Karlsruhe Institute of Technology

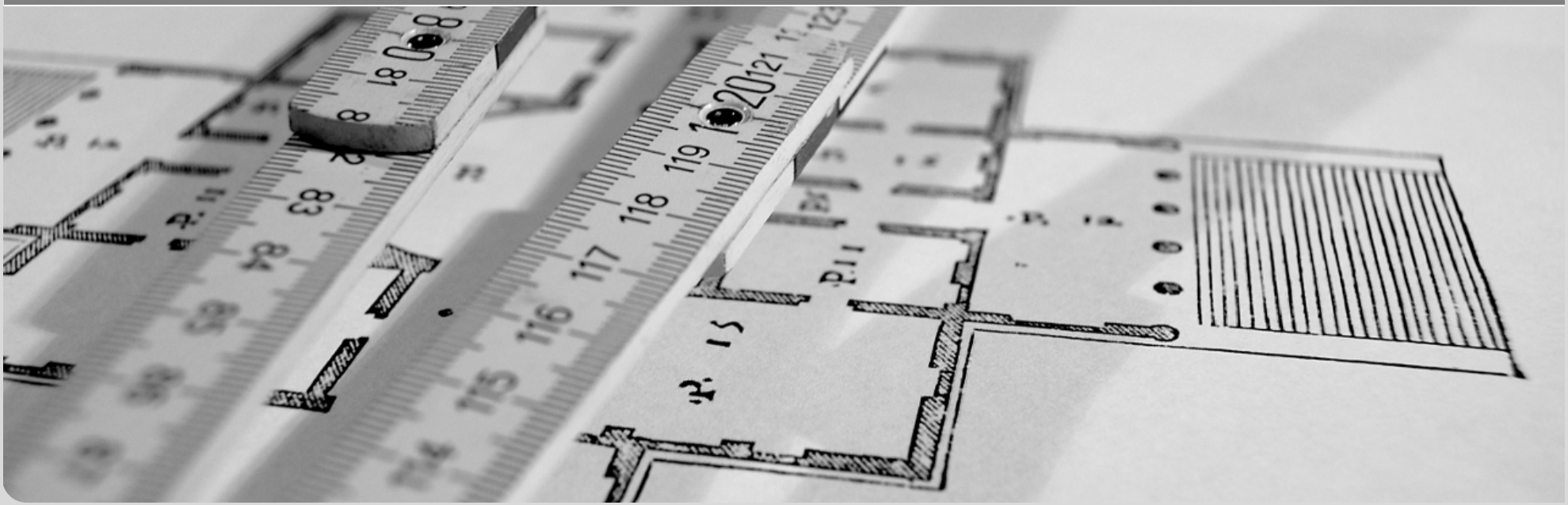
Software Design and Quality

Towards Formal Certification of Software Components

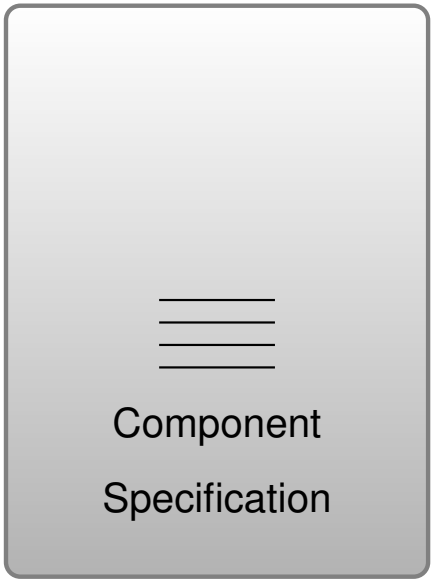
Workshop on Component-Oriented Programming (WCOP) 2010

