Automatic Generation of Feature Models from UML Requirement Models

Hernán Casaláanguida
Juan Eduardo Durán

Universidad Nacional de Córdoba - Argentina
Introduction

• **UML requirements notations:**
  – Use case diagrams and activity diagrams.
  – Extensions of them to model variabilities in the literature.

• Feature models

• **Necessity of both** UML requirement notations and feature models.

• *The development of both kind of models in parallel could introduce inconsistencies between them.*
Introduction

• **Problems of first building an feature model and from it construct UML models for requirements.**
  
  – it cannot be expected that UML specialists will construct feature models of good quality.
  
  – the construction of a use case diagram from a feature model is usually a difficult task
  
  – the construction of activity diagrams from a feature model is a difficult task and cannot be automated
  
  – we need both domain specialists and UML specialists, and misunderstandings between them could occur
Introduction

• **Decision**: to develop UML models for requirements and from them to automatically generate a feature model.
  – UML specialists need to learn how to model variabilities inside UML models for requirements.

• **Lack of work** about mapping UML notations for describing use cases into feature models.
  – We map activity diagrams into feature models.

• **Main objective**: to use MDA to define a transformation from UML requirement notations to a feature model
  – We used Atlas Transformation Language for this purpose.
Introduction

• Our previous work for requirements engineering for rich internet applications families
  – Use case diagrams and activity diagrams with variabilities.
  – A profile extending UML use case diagrams and activity diagrams for rich internet applications.

• Secondary objective: to improve this work
  – We extended use case diagrams with variabilities with two new kinds of variabilities.
  – We added three new stereotypes to the previous profile.
Use Case Diagrams with Variabilities

Use Case Diagram for an Online Library
Use Case Diagrams with Variabilities

Variability modeling elements for Use Case Diagrams
Use Case Diagrams: profile for rich internet applications

- «task»: the use case execution performs a task supporting the execution of one or more jobs.
  - a job consists of one or more operations used for input validation, information modification, calculation.

- «navigation»: the use case execution consists of a navigation performed by a user.
  - A navigation consists of one or more presentation content steps that can be either user inputs or search results.

- «RIA task»: like «task» + specific RIA user interface features or asynchronous jobs.

- «RIA navigation»: like «navigation» + specific RIA user interface features or asynchronous searches.
Activity Diagrams with Variabilities

Return Item
use case
description
Activity Diagrams with Variabilities

Lend Item use case description
Activity Diagrams with Variabilities

- An *induced activity group (IAG)* is a special class of UML activity groups.
- IAGs are used to model variants and also to define reusable diagrams representing behaviors.
Activity Diagrams: profile for rich internet applications

- **«search»**: they represent database queries
- **«job»**: they are call behavior actions, whose activity performs a job,
- **«input»**: they represent the provision of an input by an human actor,
- **«output request»**: they represent the request by the system for the provision of some inputs,
- **«output content»**: they represent the system displaying content.
Activity Diagrams: profile for rich internet applications

• «input and suggest»: represents the provision of an input by a user, and the use of selectable automatic suggestions for this input.

• «output message»: represents the system showing a message to the user.

• «execute IAG»: extends ExecutableNode and represents the execution of a behavior given by an IAG.
  – Tagged value source: its value is the name of the UC where the IAG was defined.
Feature Models

- Library
  - find items
  - reserve item
  - relend item

- Loan Management
  - lend item
  - return item
  - register loan with payment
  - register loan with credits
  - print account data
  - create user
  - delete user
  - update user
  - print access card
Transformation from UML requirement models to a feature model

UCD \rightarrow \text{UCD2FM} \rightarrow \text{initial FM} \rightarrow \text{final FM}

Development Process
# Transformation UCD2FM

<table>
<thead>
<tr>
<th>Rule</th>
<th>maps</th>
<th>onto</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D name of UCD</td>
<td>D: Feature</td>
</tr>
<tr>
<td>2</td>
<td>Association between A:Actor and U: UC that is not a variant.</td>
<td>SingleRelation: 1..1 between D, U:Feature</td>
</tr>
<tr>
<td></td>
<td>Association between A:Actor and P:Package that is not a variant.</td>
<td>SingleRelation: 1..1 between D, P: Feature</td>
</tr>
<tr>
<td>3</td>
<td>V: AssocVariability: n..m Variants: associations from A:Actor to U₁,...,Uₖ: UC, such that U₁,...,Uₖ are not inside a package.</td>
<td>Relation: n..m between D:Feature and U₁,...,Uₖ: Feature.</td>
</tr>
</tbody>
</table>
Transformation UCD2FM

<table>
<thead>
<tr>
<th>Rule</th>
<th>maps</th>
<th>onto</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Association between A:Actor and U: UC. U is inside a non variant P:Package.</td>
<td>SingleRelation: 1..1 between P:Feature and U:Feature</td>
</tr>
</tbody>
</table>
## Transformation UCD2FM

