
Master’s Thesis Proposal

Thomas Kappler (kappler@ipd.uka.de)
Chair Software Design and Quality
Institute for Program Structures and Data Organization (IPD)
Faculty of Informatics, Universität Karlsruhe (TH)
Motivation

- **Problem**: Integrating legacy software components (e.g. EJB, Java Beans) into model-based performance predictions
- **Idea**: Automatic creation of performance models for such components via code analysis
- **Benefits**: Enabling predictions for systems composed of existing and non-existing components
- **Actions**: Design and Implementation of a tool for code analysis to extract performance-relevant data from source code
Palladio Component Model Ongoing Work

Source Code (Java, C#)
- PCM2Code
  - (Reiner, Steffen)

Usage Model (Domain Expert)
- SEFF Model (Component Dev.)
- Assembly Model (Software Architect)
- Resource Env. (Deployer)

Palladio Component Model Instance
- PCM Editor
  - SimuCom
    - (Steffen)

Multi-threaded Performance Results
- Concurrency Solver
  - (Jens)

Stochastic Process Algebra
- Basic Solver
  - (Viki, Jens, Heiko)

Single-threaded Performance Results
- Simulation
  - (Henning)

Benchmarking
- (Omri, Michael)

Concurrent Resource Model
- (Henning)

Probability Function Package
- (Ihssane, Jens)

Extended SPA (Scheduling)
- (Henning)

PCM Instance (Solved Dependencies)
- DSolver
  - (Heiko)

Tools/Libs
- <<uses>> PCM2SPA
  - (Wenyun)

Design Model
- Results
- Analytical Model
- Packages

Code2SEFF
- (Thomas, Heiko, Klaus)

Code2Arch
- (Klaus, Landry)
Performance Model for a component should include

- resource usage
  (i.e., network and hard disk access, using CPUs)
- abstract control flow
  (i.e., which external services will be invoked when calling a provided service)
- influence of parameter values

Enables adjusting model parameters during development.
**Palladio** component model: resource usage in the control flow modelled with *(Resource-demanding) Service Effect Specifications*

- **SEFF** For a provided service: which required services does it call, and when

- **RDSEFF** Also models
  - internal resource usage
  - the relationship between external service calls and their resource demands
  - the influence of input parameters
public void
storeMyObject (MyObject myObject) {
    if ( myObject.condition() )
        extService.store(myObject);
}
// an external call
int test = ExternalComponent.addOne(argument1);
foo.substring(1);

// external call nested in a condition
if (argument1==1) {
    int test2 = ExternalComponent.addOne(argument1);
    if (argument2==2) {
        int test4 = ExternalComponent.addOne(argument2);
    }
} else {
    if (argument2==3) {
        int test3 = ExternalComponent.addOne(argument2);
    }
    while (i<5) {
        foo.substring(1);
    }
}

// a while loop
while (i<5) {
    foo.substring(1);
}

// a for loop
for (int j=0; j<10; j++) {
    foo.substring(1);
}

// a for loop
for (int j=0; j<10; j++) {
    for (int k=0; k<10; k++) {
        foo.substring(1);
    }
}
Tasks

1. Determine the provides and requires interfaces
2. Control flow analysis: loops, branches, OO structures, threads, ...
3. Identify resource demanding calls (other components, Java API)
4. Determine the influence of parameters
5. Evaluation
Tools

- Eclipse
- Java Development Tools (JDT)
- JDT Abstract Syntax Tree parser
- JUnit unit tests
- Eclipse Modelling Framework (creating performance model instances of PCM)
- Visualization of SEFF with generated graphical GEF-Editors
- Subversion, Latex
Starting in Karlsruhe, with

- Professor Dr. **Ralf Reussner**
- Dipl.-Inform. **Heiko Koziolek**

February 25: Transition to Delhi, India. Supervision by Professor **Pankaj Jalote** at the IIT Delhi.
Conclusions

▪ Goal: Derive abstract component performance models from source code
  – Consider external service calls and resource demanding API calls
▪ Enable predictions for systems composed of existing and non-existing components
▪ Implementation based on Eclipse