Erik Burger | 22nd June 2010

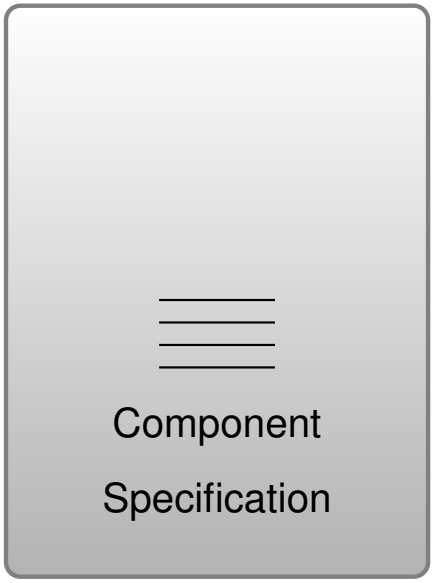
CHAIR FOR SOFTWARE DESIGN AND QUALITY

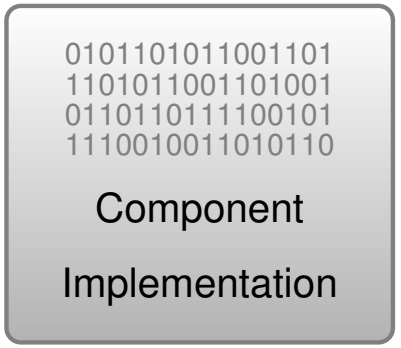


Motivation: Component Certification

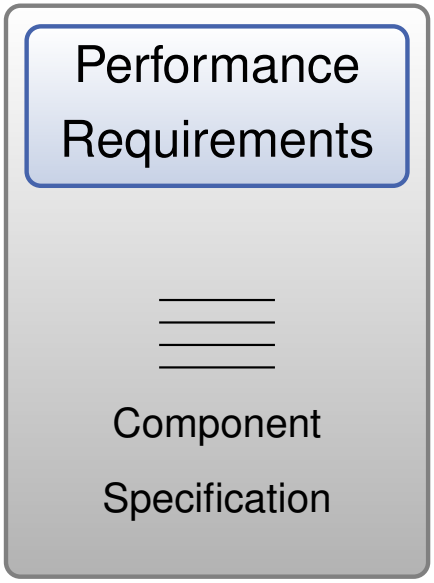


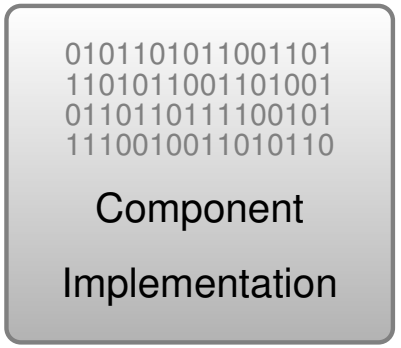
Motivation: Component Certification





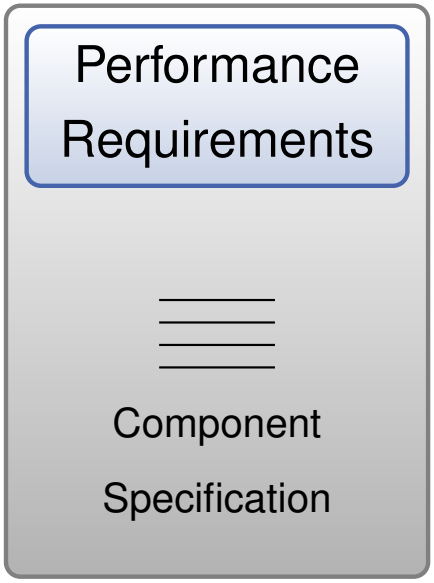
