Praktikum Ingenieurmäßige Software-Entwicklung

Palladio Component Model – Part II (PCM)

Prof. Dr. R. H. Reussner (reussner@ipd.uka.de)
Lehrstuhl Software-Entwurf und –Qualität
Institut für Programmstrukturen und Datenorganisation (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH)

Outline

1. Introduction
   a. Roles, Process Model, Example
   b. Solver (Simulation, Analytical Model)
2. Component Developer
   a. Repository
   b. Component, Interface, Data Types
   c. SEFF
3. Stochastical Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies

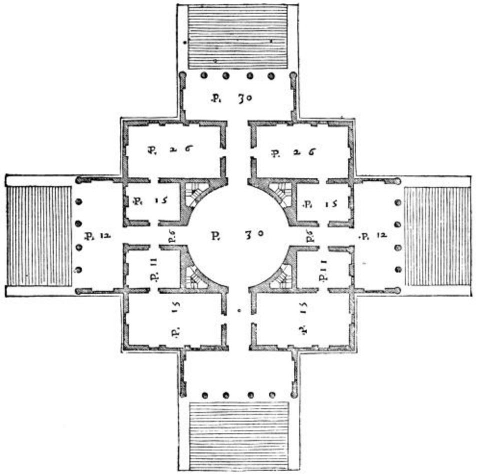
Lecture 1

Lecture 2

Lecture 3

Service Effect Specification

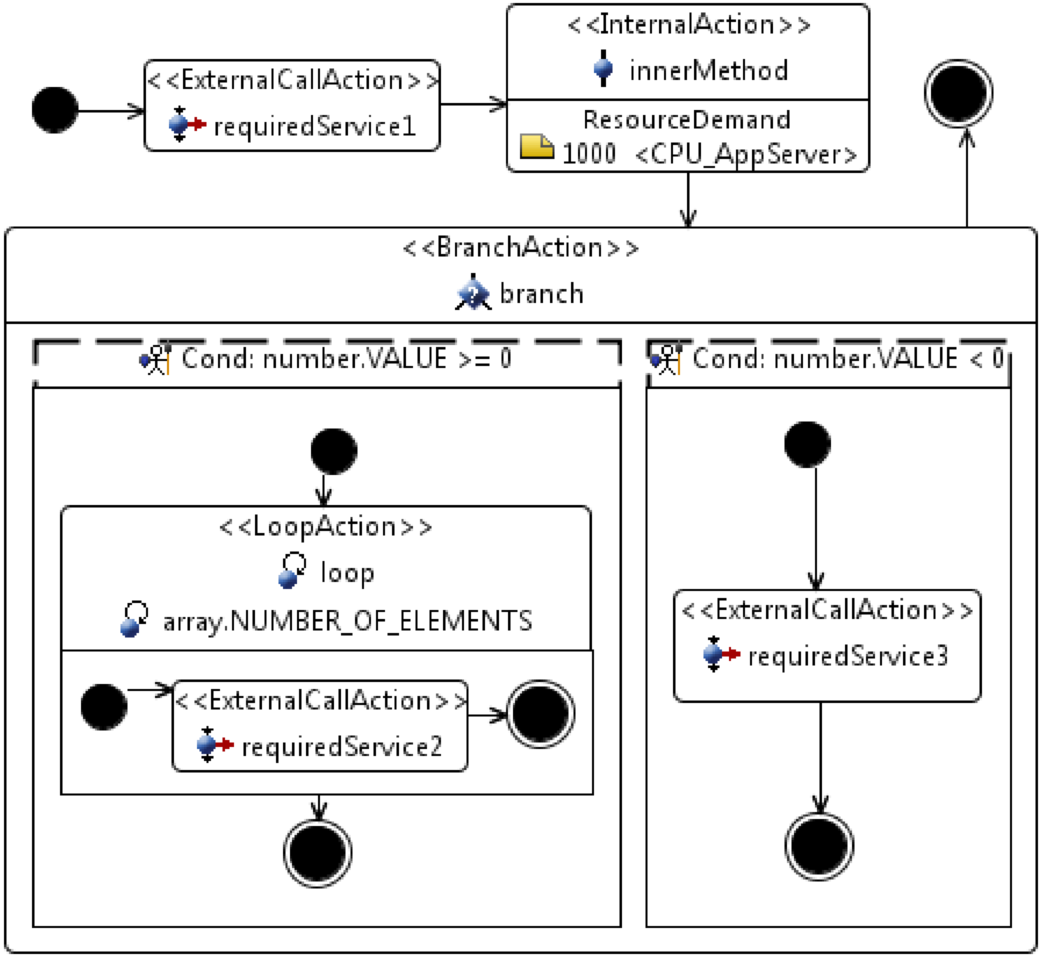
- Description of the external visible actions of a component`s service
- Abstraction of internal behaviour
- Describes relationship between provided component side and required component side
- Can be parameterised by variables (see next lecture)



