UML-F - A modeling language for object-oriented frameworks

Marcus Fontoura, Wolfgang Pree, Bernhard Rumpe
http://uml-f.org

Agenda

• Motivation example
• Proposed solution
• UML-F
• Framework implementation
• Framework instantiation
• Related work
• Conclusions & future work
Motivation example

Motivation example (i)

- Web-based educational system
- Requirements for the student subsystem (condensed example)
  - Several course selection mechanisms (e.g. require login, show all courses, show only the courses in which the student is enrolled)
  - Actions before the exhibition of the course content (e.g. tip of the day, course announcement)
Motivation example (ii)

- Solution 1 (in UML diagrams)

Motivation example (iii)

- Problems with solution 1:
  - Identification of variation points
  - Instantiation process is not clear
  - Border between framework and application
  - Maintenance
    - What happens if the definition of variation points change?
Motivation example (iv)

• Solution 2 (based on design patterns)

```java
void showCourse()
int selectCourse()
showContent(int)
```

```java
void op()
```

```java
void op()
```

Motivation example (v)

• Problems with solution 2:
  – Identification of variation points and instantiation process are clear (only if we know what patterns have been applied)
  – More complex design (especially for patterns based on recursive composition)
  – Maintenance
The proposed solution

The proposed solution (i)

• Problem generalization:
  – Identification of variation points
  – Assist development (how to implement the variation points)
  – Assist instantiation (how to instantiate the framework)
  – Assist maintenance (tool support)
The proposed solution (ii)

Framework

Implementation Phase

Framework Design

Framework

Instantiation Phase

Applications

Tool support

Framework design language

UML-F
UML-F (i)

- Functional view of the design
- Distinguish variation points from kernel
- Classify variation points (semantics)
- Instantiation restrictions
- Simple
- Allow the definition of new kinds of variation points when they are discovered (e.g. variations in structure)

UML-F (ii)
UML-F (iii)

- Sequence diagram template for \textit{selectCourse()}

Framework implementation
Framework implementation (i)

- Design patterns

```
ConcreteExtension [appl-class]
  + void op()

ConcreteSelect [appl-class]
  + int select()

ShowCourse [separation, template]
  +SelectedCourse
  +void showCourse()
  +int selectCourse()
  +showContent(int)

SelectStrategy [separation, hook]
  + void select() (dynamic)
```

**Extensible interface variation points**

Framework implementation (ii)

- Method parametrization (and MOPs)

```
ConcreteExtension [appl-class]
  + void op()

ConcreteSelect [appl-class]
  + int select()

ShowCourse [separation, template]
  +SelectedCourse
  +void showCourse()
  +int selectCourse()
  +showContent(int)

SelectMOP
  + void selectMOP(boolean login, boolean validate)
  +int select()
```

**respect instantiation restrictions (not automatic!)**
Framework implementation (iii)

- Several other implementation techniques may be used
- Transformational tools assist maintenance

Framework instantiation
Framework instantiation (i)

- ExtensionMethods (separation, hook)
  - extend (void opt) [dynamic]

- ShowCourse (separation, template)
  - SelectedCourse
  - void showCourse()
  - int selectCourse()
  - showContent(int)

- SelectStrategy (separation, hook)
  - select (int select)

TipOfDay (separation, c-hook)

Announcement (separation, c-hook)

SimpleSelect (separation, c-hook)

LogInSelect (separation, c-hook)

“Complete generalizations”

Framework instantiation (ii)

- UML-F descriptions are formal “cookbooks”
- Process-based: tool executes the “cookbooks”
Related work

• UML Collaborations & Catalysis
• Role modeling
• Design pattern tools
• Contracts & APPCs
• MOP, AOP, SOP
Conclusions and future work

Conclusions

• More than 5 large experiments (UML-F Book)
• Better design representation allows systematization of the further steps in the framework development process
• And leads to a better requirements elicitation phase
• Classification of frameworks variation points (semantics)
• Transformational tools (basis for the development of new tools)
Future work

- Architectural patterns
- Automatic derivation of the design from the requirements artifacts (commonality analysis)
- New implementation techniques
- Visual tools