Motivation: Component Certification

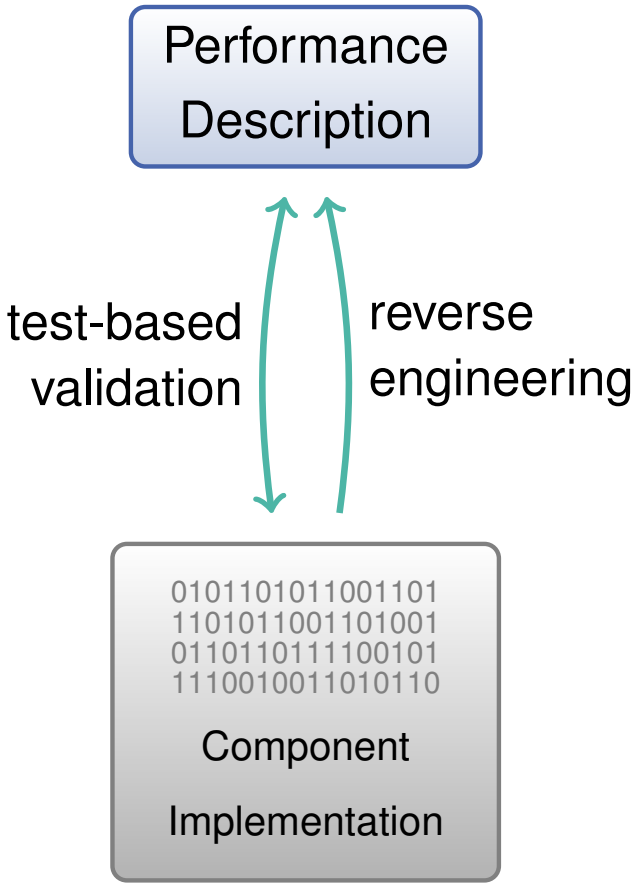




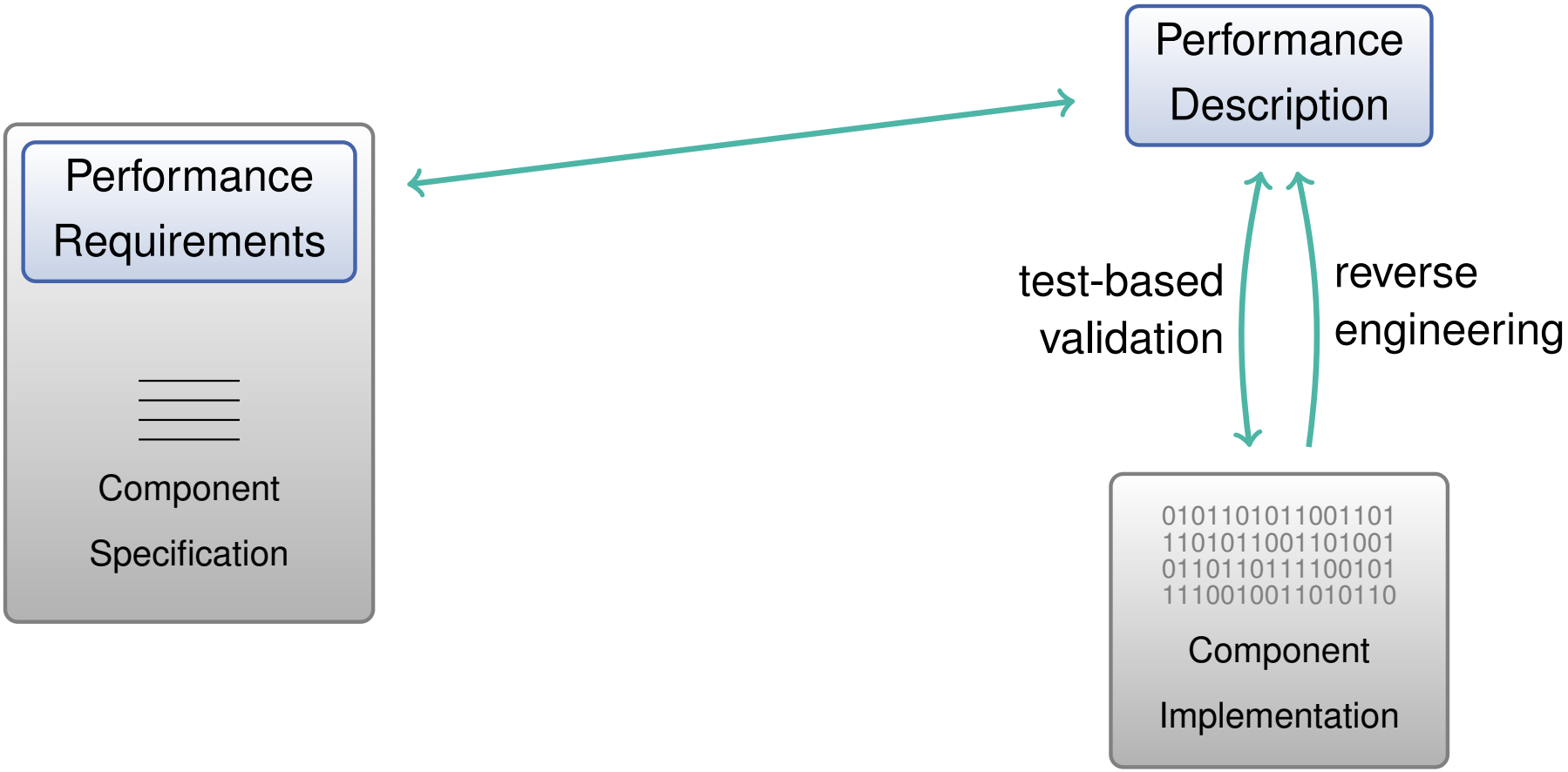
Motivation: Component Certification

Motivation: Component Certification

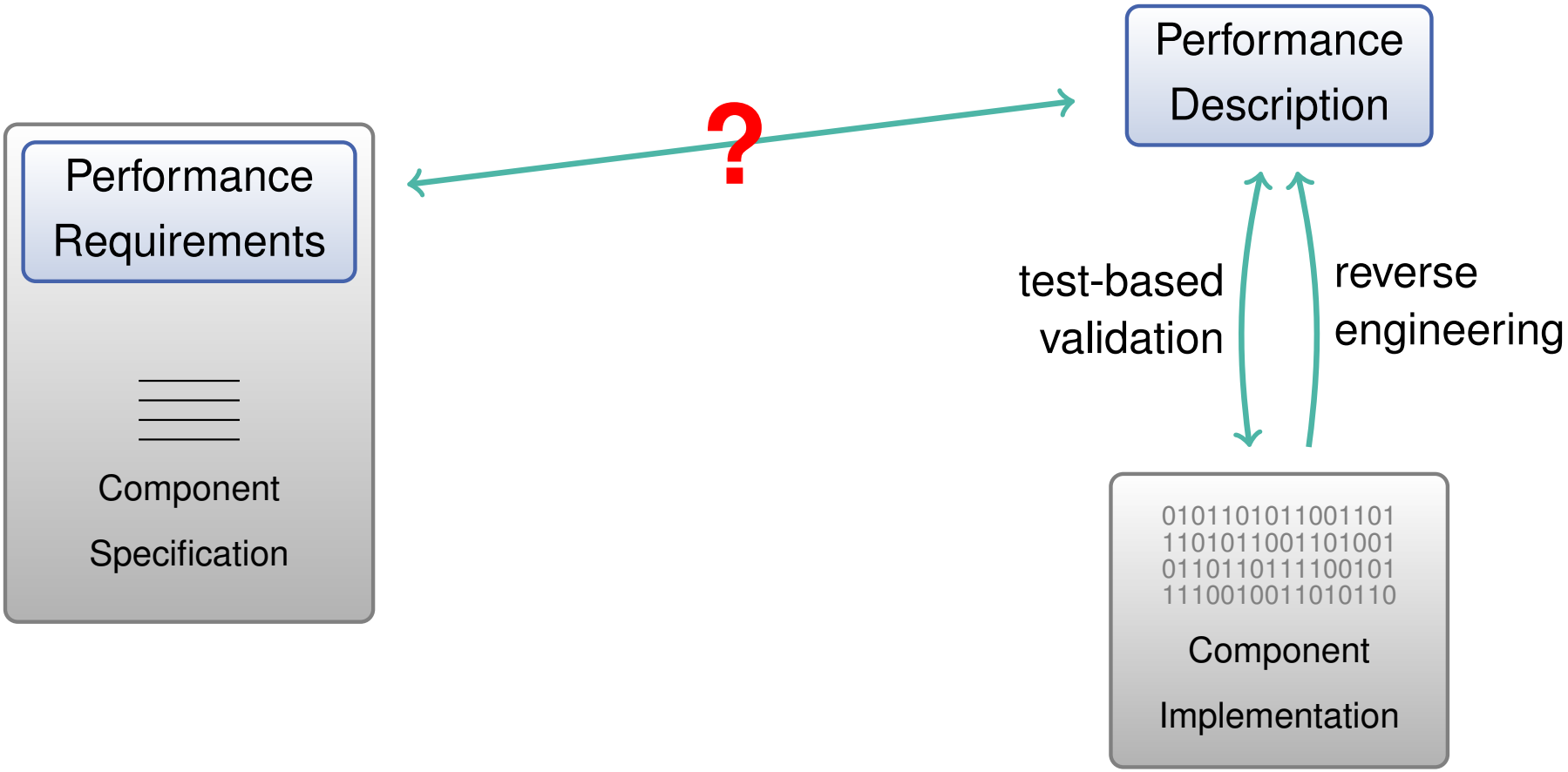




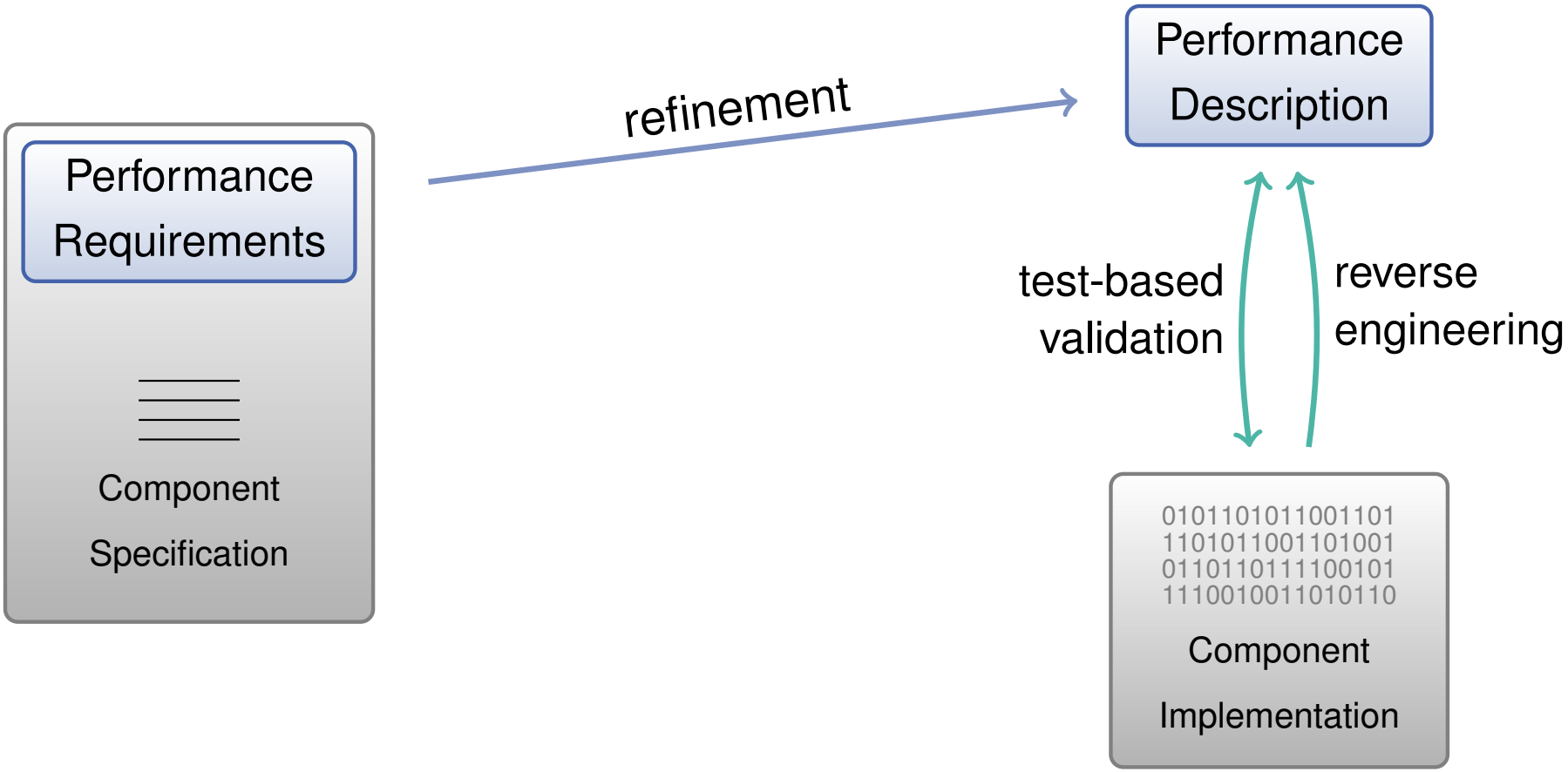
Motivation: Component Certification



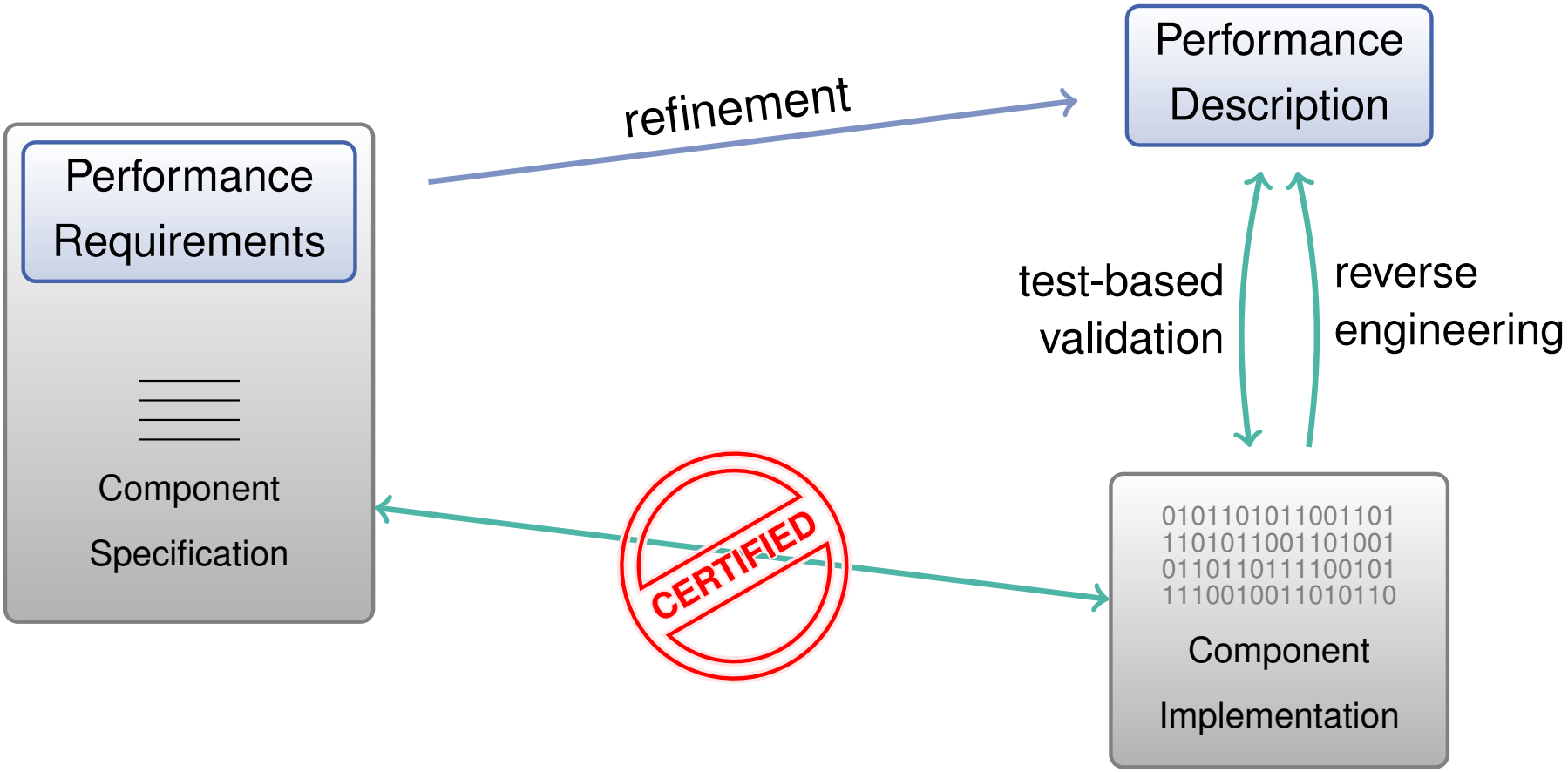