Conceptual Sources of the SEFF

- CBSE Parametric Contracts
- UML2 Activities
  - Notation
  - Some semantic ideas
- Software Execution Graphs of SPE
- Core Scenario Model (CSM) used in PUMA (Performance by unified model analysis)
- KLAPER (Kernel Language for Performance and Reliability Analyses)

Service Effect Specification Overview



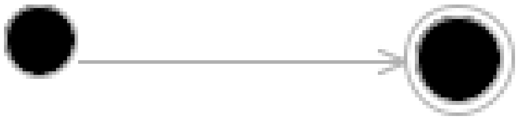
Component Developer

Conceptual Overview

- Resource Actions
  - Internal Action
  - Acquire- & Release Action
- Communication
  - External Call Action
- Control Flow
  - Loops
  - Branches
  - Fork

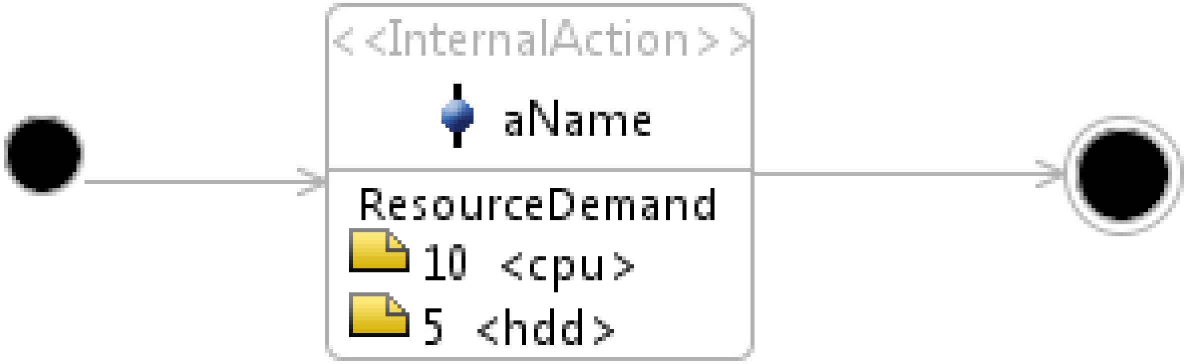
Start and Stop Action

- Mark beginning and end of activities
- Every sub activity also has to have one start and one stop action



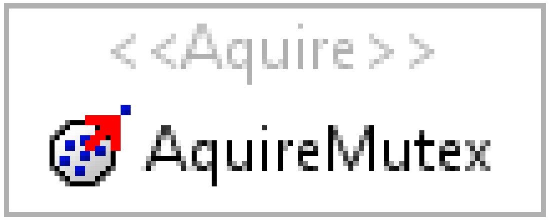
Internal Action

- Modells component internal activities like doing a computation
- Specifies the summed up resource demand for the action
- Different resources can be used