<table>
<thead>
<tr>
<th>Rule</th>
<th>maps</th>
<th>onto</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Extend from U’: UC to U: UC that is not a variant.</td>
<td>SingleRelation: 1..1 between U:Feature and if ∄ U’:Feature then U’:Feature else Reference to U’:Feature</td>
</tr>
<tr>
<td></td>
<td>Include from U: UC to U’: UC</td>
<td>SingleRelation: 1..1 between U:Feature and if ∄ U’:Feature then U’:Feature else Reference to U’:Feature</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>V: ExtendVariability: n..m Variants are Extend from U₁: UC to U: UC, ..., Extend from Uₖ: UC to U: UC</td>
<td>Relation: n..m between U:Feature and if ∄ U₁:Feature then U₁:Feature else Reference to U₁:Feature, ..., if ∄ Uₖ:Feature then Uₖ:Feature else Reference to Uₖ:Feature</td>
</tr>
</tbody>
</table>
Transformation UCD2FM

Feature model for online libraries
**Transformation Reqs2FM**

**Assumption:** the cited modeling elements are inside an AD for a use case that is mapped to a feature U in the initial feature model.

<table>
<thead>
<tr>
<th>Rule</th>
<th>maps</th>
<th>onto</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>E:EN that is not a variant, and of stereotype: «job», «search» or «execute IAG»</td>
<td>SingleRelation: 1..1 between U:Feature and if ( \not\exists E:Feature ) then E:Feature else Reference to E:Feature</td>
</tr>
<tr>
<td>2</td>
<td>V: CFV: 0..1. Variant: E: AE.</td>
<td>SingleRelation: 1..1 between U, E :Feature</td>
</tr>
<tr>
<td></td>
<td>V: CFV: 0..1. Variant: G: IAG.</td>
<td>SingleRelation: 1..1 between U, G :Feature</td>
</tr>
<tr>
<td>3</td>
<td>V: CFV: n..m with m&gt;1. Variants: ( E_1,\ldots,E_k: AE ) and ( G_1,\ldots,G_j: IAG ).</td>
<td>GroupRelation: n..m between U:Feature and ( E_1,\ldots,E_k,G_1,\ldots,G_j: Features )</td>
</tr>
</tbody>
</table>
## Transformation Reqs2FM

<table>
<thead>
<tr>
<th>Rule</th>
<th>maps</th>
<th>onto</th>
</tr>
</thead>
</table>
| 4    | V: DFV : 0..1  
SingleRelation: 0..1 between V, O: Feature. |
| 5    | V: DFV: n..m, where m>1.  
Variants:  
O₁,...,Oₖ: ObjectNode | SingleRelation:  
if n>0 then 1..1 else 0..1  
Between U, V: Feature  
GroupRelation: n..m  
between V: Feature and  
O₁,...,Oₖ: Feature. |
Transformation
Reqs2FM

A feature model for online libraries
Related Work

• **Generation of feature models from use case diagrams**
  
    
    • Presents a transformation from a use case diagram into a feature model
    
    • We extend this work with the mapping of packageVariability variants and of associationVariability variants onto features.

  — Wang etal., 2009.
    
    • Construction of a feature model from individual products that are described with use case diagrams and use case scenarios.
    
    • Not applicable when we have too few products or when we do not have previous applications.
Related Work

- **Use case diagrams with variabilities**
    - We only adopted extendVariability and added other kinds of variabilities.
    - variation points inside use case diagrams, whose variants are use cases are used;
    - however, variation point and variability relationship elements do not belong to UML metamodel
    - Our associationVariability element belongs to UML and is equivalent to variability relationship construct.
Related Work

- **ADs with variabilities approaches that use decision nodes to model variabilities**
  - This approach is more complex than our proposal for the case of variabilities of cardinality 0..1.
  - For the case of cardinality n..m with m > 1 we use fork and join nodes in our approach instead of decision and merge nodes.
Conclusions

• **Scope of Reqs2FM rule**
  – it is circumscribed to RIA product lines;
  • but it can be used for information systems product lines;
  – to use this rule to another domain modify only rule 1.

• **Impact of Reqs2FM rule**
  – For our case study with only two ADs, this rule introduced 44% additional features.
  – All variants in these ADs are mapped to critical features (they represent 50% of the features introduced by rule Reqs2FM).
  – Features *look for reservation*, *DN*, *NI*, are not obtainable from a use case diagram.
Conclusions

- **We can start with use case diagrams for single products:**
  - first, consider the semiautomatic generation of a domain use case diagram;
    - this is a work for the future
  - next, apply rule UCD2FM to generate a feature model.

- **Our transformation considers functional capability features identification in detail**
  - For Non-functional capability features identification we propose to use the NFRV notation.

- **Generation of features on a specific domain**
  - **Hypothesis:** if the UML models contemplate and use adequate domain concepts, then the automatic generated feature model will have appropriate domain technology features.
    - To check this is a work for the future.