Overview

• Classes and Variation Points
  – Abstract classes
  – Parameterized classes
• Class Reuse Categorization for SPLs
• Feature/Class dependencies
• Feature-based Impact Analysis
• Feature/Object and Feature/Class dependency modeling in UML
• Microwave Oven SPL case study
Variation Points

- A variation point identifies a location at which change will occur in a software product line
- The variation point also identifies the mechanism for a reuser to make the change
- Variation point mechanisms
  - Abstract classes and inheritance
    - Abstract superclass
    - Specialized differently for various members of SPL
  - Parameterized classes
    - Configuration Parameters
    - Different parameter values for different SPL members

Abstract Class

- Abstract Class
  - Template for creating subclasses
  - Has no instances
  - Only used as superclass
  - Defines common interface for subclasses
- Abstract operation
  - Operation declared in abstract class but not implemented
- Abstract Class defers implementation of some or all of its operations to subclasses
- Different subclasses can define different implementations of same abstract operation
Example of Abstract Class and Inheritance

• Attributes of Account Superclass
  – accountNumber, balance
• Operations of Account Superclass
  – open (accountNumber : Integer)
  – close ()
  – readBalance () : Real
  – credit (amount : Real) {abstract}
  – debit (amount : Real) {abstract}
• Example: Fig. 9.1

Example of Abstract Class and Inheritance

• Attributes of Checking Account Subclass
  – Inherits accountNumber, balance
  – Adds lastDepositAmount
• Operations of Checking Account Subclass
  – Inherits specification and implementation of
    • open, readBalance, close
  – Inherits specification of credit and debit
    • Defines implementation
      • credit sets lastDepositAmount = amount
  – Adds new operation
    • readLastDepositAmount () : Real
Example of Abstract Class and Inheritance

- Attributes of `Savings Account` Subclass
  - Inherits `accountNumber, balance`
  - Adds instance attributes `cumulativeInterest, debitCount`
  - Adds static class attributes `maxFreeDebits, bankCharge`

- Operations of `Savings Account` Subclass
  - Inherits specification and implementation of
    - `open, readBalance, close`
  - Inherits specification of `credit` and `debit`
  - Defines `debit`
    - Debit balance and deduct `bank Charge` if `debit Count > max Free Debits`
  - Adds Operations
    - `addInterest (interestRate)` Add daily interest
    - `readCumulativeInterest () : Real`
    - `clearDebitCount ()` Reinitialize debit Count to zero

Example of Parameterized Class

Oven Data (Fig. 9.2)

- **Kernel attribute:** `cookingTime`
  - Remaining time to cook food
- **Optional attributes,**
  - Relevant only if named feature is selected
  - `selectedPowerLevel {feature = Power Level}.`
    - Possible values: High, Medium, and Low; initial value is High
    - If feature not selected, power level is constant set to High
  - `itemWeight {feature = Analog Weight}`
    - Records actual weight of the item
  - `selectedRecipe {feature = Recipe}`
    - Initialized to “none selected”
  - `TODvalue {feature = TOD Clock}`
    - Time variable initialized to 12:00
  - `TODmaxHour {feature = TOD Clock}`
    - Parameterized constant set at system configuration to 12:00 or 24:00
**Variant vs. Parameterized Classes**

- Abstract superclass and variant subclasses
  - Advantage
    - Isolates each variation in one variant class
    - A variant class is impacted by only one feature
  - Disadvantage
    - Could lead to combinatorial explosion of variant classes
- Parameterized Classes
  - Advantage
    - One parameterized class instead of many variant classes
  - Disadvantage
    - A parameterized class can be impacted by more than one feature

**UML Modeling for Software Product Lines**

- Depict class and object categorization using UML stereotypes
- **Stereotype** defines
  - New modeling element derived from existing UML modeling element
  - Tailored to modeler’s problem
  - Depicted using guillemets
    - «entity», «interface», «control»
- UML 1.4 upwards supports multiple stereotypes for class
  - Use UML stereotypes to depict reuse category
  - Use UML stereotypes to depict application role category
Static Modeling for Single Systems

- UML 1.4 upwards supports multiple stereotypes for a modeling element
- Single systems (COMET)
  - Categorize each class by application role using stereotype
    - «control», «entity», «interface»
  - Object has same application role stereotype as class it is instantiated from

Static Modeling for Software Product Lines

- UML 1.4 upwards supports multiple stereotypes for a modeling element
- Single systems
  - Categorize each class by application role using stereotype
    - «control», «entity», «interface»

Software Product Lines (PLUS)
- Second UML stereotype depicts reuse category
  - «kernel», «optional», «variant»

