The Role of Feature Modeling in Software Product Line Engineering

Hassan Gomaa

Dept. of Computer Science
George Mason University
Fairfax, Virginia, USA
hgomaa@gmu.edu

16th CREST Open Workshop
Provenance and Product Lines
November 2011

Copyright © 2011 Hassan Gomaa
Software Product Line Engineering

- Software Product Line (SPL)
  - Family of products / systems (Parnas, Weiss, SEI)
- Software Product Line Engineering
  - Software engineering for a family of products
- Software Variability
  - Key problem in software product line engineering
  - Need to differentiate among common and variable software requirements and components
- Feature Modeling
  - Widely used in SPLs for specifying commonality/variability in requirements
Software Modeling and Design for SPL

• Software Modeling and Design for Single Systems
  – COMET method
  – From Use Case Models to Software Architecture

• Software Modeling and Design for SPL (PLUS method)
  – Model commonality and variability among SPL members
  – Integrate Feature Modeling with UML
    • Unifying View of Multiple-View Modeling Approach
      – H. Gomaa, “Designing Software Product Lines with UML”, Addison Wesley, 2005

Copyright 2011 H. Gomaa
Evolutionary Process Model for Software Product Lines

SPL models and architecture, Reusable components

Informal SPL Requirements

Domain (Product Line) Engineering

Product Line Reuse Library

Application Engineering

Executable Application

Unsatisfied Requirements, Errors

Copyright 2011 H. Gomaa
Model-driven Development in
Product Line Engineering

Requirements Modeling

Analysis Modeling

Design Modeling

Incremental Component Implementation

Software Testing

Copyright 2011 H. Gomaa
Software Product Line Requirements Modeling

What should SPL Modeling Method provide?

• Support variability in use case modeling
  • Kernel, optional, alternative use cases
• Integrate feature modeling with other UML views

Copyright 2011 H. Gomaa
Feature Modeling for SPL

• Feature (Kang, SEI)
  – Function or characteristic that differentiates between members of the software product line

• Feature modeling in PLUS
  – PLUS integrates feature modeling with other UML views
  – Precisely represent dependencies in variability
  – Determine features from use cases and variation points
  – Concentrate on modeling variability
Feature Modeling with UML

- Use meta-modeling notation
  - Meta-classes depict features, feature groups,
- Associations depict feature dependencies

```
<table>
<thead>
<tr>
<th>«default feature»</th>
<th>«alternative feature»</th>
<th>«alternative feature»</th>
<th>«alternative feature»</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Spanish</td>
<td>French</td>
<td>German</td>
</tr>
</tbody>
</table>
```

```
«optional feature»
Light

requires

«common feature»
Microwave Oven Kernel
```
Microwave Oven Feature Model

- MinutePlus
- Light
- Turntable
- Light
- Beeper
- TOD Clock

- Defines features and their relationships
- Some features are optional, some are required
- Features can be mutually exclusive or alternative
- Displays include One-line and Multi-line
- Heating includes One-level and Multi-level

Copyright 2011 H. Gomaa
What should SPL Design Method provide?

- Support variability and evolution
  - Static Modeling
  - Dynamic Interaction Modeling
  - Dynamic State Machine Modeling

Copyright 2011 H. Gomaa
Evolutionary Development for SPL

• Kernel First Approach
  – Develop kernel of SPL
  – Kernel is similar to single system

• Evolutionary Development Approach
  – Consider evolution as an iteration in software development

• Model-based evolution
  – Feature-based Impact Analysis
  – Consider impact of optional and alternative features on kernel
    • Emphasis on dynamic modeling
    • Interaction diagrams, state machines
Kernel statechart for Microwave Oven Control

- Incoming message to object -> input event on statechart
- Output event on statechart -> outgoing message from object

Copyright 2011 H. Gomaa
Evolutionary Development for SPL

• Feature Based Impact Analysis
  – Impact of optional/alternative features on interaction diagrams and statecharts
• Feature dependent state transition
  – Use feature condition as guard on state transition
    • Event [Feature Condition]
    – Feature condition is True if feature Selected
• Example
  – Optional feature: Minute Plus
  – Consider impact on objects
  – Consider impact on statechart
Impact Analysis of Minute Plus feature - Impact on Communication diagram for Cook Food use case

«external input device»: Keypad

«product line system»: MicrowaveOvenSystem

4M[minuteplus]: Minute Plus Input

«input»: KeypadInterface

4M.1 [minuteplus]: Minute Pressed

«timer»: OvenTimer

4M.2: Start Minute

«state dependent control»: MicrowaveOvenControl

4M.2a: Start Cooking

«entity»: OvenData

4M.3: Update Cooking Time Display

«output»: One-levelHeatingElement Interface

4M.2a.1: Start Cooking Output

«output»: One-lineDisplay Interface

4M.4: Display Time

«external output device»: One-levelHeatingElement

4M.4: Display Time

«external output device»: One-lineDisplay

Copyright 2011 H. Gomaa
Feature dependent transition: **Minute Pressed [minuteplus]**