Motivation: Component Certification



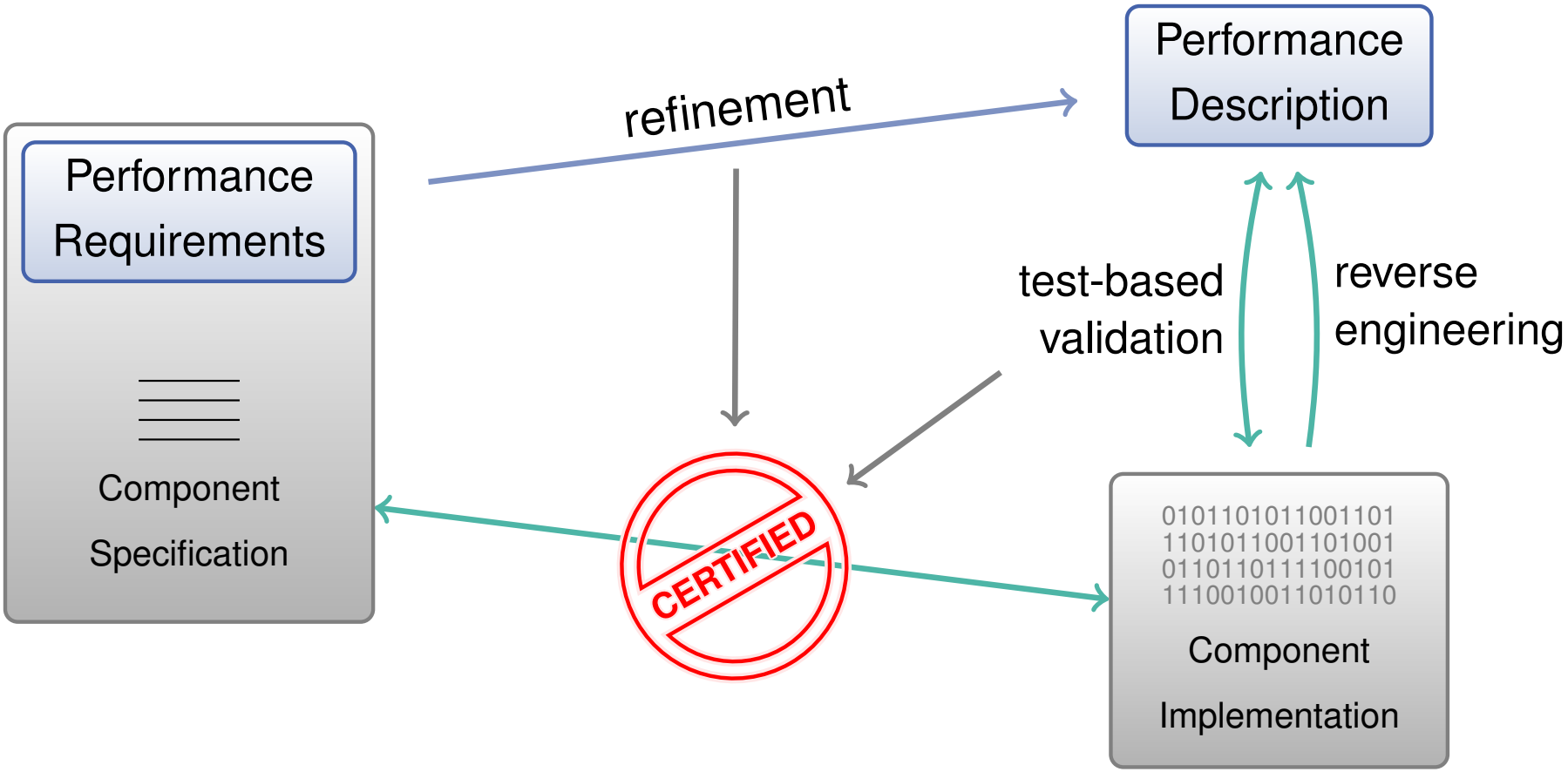
Motivation: Component Certification



Motivation: Component Certification



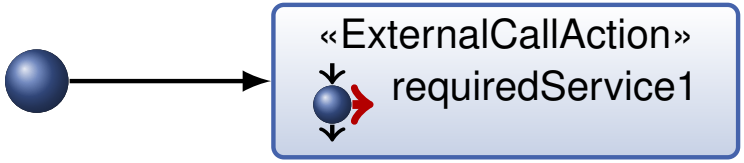
Motivation: Component Certification

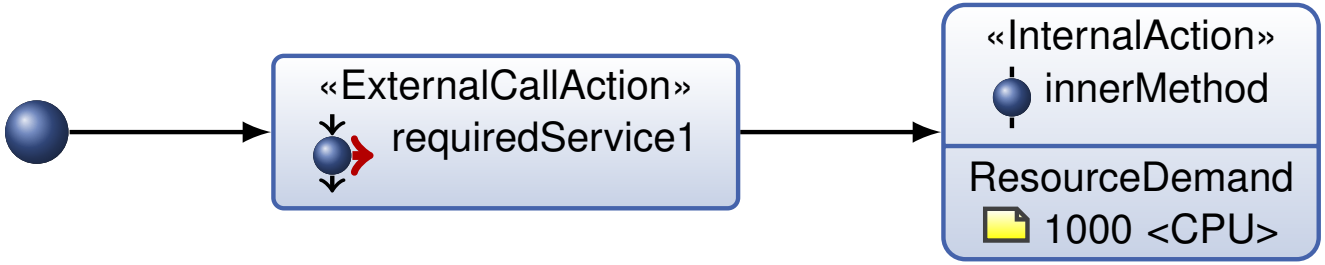


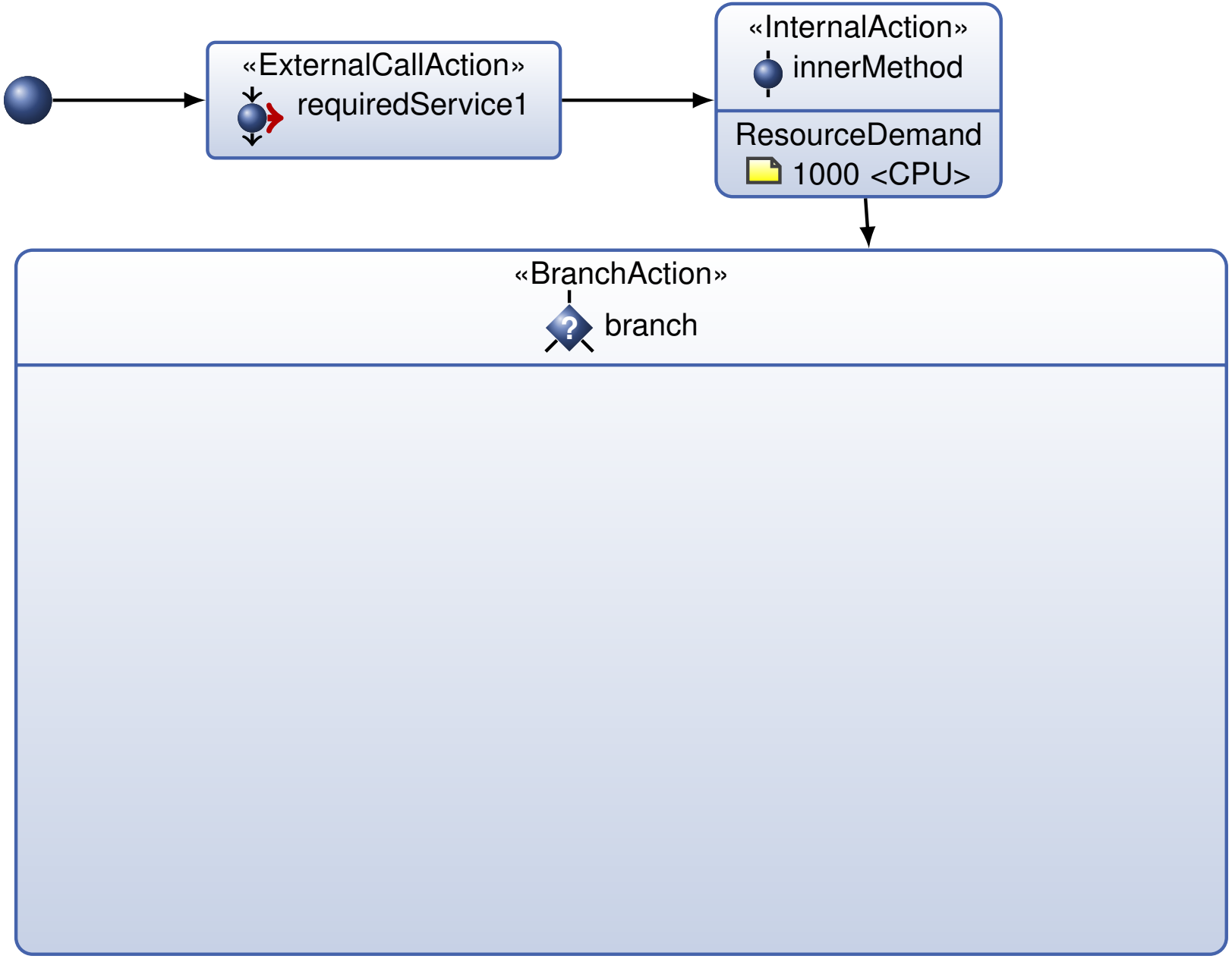