Copyright 2005 H. Gomaa
Product Line Class Reuse Categorization

- \texttt{<<kernel>>} Kernel class
  - Provided by every member of the Product Line (SPL)
  - Used without change
- \texttt{<<optional>>} Optional class
  - Provided by some members of the Product Line
  - When used, it is used without change
- \texttt{<<variant>>} Variant class
  - One of a set of similar classes provided by SPL
  - Some identical properties, others are different
- \texttt{<<default>>} Default class
  - Default class among set of variant SPL classes
- Product Line Class Reuse Categorization (Fig. 9.3)
  - Examples (Fig. 9.4)

Software Product Line Class Categorization

- Abstract or parameterized class can be source of variability
  - Class defines a variation point
  - Explicitly defined in name of stereotype - vp
- Abstract class reuse categories
- \texttt{<<kernel-abstract-vp>>} abstract kernel class
  - Provided by every member of the family
  - Abstract class provides standard interface for its subclasses
  - SPL variability introduced through specialization
- \texttt{<<optional-abstract-vp>>} abstract optional class
  - Provided by some members of the Product Line
  - Abstract class provides standard interface for subclasses
Software Product Line Class Categorization

Parameterized class reuse categories

- <<kernel-param-vp>> parameterized kernel class
  - Provided by every member of the family
  - Configuration parameters need to be set by SPL member
  - SPL variability introduced through parameterization
- <<optional-param-vp>> parameterized optional class
  - Provided by some members of the Product Line
  - Configuration parameters need to be set by SPL member
- <<variant-param-vp>> parameterized variant class
  - One of a set of variant classes provided by SPL
  - Configuration parameters need to be set by SPL member
- <<default-param-vp>> parameterized default class
  - Default class among set of variant SPL classes
  - Configuration parameters need to be set by SPL member

Concrete class reuse categories

- <<kernel-vp>> concrete kernel class
  - Provided by every member of the family
  - Class provides a standard interface for its subclasses
  - SPL variability introduced through specialization
- <<optional-vp>> concrete optional class
  - Provided by some members of the Product Line
  - Class provides a standard interface for its subclasses
- <<variant-vp>> concrete variant class
  - One of a set of variant classes provided by SPL
  - SPL variability introduced through specialization
Feature / Class Dependencies

• For every feature in Software Product Line
  – Determine classes that depend on feature

• Common feature
  – Kernel classes
  – Always selected for member of Software Product Line

• Optional and Alternative features
  – If feature is selected for member of Software Product Line
  – Then classes supported by this feature are also selected

Feature / Class Dependencies

• For each optional or alternative feature, define
  – Optional and/or variant classes required to support feature
  – If feature has prerequisite features
    • Classes in feature may use classes in prerequisite features
  – Features determine which classes can coexist in same system

• Some variants are mutually exclusive
  – Used by different members of product line
  – E.g., High Volume Workstation Controller, Flexible Workstation Controller, Monitoring Workstation Controller

• Some variants co-exist in same system
  – E.g., High Volume Workstation Controller, Receiving Workstation Controller, Shipping Workstation Controller
  – E.g., Fig. 9.5
Feature Based Impact Analysis

- Kernel First Approach
  - Develop kernel communication diagrams to realize kernel use cases
- Product Line evolution approach
  - Consider impact of optional and alternative features on kernel communication diagrams
- Analyze impact of each feature on kernel
  - Consider optional objects that need to be added
  - Consider replacement of default objects with variant objects
  - Consider impact on existing kernel objects
    - Communicate with optional / variant objects

Feature Based Impact Analysis
Examples from Microwave Oven SPL

- Microwave Oven Control (Fig. 8.15)
- Example of new actions added
  - Feature: Turntable
    - Actions: Start Turning, Stop Turning
  - Feature: Light feature
    - Actions: Switch On, Switch Off
  - Feature: Beeper
    - Action: Beep
Feature-Based Communication Diagram

- Communication diagram
  - Depicts objects required to support a use case
- Feature-based communication diagram (e.g., Fig. 9.7)
  - Feature supports one or more use cases
  - Depicts objects required to support a feature
  - For feature supporting more than one use case
    - Objects are determined from more than use case
      based communication diagram
- Interconnected objects supporting one feature
  - Depend on objects in prerequisite features
  - Can communicate with objects in prerequisite feature-based communication diagrams

