Performance Certification of Software Components

FESCA 2011

Erik Burger, Ralf Reussner | 2 April 2011
Motivation

Performance Requirements

Component Specification

The Palladio Component Model Certification

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011 2/25
Performance
Requirements
Component
Specification
Component
Implementation
Motivation

Component Specification

Performance Requirements

Performance Description

Component Implementation

The Palladio Component Model

Certification

Conclusion

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011
Motivation

Performance
Requirements
Component
Specification

Performance
Description
Component
Implementation

reverse
engineering

CERTIFIED
holds
successful
conforms to

[Groenda, 2009]
[Krogmann et al., 2010]
Motivation

Component Specification

Performance Requirements

Component Implementation

Performance Description

test-based validation
reverse engineering

refinement? [Groenda, 2009] [Krogmann et al., 2010]

CERTIFIED holds successful conforms to
Motivation

Performance Requirements

Component Specification

Performance Description

refinement?

test-based validation
reverse engineering

Component Implementation

The Palladio Component Model

Certification

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011 2/25
Motivation

Component Specification

Performance Requirements

Performance Description

refinement?

holds

test-based validation

reverse engineering

Component Implementation

CERTIFIED

holds

[Groenda, 2009]

[Krogmann et al., 2010]
Motivation

Component Specification

Performance
Requirements

Performance
Description

refinement?

holds

successful

test-based validation

reverse engineering

Component Implementation

CERTIFIED

holds

successful

[Groenda, 2009]

[Krogmann et al., 2010]
Motivation

Component Specification

Performance Requirements

Performance Description

refinement?

holds

successful

CERTIFIED

Component Implementation

test-based validation

reverse engineering

CERTIFIED

[Groenda, 2009]

[Krogmann et al., 2010]
**Motivation**

- **Performance Requirements**
- **Component Specification**
- **Performance Description**
- **Component Implementation**

- CERTIFIED holds successful
- test-based validation
- reverse engineering

- conforms to refinement?

---

The Palladio Component Model

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011
Motivation

- Performance
- Requirements
- Component Specification

Component Implementation

Performance Description

refinement?

[Groenda, 2009]

test-based validation

reverse engineering

CERTIFIED

holds

successful

conforms to

conforms to

The Palladio Component Model

Certification

Conclusion

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011 2/25
Motivation

Performance Requirements
Component Specification

[Krogmann et al., 2010]
refinement?
holds
successful
conforms to
CERTIFIED
Component Implementation

Performance Description

test-based validation
reverse engineering

The Palladio Component Model

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011

2/25
The Palladio Component Model

CERTIFIED

holds

successful

does refinement hold?

conforms to

Performance Requirements

Component Specification

Performance Description

Component Implementation

test-based validation

reverse engineering

[Groenda, 2009]

[Krogmann et al., 2010]
Outline

1. The Palladio Component Model
   - Definition
   - Example

2. Certification
   - Refinement
   - Example

3. Conclusion
Performance Models

Analysis-oriented (predictive) performance models

- Queueing Networks (Layered Queueing Networks)
- Queueing Petri Nets
- Stochastic Process Algebra
- ...

Design-oriented performance models

- KLAPER, UML SPT, UML MARTE
- Palladio Component Model (PCM)
- ...

The Palladio Component Model

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011
Performance Models

Analysis-oriented (predictive) performance models
- Queueing Networks (Layered Queueing Networks)
- Queueing Petri Nets
- Stochastic Process Algebra
- ...

Design-oriented performance models
- KLAPER, UML SPT, UML MARTE
- Palladio Component Model (PCM)
- ...

The Palladio Component Model

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011

4/25
Performance Prediction

Model
Performance model of a component-based software architecture

Performance Analysis
- Execution time
- Throughput
- Resource utilisation
Service Effect Specification (SEFF)

ComponentA

- a()
- b()
- c()
- d()

ComponentB

- b()
- c()

ComponentC

- d()
Service Effect Specification (SEFF)
Service Effect Specification (SEFF)

- SEFF abstracts the behaviour of a component
- Only performance-relevant attributes
- Different formalisms
  - counter-constraint automata
  - finite state machines
  - RDSEFF as defined in the Palladio Component Model
SEFF in the Palladio Metamodel

Resource Demanding SEFF (RDSEFF)