Performance Modeling with Palladio

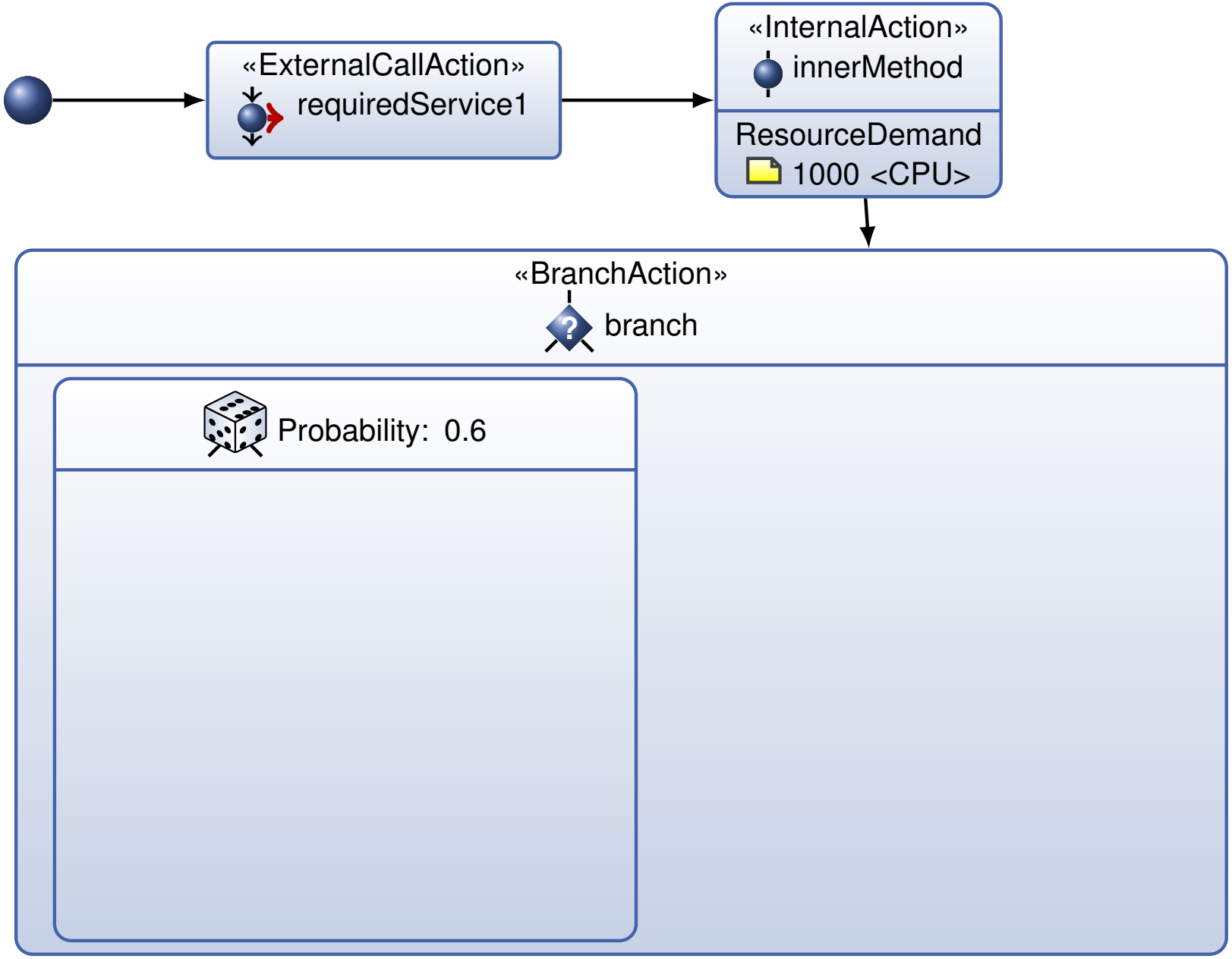
Service Effect Specification (SEFF)

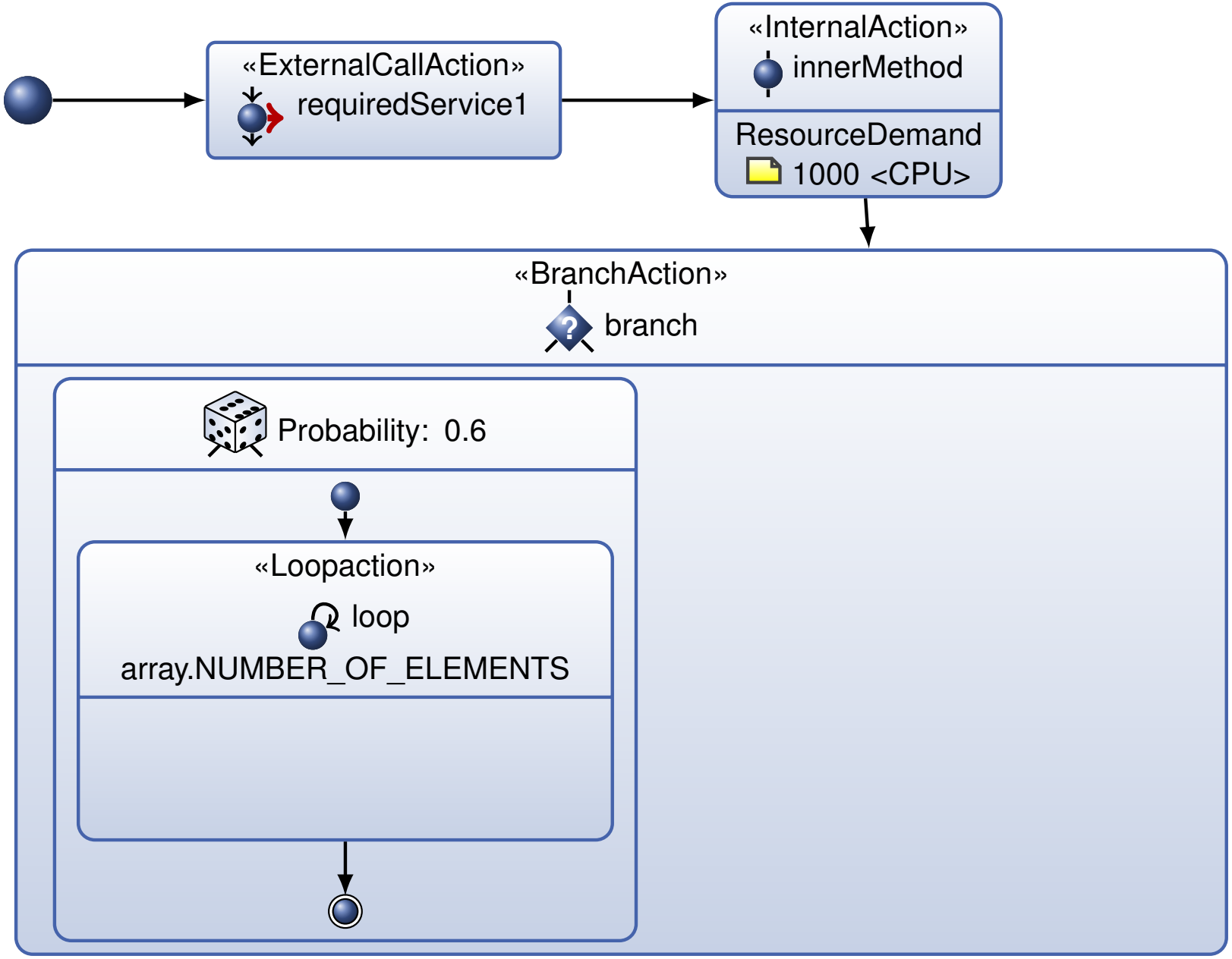
- SEFF abstracts the behaviour of a component
- Only performance-relevant attributes
- Control Flow
- Different formalisms
  - counter-constraint automata
  - finite state machines
  - *Palladio* metamodel