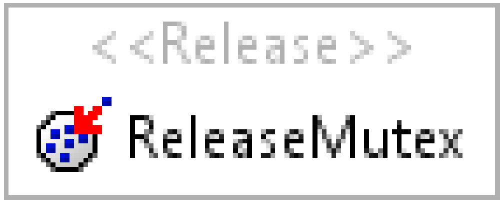
Acquire Action

- Acquire Actions model the acquisition of a limited resource (Passive Resource Type)
- Examples are Database Connections, Pooled Threads, Mutex Locks, …
- Serve as synchronisation mechanism for concurrent executions



Release Action

- Release acquired resources again
- Other waiting jobs can use the resource now
- A FIFO strategy controls the order of acquisition for the waiting jobs

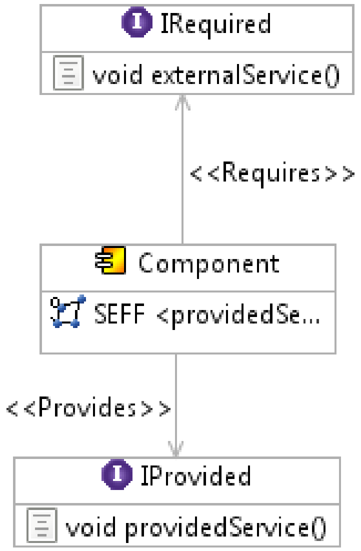


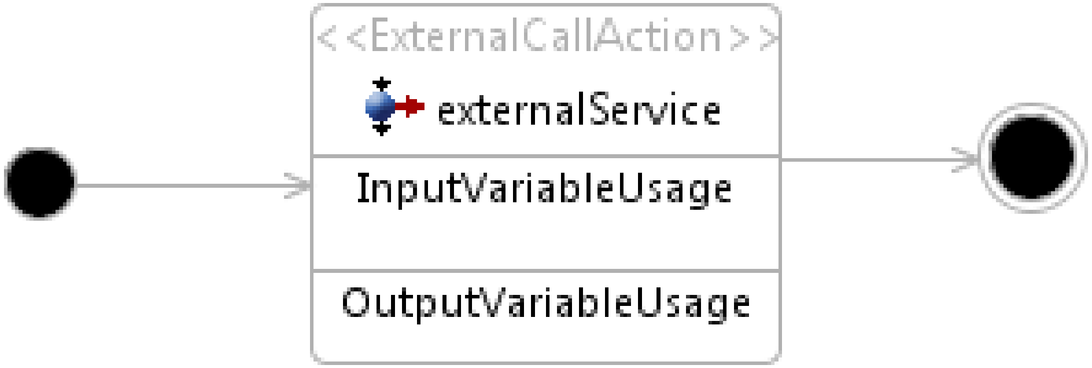
<<Release>>

ReleaseMutex

ExternalCallAction

- Models a call using any of the required roles
- A call *must* use a required role
- Parameter passing and returning can be specified (next lecture)





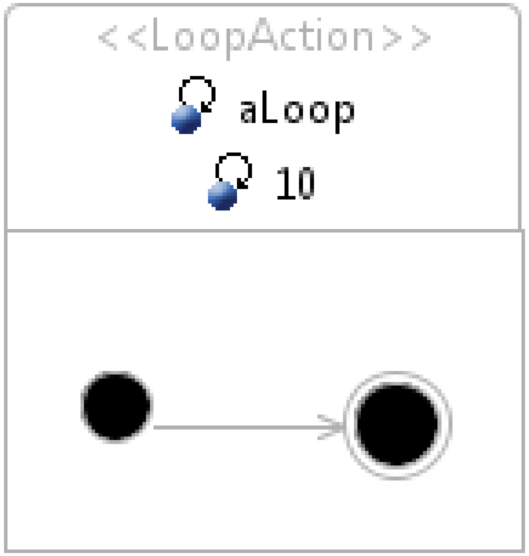
Control Flow Specification

- Control flow constructs model the course of actions like in SPE
- Concepts available
  - Loops
    - Loop
    - CollectionIterator
  - Branches
    - Probabilistic
    - Guarded
  - Forks