- Resource demands
  - CPU, memory, HDD, …
- External service calls
- Parameter dependencies
  - Resource demands
  - Loops and branches
- Variable values can be probabilistic
  - Stochastic Expressions
The Palladio Component Model

Certification

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011 9/25
The Palladio Component Model

Certification

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011
The Palladio Component Model

Certification

Erik Burger, Ralf Reussner – Performance Certification of Software Components
The Palladio Component Model Certification Conclusion

Erik Burger, Ralf Reussner – Performance Certification of Software Components

2 April 2011 9/25
Properties of RDSEFF

- structure: directed acyclic graph
- loops are not allowed

Instead, number of loop iterations can be specified with stochastic value

\[ \text{IntPMF}[(27;0.1)(28;0.2)(29;0.6)(30;0.1)] \]

\[ \ldots \]
Properties of RDSEFF

- structure: directed acyclic graph
- loops are not allowed

instead, number of loop iterations can be specified with stochastic value

IntPMF[(27;0.1)(28;0.2)(29;0.6)(30;0.1)]
Properties of RDSEFF

- structure: directed acyclic graph
- loops are not allowed
- instead, number of loop iterations can be specified with stochastic value

```
«ExternalCallAction»
  service1

«InternalAction»
  innerMethod

«Loopaction»
  loop
  IntPMF[(27;0.1)(28;0.2)(29;0.6)(30;0.1)]
  ...
```
Properties of RDSEFF

- structure: directed acyclic graph
- loops are not allowed

Instead, number of loop iterations can be specified with stochastic value

```
IntPMF[(27;0.1)(28;0.2)(29;0.6)(30;0.1)]
```
Outline

1. The Palladio Component Model
   - Definition
   - Example

2. Certification
   - Refinement
   - Example

3. Conclusion
Motivation

Performance Requirements

Component Specification

Performance Description

refinement?

conforms to

certified

holds

successful

test-based validation

reverse engineering
Refinement Hierarchy

External Calls

Acquire/Release

Resource Demands
Refinement Hierarchy

External Calls

Acquire/Release

Resource Demands
External Calls

**Description**
- externally visible behaviour of component
- interface protocols

**Refinement**
- neglect stochastic properties and parameters
- external calls and parameters as finite state machines
- use substitutibility as criterion from [Wehrheim & Reussner, 2006] for refinement
External Calls

Description

- externally visible behaviour of component
- interface protocols

Refinement

- neglect stochastic properties and parameters
- external calls and parameters as finite state machines
- use substitutibility as criterion from [Wehrheim & Reussner, 2006] for refinement
Transformation into FSM

- Start state: $q_0$
- Transition:
  - $q_0 \rightarrow q_1$: roleA.requiredService1
  - $q_1 \rightarrow q_2$: $\epsilon$
  - $q_2 \rightarrow q_3$: roleB.requiredService2
  - $q_1 \rightarrow q_3$: $\epsilon$
  - $q_3 \rightarrow q_0$: $\epsilon$
Refinement Hierarchy

External Calls

Acquire/Release

Resource Demands
Refinement of Resource Demands

Refinement Relation

A Service Effect Specification $SEFF_1$ is refined by $SEFF_2$ if the resource demands of $SEFF_1$ are greater or equal than those of $SEFF_2$.

$$SEFF_1 \geq_{RD} SEFF_2$$  \hspace{1cm} (1)$$

$SEFF_1$ can be structurally different from $SEFF_2$. 
Refinement of Resource Demands

Structural Refinement
- order of external calls
- replacement of internal actions
  - by loops/branches
  - by multiple internal actions

Stochastical Refinement
- comparison of stochastic functions
- definition of parameter ranges

\( \text{StoEx}_1 \rightarrow \text{StoEx}_2 \)
Refinement of Resource Demands

**Structural Refinement**
- order of external calls
- replacement of internal actions
  - by loops/branches
  - by multiple internal actions

**Stochastical Refinement**
- comparison of stochastic functions
- definition of parameter ranges

$StoEx_1 \rightarrow StoEx_2$
Refinement

Rule-Based refinement
- definition of refinement is *rule-based*
- constructive approach: if there is an application of rules, then the refinement relation holds

Encoding in Prolog
- usage of a rule-based language
- encoding of SEFFs in Prolog facts
- encoding of refinement rules
Refinement

Rule-Based refinement
- definition of refinement is *rule-based*
- constructive approach: if there is an application of rules, then the refinement relation holds

Encoding in Prolog
- usage of a rule-based language
- encoding of SEFFs in Prolog facts
- encoding of refinement rules

Skip Prolog expressions
Prolog Example

% checks if the demand of action A is greater
% or equal than those of the actions in a list
demand_geq(A,[Head|Tail]) :-
    demands(A,D1),
    demands(Head,D2),
    D1 >= D2,
    demand_geq(A,Tail).

demand_geq(A,[]).

