Implementation of Product Line Architectures

Instructor: Yongjie Zheng
November 9, 2015

CS 5555: Software Architecture and Design
Implementation of PLAs: Context

• Takes place in domain engineering

• Input: a PLA model that consists of common elements and variation points.

• Output: domain implementations (e.g. source code, components) that can be reused to create a single product.

• Activities: mapping of architectural constructs to code, variation refinement.
Implementation of PLAs: Considerations

- **Reusability**: the domain implementations should be reusable in the development of family products.

- **Traceability**: relating variability defined in different domain artifacts (e.g. from PLA to source code).

- **Separation of concerns**: it is preferred to separate concerns in the implementation artifacts along the boundaries of PLA variation points.

- **Integration of legacy code**.
Important to partition implementations along variation point boundaries.
Existing Implementation Techniques

• Clone-and-own, or code scavenging
• Feature-oriented software development
• Object-oriented programming mechanisms (e.g. inheritance)
• Other programming techniques: procedure parameters, C++ templates, conditional compilation
• Object-oriented frameworks
• Generative mechanisms
Clone-and-Own

• This is what most people are doing when it goes to software reuse - opportunistic reuse.

• Potential problems
  • Significant changes often have to be made to the existing code due to, for example, *architecture mismatch*.
  • Maintenance of duplicated code.
  • Improperly sharing ancestor’s characteristics in descendant programs.

• In many cases, clone-and-own is used not because it is effective, but mainly because it is the only reusable mechanism that is available.
Feature Oriented Software Development (FOSD)

• A program paradigm mainly developed by Prof. Don Batory at University of Texas, Austin around 1990s and 2000s.

• Started from the concept of layer-based design and collaboration-based design.

  • “A program was a stack of layers. Each layer added functionality to previously composed layers and different composition of layers produced different programs.”

• Emphasizes the concept of feature modularity.
FOSD, cont.

- Each feature module only contains fragments of a set of classes that are related to a feature.

- Two kinds of feature modules: *constants* and *refinements*.

- Each refinement encapsulates the implementation of a single feature, and refines a constant by
  - Adding new members to the existing classes.
  - Adding new classes.
  - Extending existing methods by using wrappers such as overriding.

- Feature modules are composed to generate complete system.
Feature-Oriented Software Development (FOSD)

Source: http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/slides/featureide-0-background.pdf
More about FOSD

- FOSD is a typical compositional development approach.
- Software design in FOSD is reduced to an algebraic expression.
- Challenges: the process of feature composition may get difficult when the software size increases.
- Similar approaches: aspect-oriented programming, multi-dimensional separation of concerns, subject-oriented programming.
- All these approaches embrace the principle of information hiding.
Object-Oriented Programming Techniques

- Techniques in this category are usually referred to as *incremental programming*, and fall into the area of programming languages.

- Object-oriented extension approaches: inheritance, mixins, nested inheritance, open classes, classbox, Jiauzzi, etc.

- Inheritance: a relationship between implementations (i.e. classes).
More about Inheritance

• Represents *Is-a* relationship.

• An important **code reuse** mechanism: subclasses reuse code of the abstract, or parent class.

  • Overriding may be used to replace the implementation of an inherited behavior, and define an application-specific behavior

• Scalability issue: inheritance operates on one class at a time, and is insufficient for organizing programs in terms of reusable software components.
More about Inheritance, cont.

• Extensibility problem: adding new subclasses (i.e. variant, type) v.s. adding new operations.

With inheritance, it is easy to add new variant, but hard to add new operations.

Other Programming Techniques

- Procedure parameters
- C++ template
- Conditional compilation

An Example of C++ Template.
Source: http://www.cplusplus.com/doc/tutorial/templates/

An Example of Conditional Compilation.
Some other mechanisms

• Object-oriented frameworks
  • White box and black box

• Code generation based approaches
  • Also known as model-driven approaches
  • Should code generation be involved in variability management?

• Design patterns
  • Visitor, strategy, observer, factory, etc.
Recall

• Clone-and-own, or code scavenging
• Feature-oriented software development
• Object-oriented programming mechanisms (e.g. inheritance)
• Other programming techniques: procedure parameters, C++ templates, conditional compilation
• Object-oriented frameworks
• Generative mechanisms