Loops

- Models repeated behaviour
- Iteration count has to be specified explicitly

```
for(int i=0; i<10; i++){
    …
}
```



CollectionIteratorAction

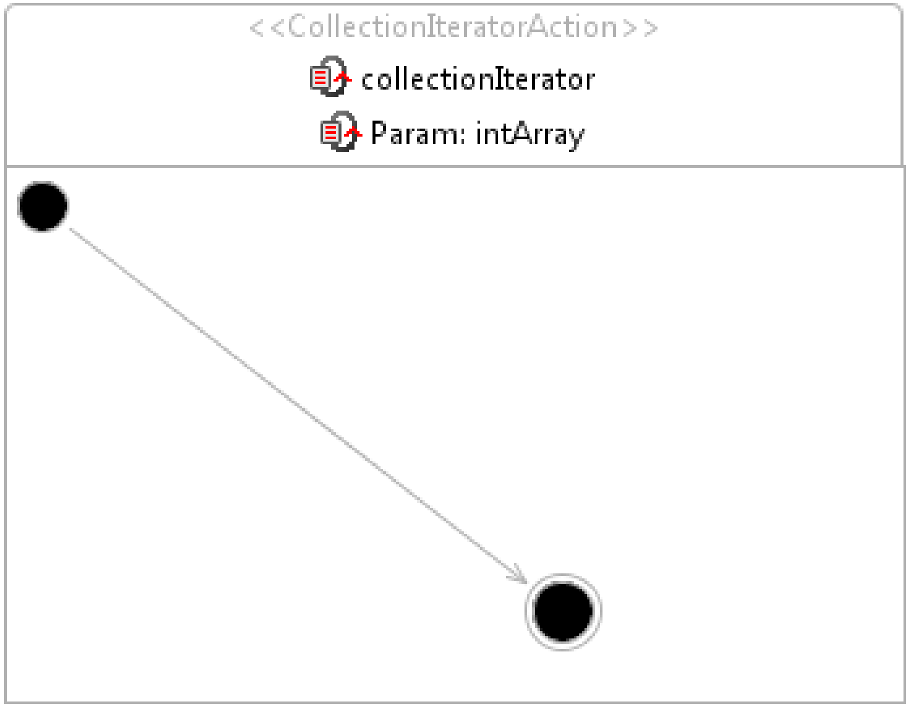
- CollectionIteratorActions iterate over all elements in an instance of a CollectionDataType
- The behaviour is executed for every element

CollectionIteratorAction

```
void myMethod(int[] intArray)
{

   for (int x:intArray) {
      do
      …
   }

}
```



Semantic details

- Loop and CollectionIterator semantics preview
  - Inner Actions are evaluated *stochastically independent* wrt. to contained parametric dependencies
  - Collection Iterator Actions are evaluated *stochastically dependent* wrt. to the characterisation of the parameter being iterated
- Examples and further details in next lecture