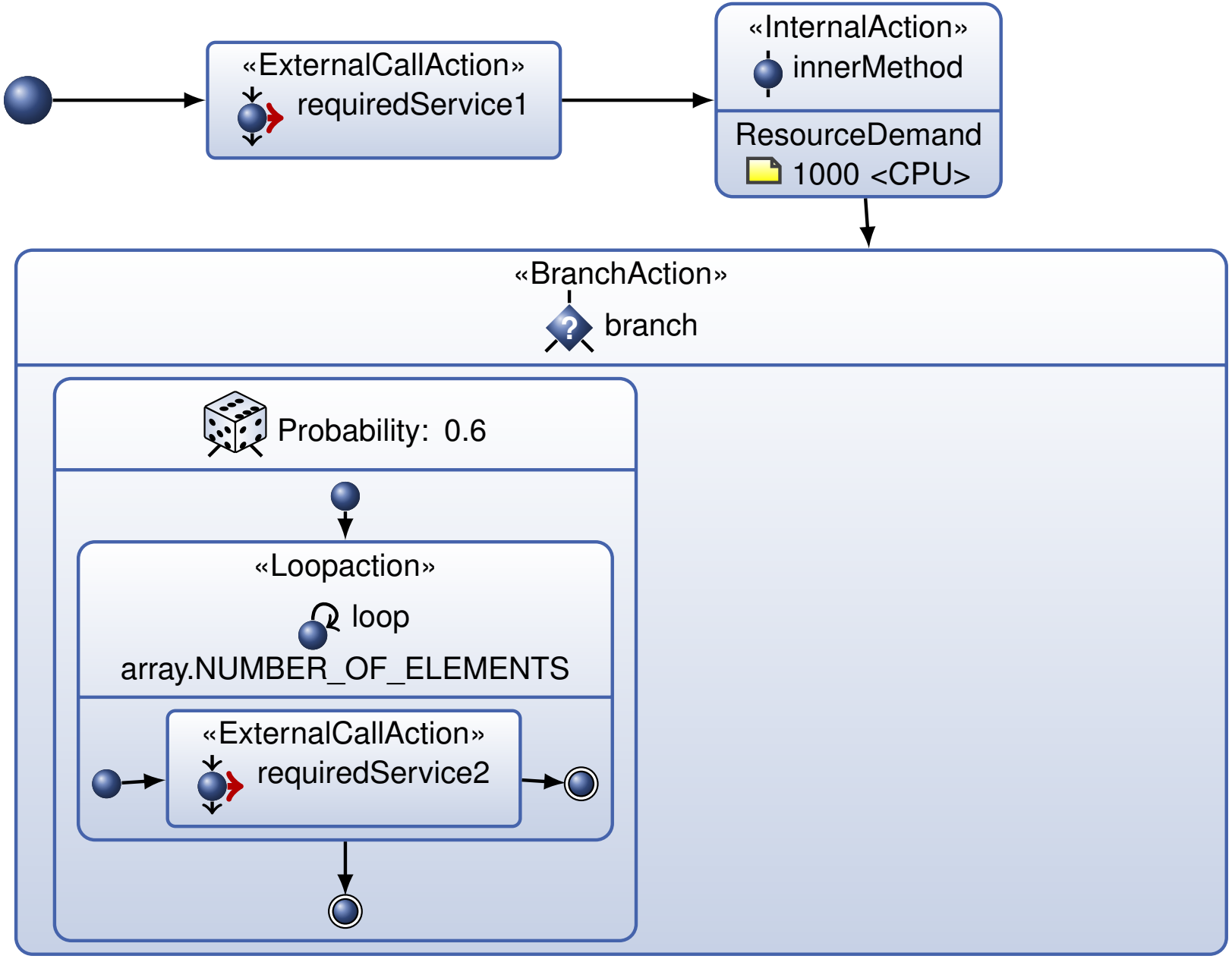

«ExternalCallAction»
requiredService1

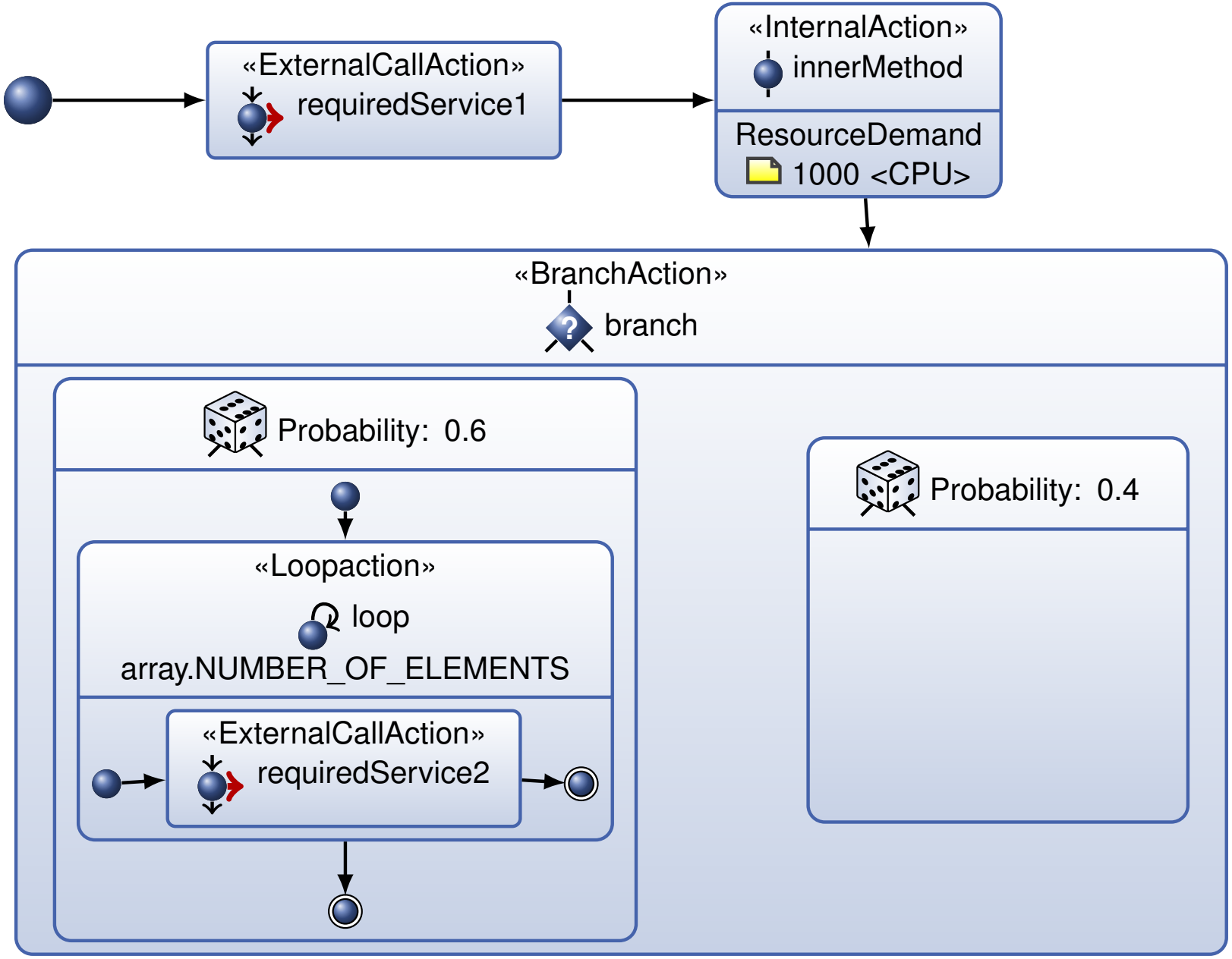

«ExternalCallAction»
requiredService1
«InternalAction»
innerMethod
ResourceDemand
1000 <CPU>