- **Cooking**
  - entry/ 4M.2a, 6.2: Start Cooking,
  - exit/ 8.4: Stop Cooking

- **4M.1: Minute Pressed [minuteplus]/ 4M.2: Start Minute**

- **Door Shut With Item**
  - 8.3: Timer Expired/ 8.4b: Switch Off [light]

- **Ready To Cook**
  - 6.1: Start/ 6.2a: Start Timer

Copyright 2011 H. Gomaa
Feature Based Impact Analysis

• Feature dependent action
• Action is only executed if Feature Condition is True
  – Action [Feature Condition]
    • Switch On [light], Switch Off [light]
  – Feature condition is
    • True if feature Selected
• Example
  – Optional features: Light, Beeper
  – Consider impact on objects
  – Consider impact on statechart
Feature-Based Impact Analysis

- Impact Analysis of **Beeper** and **Light** optional features
  - 2 optional objects added
  - Impact on control object
    - Feature dependent messages

8.4a [beeper]: Beep

6.2b [light]: Switch On
8.4b [light]: Switch Off

Copyright 2011 H. Gomaa
Feature Based State Machine

Feature conditions: [light], [beeper]
Feature dependent actions: Switch On [light], Switch Off [light], Beep [beeper]
Software Product Line Design Modeling
What should SPL Design Method provide?

- Software Architectural Patterns
- Evolutionary Component-Based Software Architectures
Develop Kernel Software Architecture
(Example based on Centralized Control Pattern)

- One control component
  - Executes statechart
- Receives sensor input from input components
- Controls external environment via output components

Key
Design of Optional Components

- Feature dependent interfaces & components

```
IBeeper

- Feature dependent interfaces & components

```

```
interface IBeeper
{feature = Beeper}

initialize ()
beep ()

PLamp

- Feature dependent interfaces & components

```

```
interface ILamp
{feature = Lamp}

initialize ()
switchOn ()
switchOff ()

ILamp

- Feature dependent interfaces & components

```

```
optional IBeeper

- Feature dependent interfaces & components

```

```
optional ILamp

- Feature dependent interfaces & components

```

```
output component IBeeperComponent

- Feature dependent interfaces & components

```

```
output component ILampComponent

- Feature dependent interfaces & components

```

```
output component ILampComponent

- Feature dependent interfaces & components

```

```
output component ILampComponent

- Feature dependent interfaces & components

```

```
output component ILampComponent

- Feature dependent interfaces & components

```

```
output component ILampComponent

- Feature dependent interfaces & components

```

Copyright 2011 H. Gomaa
Feature-based Microwave Oven Architecture

- Feature dependent components, connectors and messages
- Role and reuse UML stereotypes
Application Engineering

Informal SPL Requirements → Domain (Product Line) Engineering → SPL models and architecture, Reusable components

Domain (Product Line) Engineering → Product Line Reuse Library

Application Requirements → Application Engineering

Application Engineering → Executable Application

Unsatisfied Requirements, Errors

Copyright 2011 H. Gomaa
Software Application Engineering

- Software Application
  - Member of software product line
- Software Application Engineering
  - Derive application architecture from SPL architecture
- Select application features subject to
  - Feature dependencies and relationships
- Derive software application architecture
  - Kernel components always selected
  - Optional and variant components correspond to features selected
Microwave Oven Application architecture

«product line system»
MicrowaveOvenSystem

«kernel»
«input component»
DoorComponent

RMWControl

«kernel»
«input component»
BooleanWeightComponent

RMWControl

«kernel-param-vp»
«input component»
KeypadComponent

RMWControl

«default»
«input component»
PMWControl

PMWControl

«kernel»
«control component»
RMWControl

RDisplay

PBeeper

RLamp

RHeater

PDisplay

PLamp

PHeater

«default»
«output component»
One-levelHeatingElementComponent

«optional»
«output component»
LampComponent

«optional»
«output component»
BeeperComponent

«variant»
«output component»
Multi-lineMicrowaveDisplay

Copyright 2011 H. Gomaa
Conclusions

• Software Product Line Engineering
  – Software engineering for a family of products
• Software Variability
  – Key problem in software product line engineering
• Feature modeling
  – Unifying view in multiple view SPL
  – Feature relationships with other views explicitly depicted
• Evolutionary development
  – Kernel first approach for new SPL
  – Reverse SPL engineering from existing systems
  – Use feature modeling to model evolution of requirements
Related Research

- Multiple-view modeling and meta-modeling for SPL (with L. O’Hara, M. Shin, M. Abu-Matar)
- Software process modeling and SPL (with L. Kerschberg)
- Tool support for SPL development and product derivation (with V. Sugumaran, M. Shin, M. Abu-Matar)
- Model-based SPL Testing (with E. Olimpiew)
- Aspect-oriented modeling and SPL (with M. Shin)
- Executable Architectural Patterns for RT SPLs (with J. S. Fant)
- Feature Modeling & Variability Modeling in Service-oriented SPL (with M. Abu-Matar)