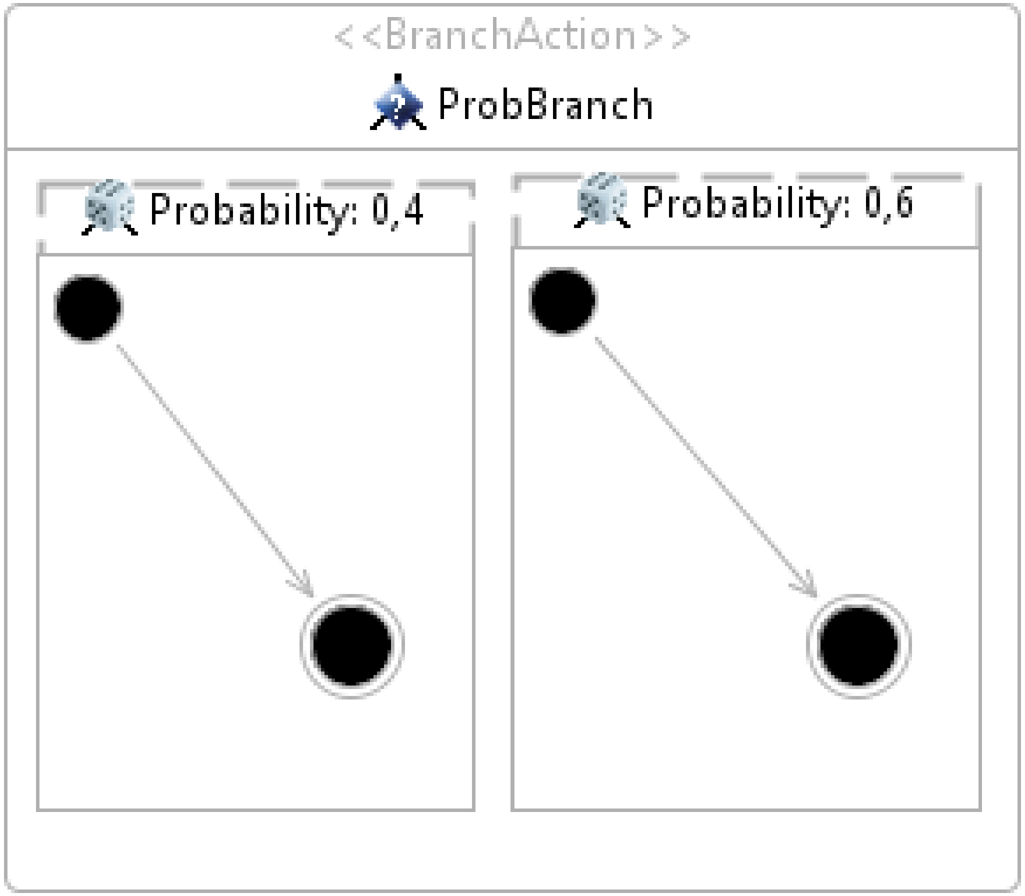
Branches

- A branch models optional parts of the control flow
- Exactly one branch must be executed, to model an option an empty alternative branch has to be specified
- Two flavours:
  - Probabilistic Branch Transitions: A probability can be specified for every branch which is the probability of executing the branch. Probabilities have to sum up to 1
  - Guarded Branch Transitions: Guards „protect“ the execution of the branch. Execute branch which guard is true

