PhD Thesis Progress Report

http://d3s.mff.cuni.cz

Michal Malohlava
michal.malohlava@d3s.mff.cuni.cz
Clarification

• Why?
  - Self-motivation to finish :-)  
  - Your feedback is important

• What do I expect?
  - What is your opinion?
  - Is it sound?
  - Any observation, notes,...
Thesis Overall Strategy

Existing papers + Glue text = PhD thesis

Prototype
Selected papers (7/11+1) cover following domains

- Meta-component systems
- Component systems
- DSL and code generation
- RTSJ
Interoperable DSL Families for Code Generation

Extensible Polyglot Programming Support in Existing Component Frameworks

Introducing Distribution into a RTSJ-based Component Framework

Constructing Domain-Specific Component Frameworks through Architecture Refinement

Using a Product Line for Creating Component Systems

Using DSL for Automatic Generation of Software Connectors

jPapabench (Exhaustive testing of safety critical Java.)

Q-ImPrESS
The Plan

- Identify driving idea
- Research questions & objectives
- Domain analysis
- Solution
- Evaluation
The Plan

- Identify driving idea
- Research questions & objectives
- Domain analysis
- Solution
- Evaluation

Using a Product Line for Creating Component Systems
Recap: Meta-component system

Classical view of CBSE development

Using a Product Line for Creating Component Systems
Recap: Meta-component system

Using a Product Line for Creating Component Systems

Meta-component system idea

- Configuration tool
  - Problem space: describes all possible combinations of variabilities and commonalities
  - Solution space: formed by core assets
- Component system
  - Model
  - Development & design tools
  - Deployment tools
  - Execution environment

- Application design and development
- Application deployment
- Application monitoring and managing

Application specification
Analysis of application requirements
Target platform properties
(Research/Technical) Questions

- How to derive component system entities?
  - Meta-model
  - Execution environment
  - Tools

- What are main constituents of meta-component system?

- How to achieve interoperability among the constituents?
Construction of execution environment

Models interoperability
How to deal with missing parts...

- **Meta-model preparation**
  - Rich meta-model (e.g., SOFA, AADL) adjusted via model aspects

- **Tools preparation**
  - Open question ;-)  
  - Eclipse platform can help
Thesis Objectives

Construction of execution environment

WHERE TO BEGIN?

Models interoperability
The Plan

- Identify driving idea
- Research questions & objectives
- Domain analysis
- Solution
- Evaluation

Introducing Distribution into a RTSJ-based Component Framework

jPapabench (Exhaustive Testing of Safety Critical Java.)

Extensible Polyglot Programming Support in Existing Component Frameworks

Q-ImPrESS
Domain Analysis

- Each of the paper utilizes a kind of component system
- Helps to identify the major concepts & features used by case-studies

- Simple light-weight form of “components”
- Different forms of execution
  - Model interpretation, compilation, hybrids
- Configurability
  - Runtime extension via aspects
- Configurable target technology
  - plain Java, RTSJ, SCJ
- Various interconnected models
  - Structure, model of code
- Code generation
Already presented on the seminar

Recap

- Simplified (highly) extensible flat component model to construct runtime environments
- No target technology explicitly assumed
  - Requires technology mapping (OSGi, Java)
- Variability configured via aspects

- Implementation still in progress :-/
Code generation is an inherent part of uSOFA

Typical example is an interceptor
Elements preparation in component-based systems

- Interceptors
- Connectors

Every element has defined structure (e.g., ADL) and a model of code

Model of code requires information which is unknown during design time of element

- E.g., signature of intercepted interface, number of connected clients
Models Interoperability

- Models are defined via a **domain-specific language**
  - Language is defined by a **grammar**

- To describe various system’s models, used languages has to somehow interact
  - E.g., a model of code needs somehow to refer a model of structure

- Static v. dynamic interaction
Dynamic Interaction Approach

- Elements generation described via 3 models
  - **(Design time) Code pattern**
    - Target code template referencing information stored in Architecture and Context descriptors
  - **(Design time) Architecture description**
    - The structure of element
  - **(Runtime) Context description**
    - Derived from the deployment model of application
How does it work?

- **Code pattern is described by EPLang-***
  - Small DSL which is targeted to be incorporated into another language (e.g., Java => EPLang-J)
  - Provides simple expressions – *foreach, if, set, include* to manipulate with expressions of target language
  - Provides *language queries* supporting interoperability
    - Traversing a destination AST (or a generic model) and transporting desired information
    - Similar to OCL or Xpath
Queries

Side AST

Context specification
(in CDL-J)

\{ determined by `<NAVIGATE>` part: \texttt{ports.port} \}

\{ determined by `<EXTRACT>` part: \texttt{type} \}

\textit{AST sub-trees integration}

Target AST

Code pattern specification
(in EILang-J)

\{ query: `_${\texttt{ports.port}}`.$\texttt{type}_` \}

\{ result of the query \}
Evaluation and Case studies

- **uSOFA**
  - Simple watch example
  - Construction of OSGi, Guice application

- **Language interoperability**
  - EPLang and connector generation
What is still missing...

- **In text**
  - State of the art, Conclusion
  - Section about uSOFA

- **In prototype**
  - Finalize uSOFA implementation
Thank you for your attention