% refinement of internalAction to branchAction
refinesIntBranch(I1,B1) :-
    intaction(I1),
    branchaction(B1),
    demands(I1,D1),
    branches(B1,L),
    demand_geq(I1,L).
Prolog Example

1 intaction(intaction1).
2 demands(intaction1,10).
3 branchaction(branchaction1).
4 intaction(action2).
5 intaction(action3).
6 demands(action2,8).
7 demands(action3,5).
8 branches(branchaction1,[action2, action3]).
Prolog Example

1  intaction(intaction1).
2  demands(intaction1,10).
3  branchaction(branchaction1).
4  intaction(action2).
5  intaction(action3).
6  demands(action2,8).
7  demands(action3,5).
8  branches(branchaction1,[action2, action3]).
Prolog Example

1 \textit{intaction(intaction1).}
2 \textit{demands(intaction1,10).}
3 \textit{branchaction(branchaction1).}
4 \textit{intaction(action2).}
5 \textit{intaction(action3).}
6 \textit{demands(action2,8).}
7 \textit{demands(action3,5).}
8 \textit{branches(branchaction1,[action2, action3]).}
Prolog Example

1  intaction(intaction1).
2  demands(intaction1, 10).
3  branchaction(branchaction1).
4  intaction(action2).
5  intaction(action3).
6  demands(action2, 8).
7  demands(action3, 5).
8  branches(branchaction1, [action2, action3]).

1  ?- refinesIntBranch(X, Y).
2  X = intaction1,
3  Y = branchaction1 ?
4  yes
Outline

1 The Palladio Component Model
   • Definition
   • Example

2 Certification
   • Refinement
   • Example

3 Conclusion
Future Work

Comparability of Stochastical Expressions
- extend “greater than” relation to stochastical functions
- restricting the comparison to an interval

Refinement Rules
- rules for the matching of all elements
- ordering of actions
- preconditions for refinement
  (e.g. certain type of stochastical function)
- proof of correctness for rules
Conclusion

Contribution

- hierarchy of refinement notions for performance certification based on the *Palladio Component Model*
- compatibility of external actions expressed in terms of finite state machines

Thank you for your attention!
References I


# SEFF constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StartAction/StopAction</strong></td>
<td>Start and stop of the control flow</td>
</tr>
<tr>
<td><strong>InternalAction with Resource Demands</strong></td>
<td>Abstraction of the internal logic and the resulting resource demands</td>
</tr>
<tr>
<td><strong>ExternalCallAction</strong></td>
<td>Use a service provided by another component</td>
</tr>
<tr>
<td></td>
<td>Includes variable usage and variable characterisation for definition and value assignment</td>
</tr>
<tr>
<td><strong>SetVariableAction</strong></td>
<td>Set and change of variables</td>
</tr>
<tr>
<td></td>
<td>Assignment of parameter values is done analogously to ExternalCallAction</td>
</tr>
<tr>
<td><strong>AcquireAction/ReleaseAction</strong></td>
<td>Acquire and release passive resources</td>
</tr>
<tr>
<td></td>
<td>(e.g. a semaphore)</td>
</tr>
<tr>
<td>SEFF constructs</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td><strong>Loops</strong></td>
<td>LoopActions are evaluated stochastically independent wrt. to contained parametric dependencies</td>
</tr>
<tr>
<td></td>
<td>CollectionIteratorActions are evaluated stochastically dependent wrt. to the characterisation of the parameter being iterated</td>
</tr>
<tr>
<td><strong>BranchAction</strong></td>
<td>Probabilistic-based (Probabilistic Branch Transition)</td>
</tr>
<tr>
<td></td>
<td>Value-based (Guarded Branch Transition)</td>
</tr>
<tr>
<td><strong>ForkAction</strong></td>
<td>Parallel and asynchronous execution</td>
</tr>
<tr>
<td></td>
<td>Synchronous execution is possible using a Synchronisation Point</td>
</tr>
</tbody>
</table>