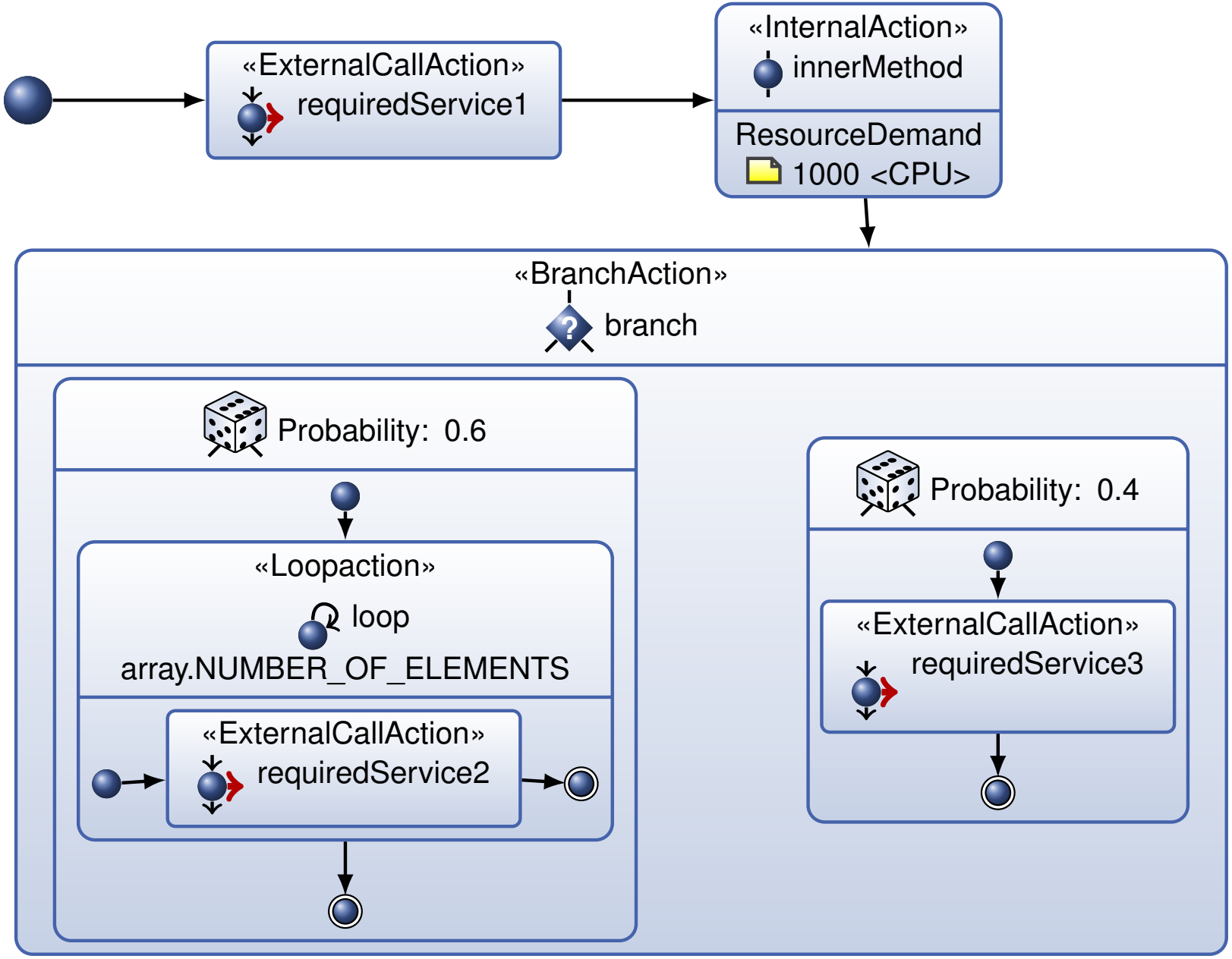

«ExternalCallAction»
requiredService1
«InternalAction»
innerMethod
ResourceDemand
1000 <CPU>
«BranchAction»
branch

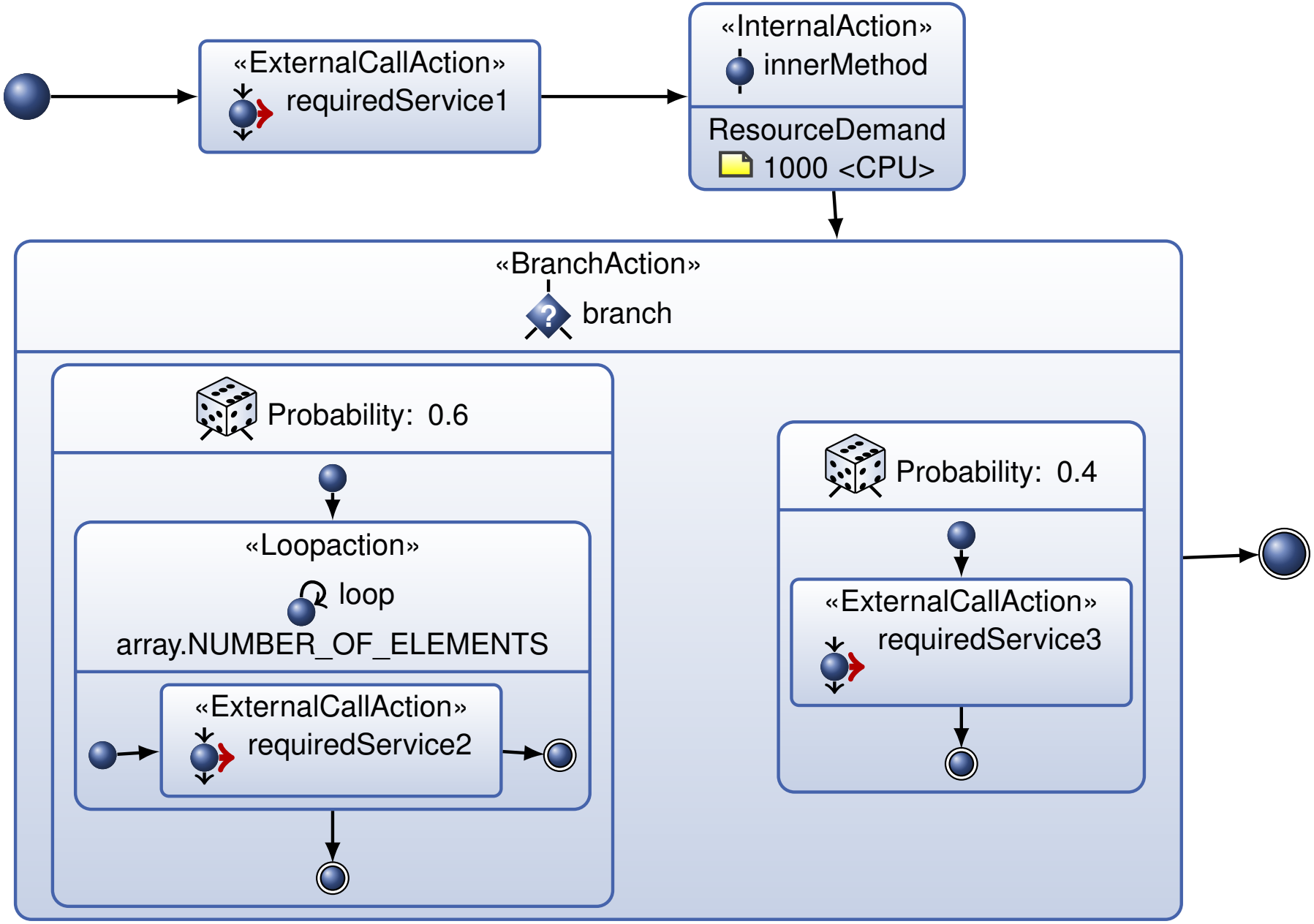

«ExternalCallAction»
requiredService1
«InternalAction»
innerMethod
ResourceDemand
1000 <CPU>
«BranchAction»
branch
Probability: 0.6


«ExternalCallAction»
requiredService1
«InternalAction»
innerMethod
ResourceDemand
1000 <CPU>
«BranchAction»
branch
Probability: 0.6
«Loopaction»
loop
array.NUMBER_OF_ELEMENTS


«ExternalCallAction»
requiredService1
«InternalAction»
innerMethod
ResourceDemand
1000 <CPU>
«BranchAction»
branch
Probability: 0.6
«Loopaction»
loop
array.NUMBER_OF_ELEMENTS
«ExternalCallAction»
requiredService2


«ExternalCallAction»
requiredService1
«InternalAction»
innerMethod
ResourceDemand
1000 <CPU>
«BranchAction»
branch
Probability: 0.6
«Loopaction»
loop
array.NUMBER_OF_ELEMENTS
«ExternalCallAction»
requiredService2
Probability: 0.4


«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


«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

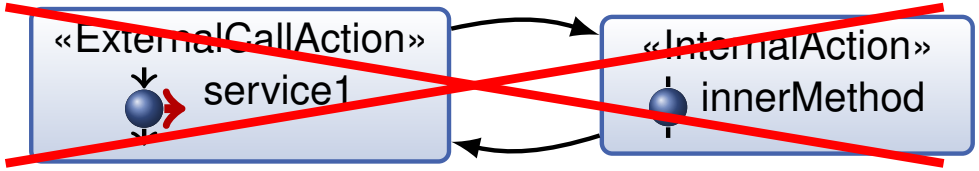
Properties of SEFF

- structure: directed acyclic graph
- loops are not allowed

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

Properties of SEFF

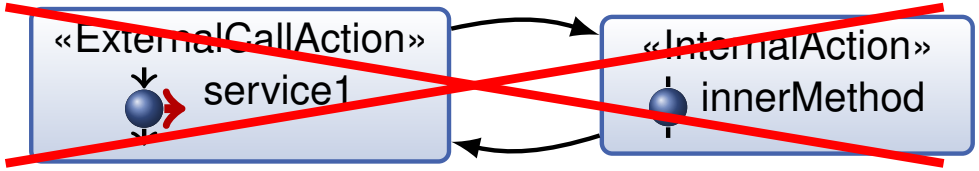
- structure: directed acyclic graph
- loops are not allowed



