Use Case Modeling for Software Product Lines

Hassan Gomaa
Department of Information and Software Engineering
George Mason University

Reference: Hassan Gomaa, Chapter 4 in “Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures”, Addison-Wesley Object Technology Series, 2005

Copyright © 2005 Hassan Gomaa
All rights reserved. No part of this document may be reproduced in any form or by any means, without the prior written permission of the author.

Evolutionary Process Model for Software Product Lines
Requirements Modeling
What should SPL Design Method provide?

- Support variability in use case modeling
- Integrate feature modeling with other UML views

Use Case Modeling for Single Systems

- Define system functional requirements in terms of Actors and Use cases
  - Each use case defined in terms of sequence of interactions between Actor and System
    - Narrative description
  - Use case is a complete sequence of interactions initiated by an actor
    - Use case starts with input from an actor
    - Describes interactions between actor and system
    - Provides value to actor
    - Basic path
      - Most common sequence
    - Alternative branches
      - Variations of basic path
      - E.g., for error handling
    - Figure 4.1 Example of actor & use cases
Actors

• Actor models external entities of system
• Actor interacts directly with system
  – Human user
  – External I/O device
  – External system
  – Timer
• Primary actor initiates use case
• Secondary actor participates in use case
• Identifying use cases from actors
  – Consider each major function an actor needs to perform
  – Provides value to actor
• Examples of actors: Figs. 4.2-4.4

Use Case Modeling for Software Product Lines

• Single Systems
  – All use cases required
• Software product lines
  – Categorize use cases using UML stereotypes
  – Kernel use cases
    • Used by all members of SPL
    • <<kernel>>
  – Optional use cases
    • Used by some members of SPL (e.g., Fig. 4.5)
    • <<optional>>
  – Alternative use cases
    • Choice of mutually exclusive use cases (e.g., Fig. 4.6)
    • <<alternative>>
Documenting Product Line Use Cases

- Name
- Reuse Category
- Summary
- Actors
- Dependency (on other use cases)
- Preconditions
- Description
  - Narrative description of basic path
- Alternatives
  - Narrative description of alternative paths
- Variation Points
  - Places where different functionality can be added
- Postcondition

Example of Use Case Modeling for Software Product Lines

- Example of Microwave Oven Product Line
  - Actors
    - User
    - Timer
  - Use Cases
    - Kernel use case
      - Cook Food
    - Optional use cases
      - Set Time of Day
      - Display Time of Day
      - Cook Food with Recipe
    - E.g., Figure 4.5: Microwave Oven Product Line use cases
Variation Points in a Use Case Model

• Variation Point
  – Location in a use case where a change can take place
• Variation Point can be handled by
  – Variation Point within use case
    • Identify line # in use case where variability can be introduced
  – Conditional use case relationship
    • Extend or include use case if *product line condition* is True

Documenting Small Variations

• For each variation point within use case
  – *Name* of variation point
  – *Type of functionality* to be inserted at variation point
    • Optional
    • Mandatory alternative
    • Optional alternative
  – *Line number in use case* where variability introduced
  – *Description of functionality* inserted at variation point
    • Optional functionality
    • Alternative functionality
Documenting Small Variations

• Example of optional functionality
  – **Name**: Light.
  – **Type of functionality**: Optional.
  – **Line number(s)**: 1, 5, 8, 9.
  – **Description of functionality**: If Light option is selected, light is switched on for duration of cooking and when the door is open. Light is switched off when the door is closed and when cooking stops.

• Example of alternative functionality
  – **Name**: Display Language.
  – **Type of functionality**: Mandatory alternative.
  – **Line number(s)**: 3, 8.
  – **Description of functionality**: There is a choice of language for displaying messages. The default is English. Alternative mutually exclusive languages are French, Spanish, German, and Italian.