Feature / Class Dependencies

- Feature-based class diagram
  - Depict classes that depend on feature
- Feature dependencies
  - Map to relationships between classes
    - Association
    - Specialization
- Feature / Class Dependency Table
  - Columns
    - Feature name
    - Feature category
    - Class name
    - Class reuse category
    - Class parameter (affected by feature)
  - E.g., Table 9.1, 13.3
Dynamic Modeling 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 communication diagram
    - E.g., Cook Food use case
    - Figure 9.6

Microwave Oven SPL Kernel Classes

- Input Device Interface Classes
  - Door Sensor Interface
  - Weight Sensor Interface
  - Keypad Interface
- Output Device Interface Classes:
  - Heating Element Interface
  - Display Interface
- Control Classes
  - Microwave Oven Control
  - Oven Timer
- Entity Classes
  - Oven Data
  - Display Prompts
Microwave Oven SPL
Default Classes

• Default classes from kernel use case and kernel features
  – Boolean Weight Sensor Interface
  – One-level Heating Element Interface
  – One-line Display Interface
  – English Display Prompts

Feature Based Impact Analysis

• Product Line evolution approach
  – Consider impact of optional and alternative features on kernel communication diagram
  – Feature Based Impact Analysis
    • Analyze impact of each feature on kernel communication diagram

• Optional features
  – Analyze impact of each feature on kernel communication diagram
    • Consider optional objects that need to be added
    • Consider replacement of default objects with variant objects
    • Consider impact on existing kernel objects that communicate with optional / variant objects
Impact Analysis of Optional Microwave Oven Features

- **<<optional feature>> Light**
  - New object: Lamp Interface
  - Affected object: Microwave Oven Control
- **<<optional feature>> Turntable**
  - New object: Turntable Interface
  - Affected object: Microwave Oven Control
- **<<optional feature>> Beep**
  - New object: Beep Interface
  - Affected object: Microwave Oven Control
- **<<optional feature>> Minute Plus**
  - Affected objects: Keypad Interface, Microwave Oven Control, Oven Timer

Feature Based Impact Analysis

- **Alternative feature**
  - Analyze impact of each feature on kernel communication diagram
    - Consider additional optional objects resulting from feature
    - Consider replacement of default object with variant objects resulting from feature
    - Consider impact on existing kernel objects
Impact Analysis of Alternative Features

• «exactly-one-of feature group>> Weight {default = Boolean Weight, alternative = Analog Weight}
  – « alternative feature>> Analog Weight
    • Variant object: Analog Weight Sensor Interface
    • Affected object: Oven Data
• «exactly-one-of feature group>> Display Unit {default = One-line Display, alternative = Multi-line Display}
  – « alternative feature>> Multi-line Display
    • Variant object: Multi-line Display Interface
• «exactly-one-of feature group>> Heating Element {default = One-level Heating, alternative = Multi-level Heating}
  – « alternative feature>> Multi-level Heating
    • Variant object: Multi-level Heating Interface

Affected object: Microwave Oven Control, Oven Data

Impact Analysis of Alternative Features

• «exactly-one-of feature group>> Display Language
  {default = English, alternative = French, Spanish, German, Italian}
  – « alternative feature>> French,
    • Variant object: FrenchDisplayPrompts
  – « alternative feature>> Spanish,
    • Variant object: SpanishDisplayPrompts
  – « alternative feature>> German,
    • Variant object: GermanDisplayPrompts
  – « alternative feature>> Italian,
    • Variant object: ItalianDisplayPrompts
Impact Analysis of Optional Features with Prerequisites

- "optional feature>> Power Level \{prerequisite = Multi-level Heating\}
  - Affected objects: Keypad Interface, Microwave Oven Control, Oven Data
  - Prerequisite object: Multi-level Heating Interface

- "optional feature>> Recipe \{prerequisite = Analog Weight, Multi-level Heating, Multi-line Display\}
  - Optional objects: Recipes, Recipe
  - Affected objects: Keypad Interface, Microwave Oven Control, Oven Data, Oven Timer
  - Prerequisite objects: Multi-level Heating Interface, Multi-line Display Interface, Analog Weight Sensor Interface

Impact Analysis of Features with Prerequisites

- Optional Feature with Prerequisite
  - "optional feature>> TOD Clock \{prerequisite = Multi-line Display\}
    - Optional object: TOD Timer
    - Affected objects: Keypad Interface, Microwave Oven Control, Oven Data
    - Prerequisite object: Multi-line Display Interface

- Alternative Feature with Prerequisite
  - «parameterized feature» 12/24 Hour Clock \{type = Time, permitted value = 12:00, 24:00, default value = 12:00, mutually includes = TOD Clock\}
    - Affected object: Oven Data
      - Stores configuration parameter TODMaxHour