Probabilistic Branches

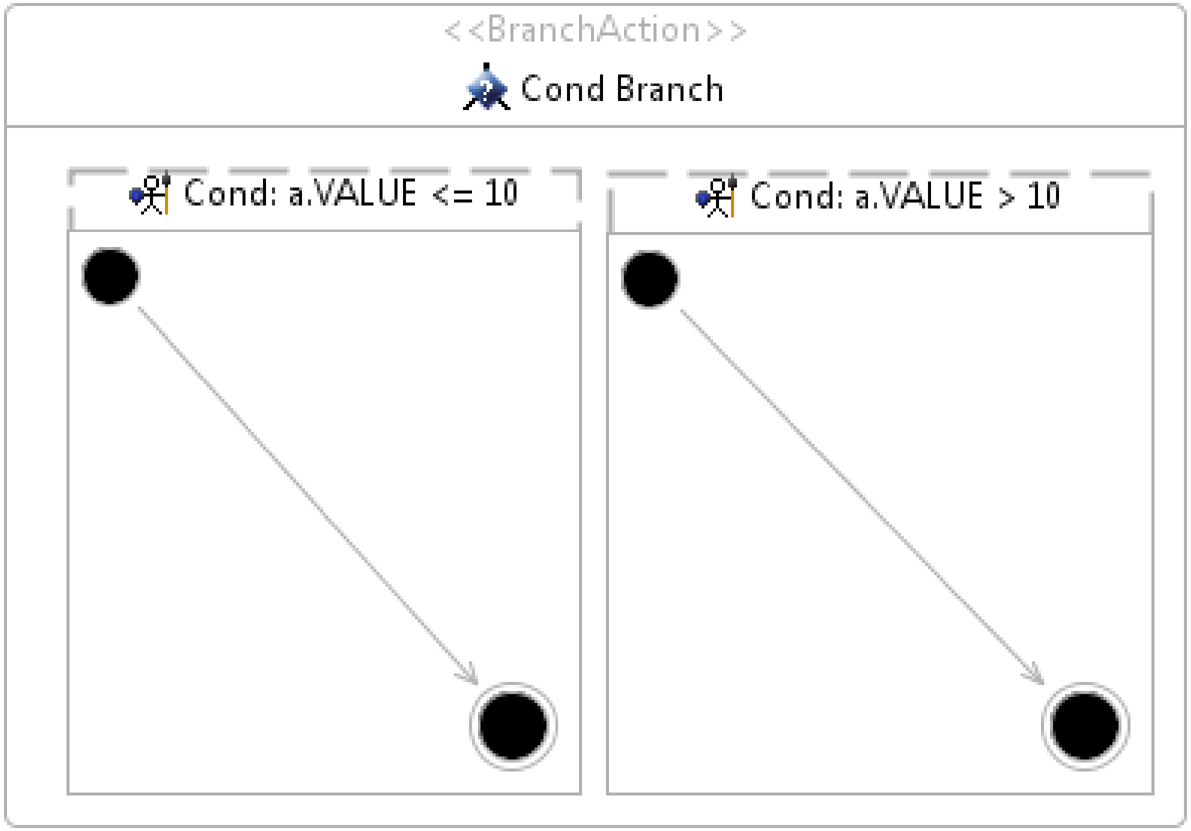
```
If (someCondition) {
        …
}
else{
        …
}
```

someCondition == true in 40% of all cases



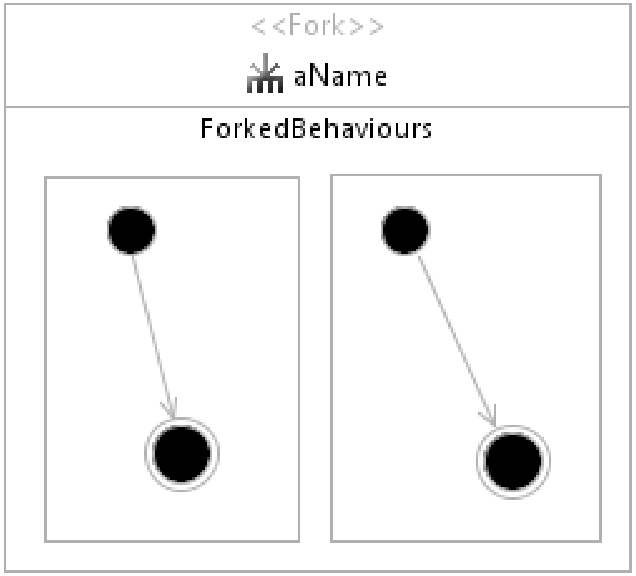
Guarded Branches

```
a = …
If (a <= 10) {
      …
} else {
      …
}
```



Fork

- A fork spawns n threads and waits for them to finish
- After finishing the forked threads the main thread continues



Now: Exercises in the Tool

- Switch to Eclipse!

Lessons Learned Today

- What is a SEFF?
- What is it used for?
- Concepts
  - Resource Actions
    - Internal Action
    - Acquire- & Release Action
  - Communication
    - External Call Action
  - Control Flow
    - Loops
    - Branches
    - Forks