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

Properties of SEFF

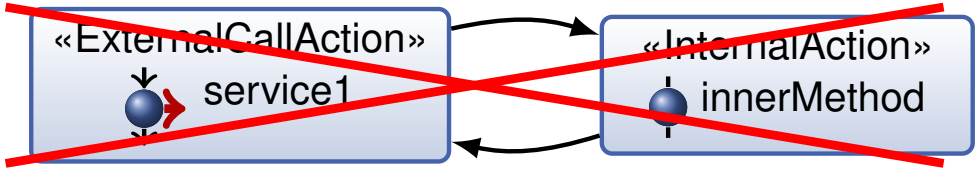
- structure: directed acyclic graph
- loops are not allowed



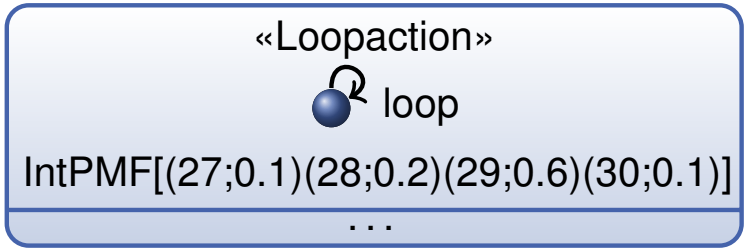
- instead, number of loop iterations can be specified with stochastic value

Properties of SEFF

- structure: directed acyclic graph
- loops are not allowed



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



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 $SEFF_2$.

$$SEFF_1 \geq_{RD} SEFF_2 \tag{1}$$

$SEFF_1$ 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 $SEFF_2$.

$$SEFF_1 \geq_{RD} SEFF_2 \tag{1}$$

$SEFF_1$ 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 $SEFF_2$.

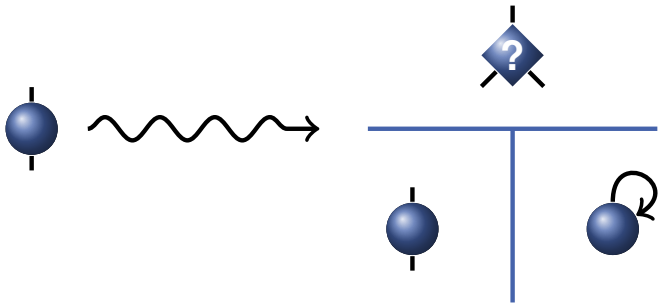
$$SEFF_1 \geq_{RD} SEFF_2 \tag{1}$$

$SEFF_1$ 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

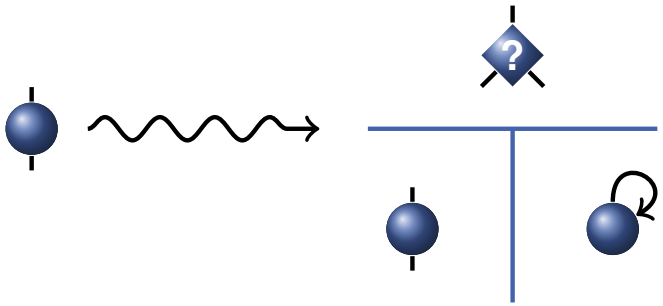
- comparison of stochastic functions
- definition of parameter ranges

$StoEx_1 \rightarrow StoEx_2$

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

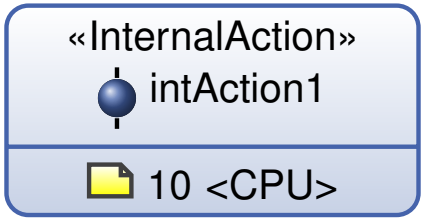
```
% checks if the demand of action A is greater
% or equal than those of the actions in a list
demand_geq(A,[Head|Tail]) :-
    demands(A,D1),
    demands(Head,D2),
    D1 >= D2,
    demand_geq(A,Tail).
demand_geq(A,[]).

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

Prolog Example

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

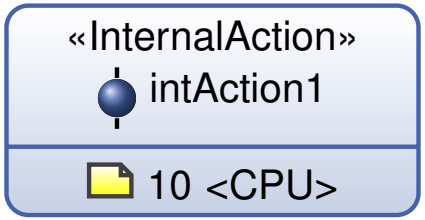
Prolog Example

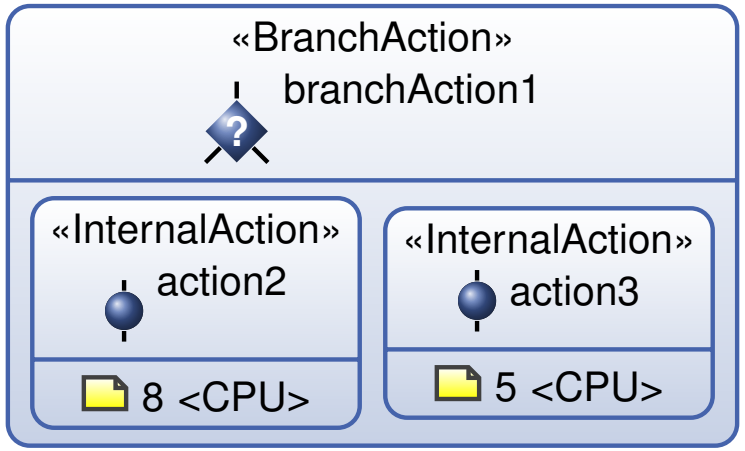
```
1  intaction(intaction1).
2  demands(intaction1,10).
3  branchaction(branchaction1).
4  intaction(action2).
5  intaction(action3).
6  demands(action2,8).
7  demands(action3,5).
8  branches(branchaction1,[action2,
       action3]).
```



Prolog Example

```
1  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]).
```

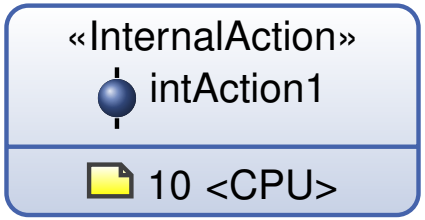


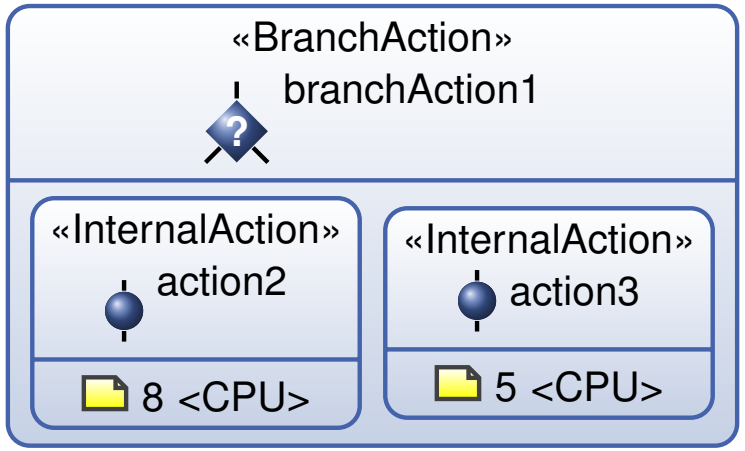


Prolog Example

```
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]).
```

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

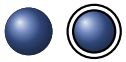
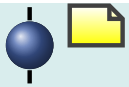
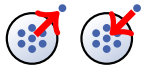
SEFF in the Palladio Metamodel

Karlsruhe Institute of Technology

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

SEFF constructs

StartAction/ StopAction		Start and stop of the control flow
InternalAction with Resource Demands		Abstraction of the internal logic und the resulting resource demands
ExternalCallAction		Use a service provided by another component Includes variable usage and variable characterisation for definition and value assignment
SetVariableAction		Set and change of variables Assignment of parameter values is done analogously to ExternalCallAction
AcquireAction/ ReleaseAction		Acquire and release passive resources (e.g. a semaphore)

SEFF constructs

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