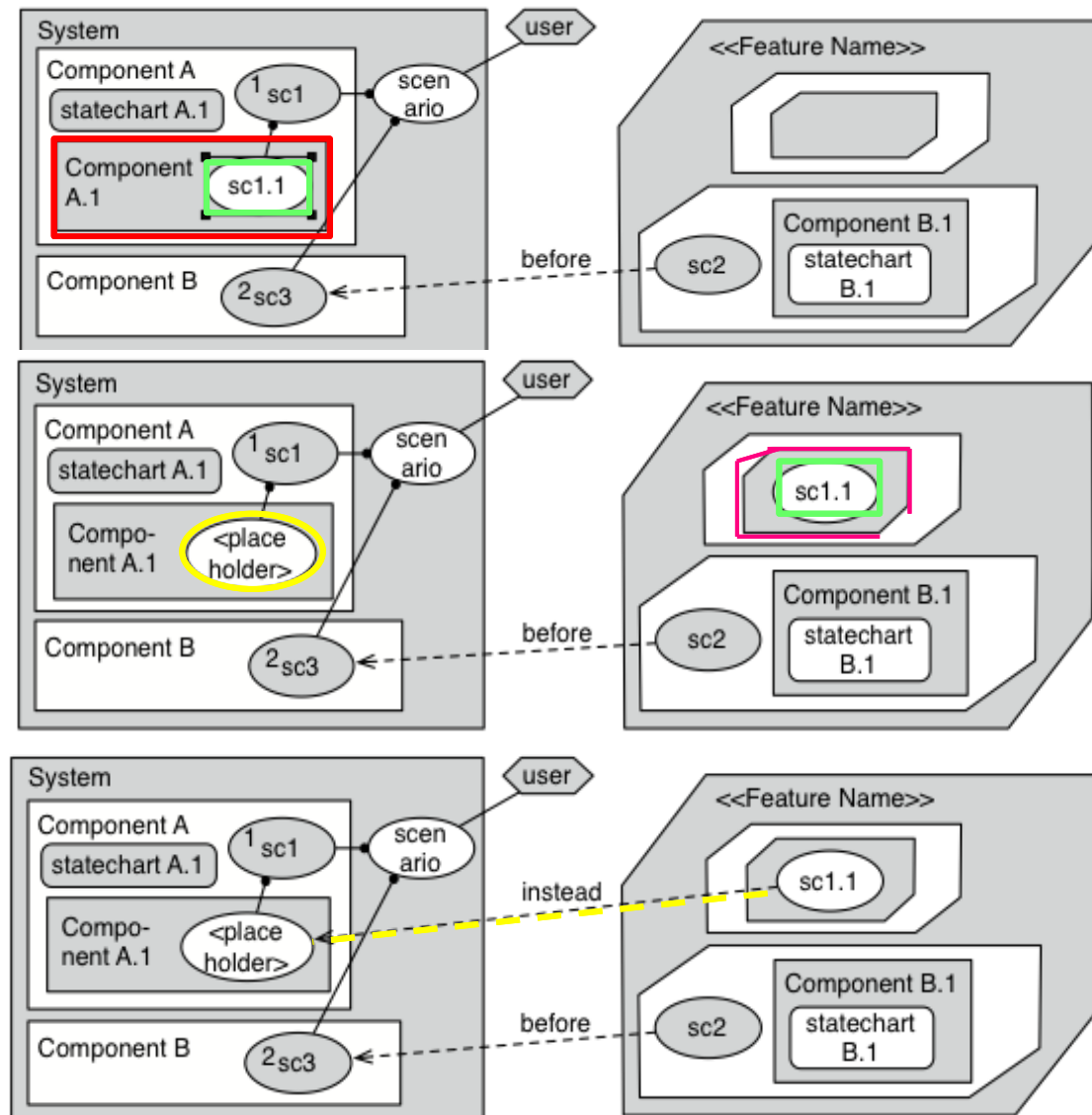
■ 2.4 Creation of a new nested AC

2.1 PC: Component A.1

2.2 Extraction of sc1.1 (Remove the connection to its parent scenario and relocate into the new AC)

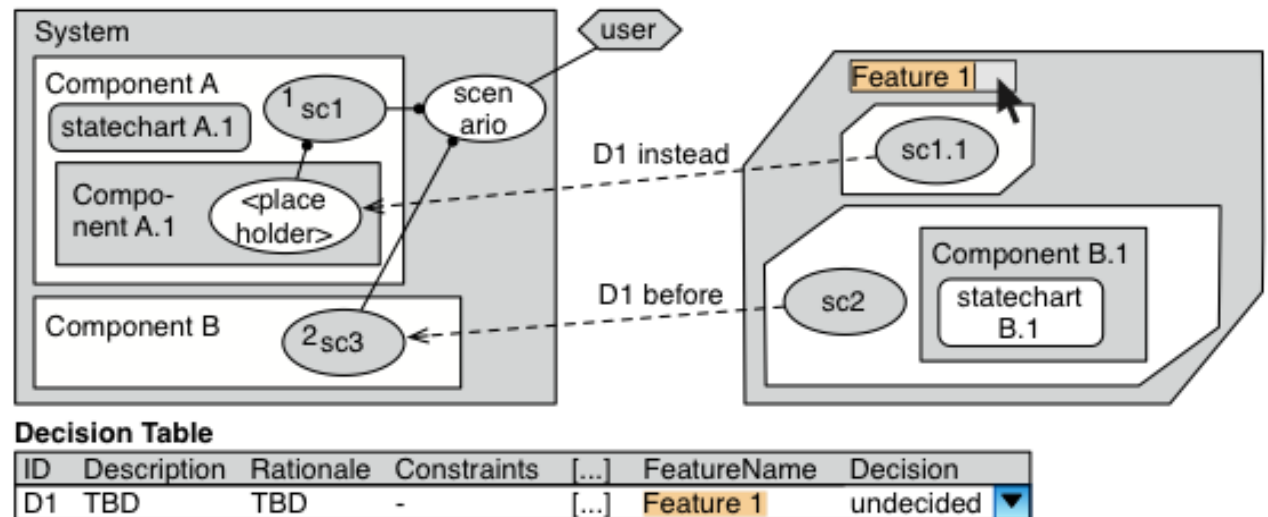
2.3 Creation of a placeholder (no siblings: no join point) and an “instead” relationships

2.4 Termination (no further PC)



Refactored model

Addition of a
unique decision
item in the
decision table





CASE STUDIES

Case Study 1

- Evaluation of the efficiency of the feature unweaving compared to AND/OR table approach
- Improvements in efficiency of
 - ▣ product line specification,
 - ▣ Product variant definition,
 - ▣ Release planning, and
 - ▣ Consistency maintenance.

Case Study 2

- Evaluation of feature unweaving compared to an UML feature model approach
 - ▣ Using ADORA for specifying requirements and variability is more efficient
 - ▣ Reduces the complexity and size of the specification

Conclusions



- ❑ Modeling variability is essential for software product line requirements modeling
- ❑ Existing modeling languages can be extended, but are not especially well-suited for the problem
- ❑ Feature unweaving reduces the effort and cost for creating variable features

Questions