Modeling Variability with Extend Relationship

• **Extend** relationship
  – Extend use case if given *extension condition* holds

• Extension Point
  – Precise location in base use case at which extensions can be added
    • E.g., <payment>

• Extension use case
  – Insertion segment
    • Behavior sequence to be executed when extension point reached **IF** *extension condition* is True

• Selection Condition
  – Condition for selecting extension use case is **True** **during** execution

• Product line condition
  – Variable SPL functionality provided if *product line condition* is **True**

• For optional or alternative use cases, extension use case is executed
  – If [**selection condition** **AND** *product line condition*] = **True**
  – E.g., Fig. 4. 9: credit card option, debit card option
Modeling Variability with Include Relationship

- **Include** relationship
  - Identify common patterns (sequences) in several use cases
    - Extract common pattern into **abstract use case**
    - Concrete use cases **include** abstract use case
  - Structure a lengthy use case
    - Concrete use case provides high level sequence of interactions between actor(s) and system
    - Abstract use cases provide lower level sequences of interactions between actor(s) and system
    - E.g., Fig. 4.11
- **Include** relationship in product lines
  - Optional use case includes kernel abstract use case if *product line condition* is True
    - E.g., Fig. 4.10, 4.12: deposit option

Developing Use Case Model for Software Product Lines

- **Forward Evolutionary Engineering**
  - **Kernel First Approach**
    - Develop kernel use cases that are common to all members of the software product line
- **Reverse Evolutionary Engineering**
  - Specify use cases for each member of product line
  - Compare use cases for different members
  - Determine kernel, optional, and alternative use cases
- **Product Line Evolution Approach**
  - Evolutionary part of Forward & Reverse Evolutionary Engineering
  - Consider impact of each optional and alternative use case on kernel use case model
Developing Use Case Model for Software Product Lines - Forward Evolutionary Engineering

- Kernel First Approach
  - Develop kernel use cases initially
- Example of Kernel First Approach (E.g., Fig. 4.5)
  - Kernel use case in Microwave Oven Product Line
    - Cook Food (use case description – page 74)
- Product Line Evolution Approach
  - Address Product Line use case variability
  - Cook Food kernel use case
    - Several variation points
  - Optional use cases in Microwave Product Line
    - Set Time of Day
    - Display Time of Day
    - Cook Food with Recipe

Cook Food Use Case - Variation Points

- Lines 3,8: Display Language – Mandatory alternative
  - Default = English
  - Alternatives = French, Spanish, German, Italian
- Line 1: Weight Sensor – Mandatory alternative
  - Default = Boolean weight
  - Alternative = Analog weight
- Line 5: Heating Element – Mandatory alternative
  - Default = One-level Heating Element
  - Alternative = Multi-level Heating Element
- Line 2: Power level – Optional
  - Power level buttons = high (default), medium, or low.
Cook Food Use Case - Variation Points

- Lines 3,4,6,8,10: Display Unit - Mandatory alternative
  - Default = one-line display
  - Alternative = multi-line display
- Lines 2, 6: Minute Plus. Optional
  - Add one minute to cooking time.
  - Start cooking if not yet cooking.
- Lines 1,5,8,9: Light – Optional
  - Lamp switched on when cooking & when door is open.
- Lines 5,8: Turntable – Optional
  - Turntable rotates for duration of cooking
- Line 8: Beeper – Optional
  - Switch Beeper on when cooking stops

Cook Food with Recipe Use Case - Variation Points

- Lines 3,8: Display Language – Mandatory alternative
  - Default = English
  - Alternatives = French, Spanish, German, Italian
- Lines 1,5,8,9: Light – Optional
  - Lamp switched on when cooking & when door is open.
- Lines 5,8: Turntable – Optional
  - Turntable rotates for duration of cooking
- Line 8: Beeper – Optional
  - Switch Beeper on when cooking stops
Use Case Model for
Software Product Lines
- Reverse Evolutionary Engineering

• Specify use cases for each member of Product Line
• Compare use cases for different members
• Determine kernel, optional, and alternative use cases
  – Kernel use cases needed by all Product Line members
  – Optional use cases needed by some Product Line members
  – Alternative use cases are mutually exclusive
• Example: Figures 4.13 -4.15: E-Commerce SPL