Dynamic Interaction Modeling in Single Systems

- Use cases realized in Dynamic Model
  - Show objects participating in each use case
- Determine how objects participate in use case
  - Use object structuring criteria to determine objects
    - Stereotype for each object structuring criterion
    - Fig. 6.13
  - Shows sequence of object interactions in use case
    - Depict on communication (collaboration) diagram or sequence diagram
- Message sequence description
  - Narrative description of sequence of object interactions
- Dynamic Analysis
  - Approach to determine how objects interact with each other to support use case
Communication Diagram

• Called Collaboration Diagram in UML 1.x
• Graphically depicts objects participating in a use case
  – Show objects as boxes
  – Show their message interactions as arrows
  – Number sequence of messages
• Communication Diagram(s) developed for each use case
  – Main sequence
  – Alternative sequence
• Instance form
• Generic form
• Example: Fig. 7.1

Sequence Diagram

• Shows sequence of object interactions in use case
• Emphasis on messages passed between objects
  – Objects represented by vertical lines
    • Actor is on extreme left of page
  – Messages represented by labeled horizontal arrows
    • Only source and destination of arrow are relevant
    • Message is sent from sending object to receiving object
  – Time increases from top of page to bottom
  – Spacing between messages is not relevant
  – Message sequence numbering is optional
• Example: Fig. 7.2
Dynamic Interaction Modeling for Software Product Lines

- Object interaction model
  - Defines how objects participate in use cases
  - Use communication (collaboration) diagrams or sequence diagrams
- Software Product Line Interaction model
  - Determine objects that participate in each use case
  - Determine sequence of messages sent between objects
  - Develop at least 1 interaction diagram for each use case
    - Kernel communication diagrams
    - Optional communication diagrams
    - Alternative communication diagrams
  - Develop feature based communication diagrams
    - Communication diagram for each functional feature

Dynamic Analysis for Software Product Lines

- Kernel First Approach
  - Communication diagrams for kernel system are similar to Communication diagrams for single system
  - Only kernel objects participate in kernel Communication diagrams
    - May also need to consider default objects
  - Kernel system is a minimal member of product line
    - Kernel objects
    - Kernel objects + default objects
- Software Product Line Evolution
  - Start with kernel Communication diagrams
  - Consider optional and alternative Communication diagrams
Kernel First Approach -
Non-State Dependent Dynamic Analysis

- Start with use case
- Determine interface objects
  - Receives external events from actor
- Determine internal objects
  - Receive messages from interface objects
- Determine object communication
  - Sequence of messages passed
- Consider alternative branches
  - For error handling or less frequently occurring conditions
- Consider variation points
  - Variant communication diagrams
- Examples from Factory Automation (Fig 7.1) and E-Commerce (Fig 7.3) product lines

Software Product Line Evolution -
Dynamic Analysis of Optional Communication Diagrams

- Develop optional Communication diagrams
  - For optional use cases involving optional objects
  - Use separate optional Communication diagram
    • Depict optional objects
      – Example from E-Commerce (Fig 7.4) product line
- Develop optional branches on kernel Communication diagrams
  – Optional objects can interact with kernel objects
Software Product Line Evolution -
Dynamic Analysis of Alternative Communication Diagrams

• Develop alternative Communication diagrams
  – For alternative (mutually exclusive) use cases
  – Use separate alternative Communication diagram
    • Depict optional and/or variant objects
  – Example from E-Commerce (Fig 7.5) product line
• Develop alternative branches on kernel Communication diagrams
  – Variant and optional objects can interact with kernel objects
• Based on feature dependencies
  – Some optional / alternative communication diagrams may need combination of optional and variant objects

Example of Evolutionary Dynamic Analysis
for Software Product Lines

• Kernel first approach
  – Determine objects in kernel use case
    • Determine kernel objects
      – Objects used by every member of product line
    • Determine default objects
      – Variant objects, one of which is needed by every member of product line
  – Develop kernel collaboration diagram
    • E.g., Cook Food use case
Microwave Oven SPL Kernel Objects

- Input device interface objects
  - Door Sensor Interface
  - Weight Sensor Interface
  - Keypad Interface
- Output device interface objects
  - Heating Element Interface
  - Display Interface
- Control objects
  - Microwave Oven Control
  - Oven Timer
- Entity objects
  - Oven Data
  - Display Prompts

Microwave Oven SPL Default objects

- Default objects from kernel use case and variation points
  - Boolean Weight Sensor Interface
  - One-level Heating Element Interface
  - One-line Display Interface
  - English Display Prompts
- Develop kernel communication diagram
  - Fig. 7.6
Feature Based Impact Analysis

• Kernel First Approach
  – Develop kernel interaction diagrams to realize kernel use cases
• Product Line evolution approach
  – Consider impact of optional and alternative features on kernel
• Analyze impact of each feature
  – Optional object(s) can be added
  – Variant object can replace default object
  – Determine impact on existing kernel objects
    • Communicate with optional / variant objects

Example of Evolutionary Dynamic Analysis for Software Product Lines

• Example of Product Line Evolution Approach
• Feature-based Impact Analysis
  – Depict impact of each feature on separate communication diagram
• Optional feature corresponding to variation point
  – E.g., Beeper
  – Add optional object
    • Impact on Kernel communication diagram (Fig 7.7)
    • Variant branch (Fig 7.8)
Example of Evolutionary Dynamic Analysis for Software Product Lines

• Example of Product Line Evolution Approach
• Feature-based Impact Analysis
  – Depict impact of each feature on separate communication diagram
• Alternative feature corresponding to variation point
  – E.g., Display Language
  – Replace default object with variant object
  – Variant branch (Fig 7.9)

Impact of Variation Points and Features

• Impact of Variation Points that lead to new concurrent actions
  – E.g., Beeper
  – Figures 7.7, 7.8
• Impact of Variation Points that lead to additional sequential actions
  – E.g., Minute Plus (during cooking)
  – Figure 7.10
• Impact of Variation Points that lead to alternative sequential actions
  – E.g., Minute Plus (before cooking)
  – Figure 7.11
Impact of Optional Use Cases and Features

• Impact of new features resulting from optional use cases
  – E.g., TOD Clock feature captured by 2 use cases
    • Set Time of Day
    • Display Time of Day
  – Figure 7.12