Towards Formal Certification of Software Components

Workshop on Component-Oriented Programming (WCOP) 2010

Erik Burger | 22nd June 2010
Motivation: Component Certification

Component Specification

Component Implementation
Motivation: Component Certification

Component Specification

Component Implementation

Motivation

Foundations

Approach

Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Motivation: Component Certification

Performance Requirements

Component Specification

Performance Description

Component Implementation

0101101011001101
1101011001101001
0110110111100101
1110010011010110
0101101011001101
1101011001101001
0110110111100101
1110010011010110

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Motivation: Component Certification

- Performance Requirements
  - Component Specification

- Performance Description

Component Implementation

010101011001101
1101011001101001
0110110111100101
1110010011010110

0101101011001101
1101011001101001
0110110111100101
1110010011010110

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Motivation: Component Certification

Performance

Requirements

Component
Specification

Performance
Description

test-based validation
reverse engineering

Component
Implementation

Motivation

Foundations

Approach

Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Motivation: Component Certification

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

- test-based validation
- reverse engineering

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Motivation: Component Certification

- Performance Requirements
  - Component Specification

Performance Description
  - test-based validation
  - reverse engineering

Component Implementation

Motivation
- Erik Burger – Towards Formal Certification of Software Components
Motivation: Component Certification

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

refinement

test-based validation
reverse engineering
Motivation: Component Certification

- Motivation
- Foundations
- Approach
- Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Motivation: Component Certification

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

refinement

test-based validation

reverse engineering

CERTIFIED

Motivation
- Foundations
- Approach
- Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Service Effect Specification (SEFF)

- SEFF abstracts the behaviour of a component
- Only performance-relevant attributes
- Control Flow
- Different formalisms
  - counter-constraint automata
  - finite state machines
  - Palladio metamodel
«ExternalCallAction»
requiredService1

«InternalAction»
innerMethod
ResourceDemand
1000 <CPU>

«BranchAction»
?
branch
Probability: 0.6

«Loopaction»
loop
array.NUMBER_OF_ELEMENTS

«ExternalCallAction»
requiredService2
Probability: 0.4

«ExternalCallAction»
requiredService3
requiredService1

innerMethod

ResourceDemand

1000 <CPU>

branch

Probability: 0.6

Probability: 0.4

Motivation

Foundations

Approach

Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Motivation

Foundations

Approach

Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
«ExternalCallAction»
requiredService1

«InternalAction»
innerMethod

ResourceDemand
1000 <CPU>

«BranchAction»
branch

Probability: 0.6

«Loopaction»
loop
array.NUMBER_OF_ELEMENTS

Motivation

Foundations
Erik Burger – Towards Formal Certification of Software Components

Approach

Conclusion
22nd June 2010
«ExternalCallAction»
requiredService1

«InternalAction»
innerMethod

ResourceDemand
1000 <CPU>

«BranchAction»
branch

Probability: 0.6

«Loopaction»
loop
array.NUMBER_OF_ELEMENTS

«ExternalCallAction»
requiredService2
«ExternalCallAction»
requiredService1

ResourceDemand
1000 <CPU>

«InternalAction»
innerMethod

«BranchAction»
branch

Probability: 0.6

«Loopaction»
loop
array.NUMBER_OF_ELEMENTS

«ExternalCallAction»
requiredService2

Probability: 0.4
Properties of SEFF

- 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]
```

Motivation

Foundations

Approach

Conclusion
Properties of SEFF

- structure: directed acyclic graph
- loops are not allowed

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

Motivation

Foundations

Approach

Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
Properties of SEFF

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

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

Motivation  Foundations  Approach  Conclusion

Erik Burger – Towards Formal Certification of Software Components

22nd June 2010  5/11
Certification Statement

If $SEFF_2$ is a refinement of $SEFF_1$, the component annotated with $SEFF_2$ fulfills the performance requirements which are expressed by $SEFF_1$.

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 $SEFF_2$.

$$SEFF_1 \geq_{RD} SEFF_2 \quad (1)$$

$SEFF_1$ can be structurally different from $SEFF_2$. 
Certification Statement

If $SEFF_2$ is a refinement of $SEFF_1$, the component annotated with $SEFF_2$ fulfills the performance requirements which are expressed by $SEFF_1$.

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 $SEFF_2$.

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

$SEFF_1$ can be structurally different from $SEFF_2$. 
Refinement Relation for Certification

Certification Statement

If $SEFF_2$ is a refinement of $SEFF_1$, the component annotated with $SEFF_2$ fulfills the performance requirements which are expressed by $SEFF_1$.

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 $SEFF_2$.

$$SEFF_1 \geq_{RD} SEFF_2 \quad (1)$$

$SEFF_1$ can be structurally different from $SEFF_2$. 
Refinement

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

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
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]).
intaction(intaction1).
demands(intaction1,10).
branchaction(branchaction1).
intaction(action2).
intaction(action3).
demands(action2,8).
demands(action3,5).
branches(branchaction1,[action2, action3]).

Motivation
Foundations
Approach
Conclusion
Erik Burger – Towards Formal Certification of Software Components

22nd June 2010
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]).

?- refinesIntBranch(X,Y).
  X = intaction1,
  Y = branchaction1 ?
  yes
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

Thank you for your attention!
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
  - StoEx-Editor
<table>
<thead>
<tr>
<th>SEFF constructs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StartAction/</strong></td>
<td><strong>StopAction</strong></td>
</tr>
<tr>
<td><strong>InternalAction</strong></td>
<td><strong>with Resource Demands</strong></td>
</tr>
<tr>
<td><strong>ExternalCallAction</strong></td>
<td></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></td>
</tr>
<tr>
<td></td>
<td>Assignment of parameter values is done analogously to ExternalCallAction</td>
</tr>
<tr>
<td><strong>AcquireAction/</strong></td>
<td><strong>ReleaseAction</strong></td>
</tr>
</tbody>
</table>
SEFF constructs

<table>
<thead>
<tr>
<th>Loops</th>
<th>LoopActions are evaluated stochastically independent wrt. to contained parametric dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CollectionIteratorActions are evaluated stochastically dependent wrt. to the characterisation of the parameter being iterated</td>
</tr>
<tr>
<td>BranchAction</td>
<td>Probabilistic-based (Probabilistic Branch Transition)</td>
</tr>
<tr>
<td></td>
<td>Value-based (Guarded Branch Transition)</td>
</tr>
<tr>
<td>ForkAction</td>
<td>Parallel and asynchronous execution</td>
</tr>
<tr>
<td></td>
<td>Synchronous execution is possible using a Synchronisation Point</td>
</tr>
</tbody>
</table>