Universität Karlsruhe (TH)
Research University ▪ founded 1825

Fakultät für Informatik

SDQ

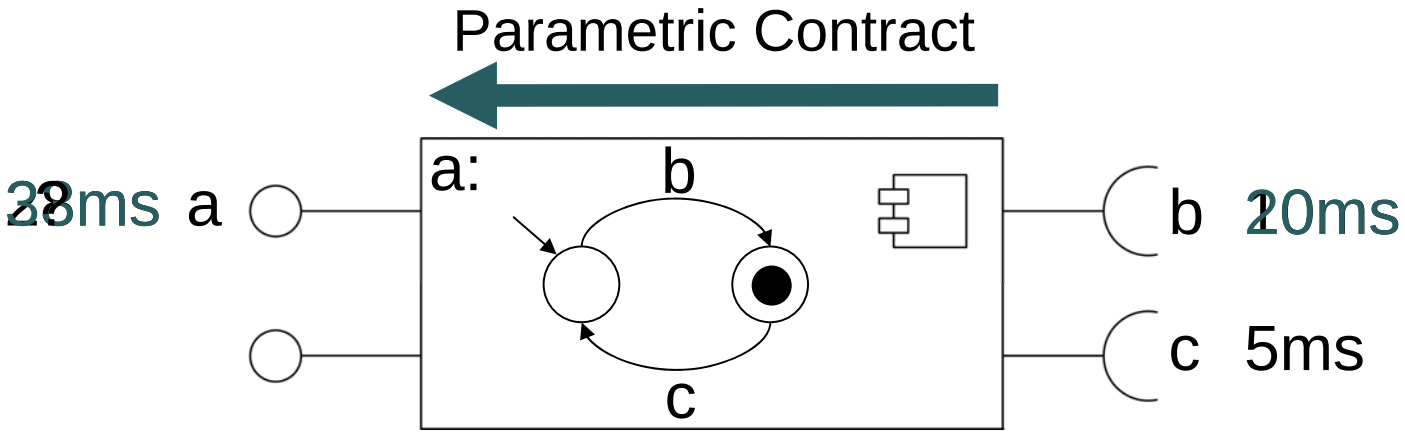
Describing Software Components with Parametric Contracts

Ralf Reussner (reussner@ipd.uka.de)

Chair Software Design and Quality
Institute for Program Structures and Data Organization (IPD)
Fakultät für Informatik, Universität Karlsruhe (TH),
Karlsruhe Institute of Technology

Karlsruhe Institute of Technology

Parametric Component Properties



- Design-by-Contract for Software-Components
- Specification of the relation between provides and requires-interface(s) (and the usage profile and execution environment)

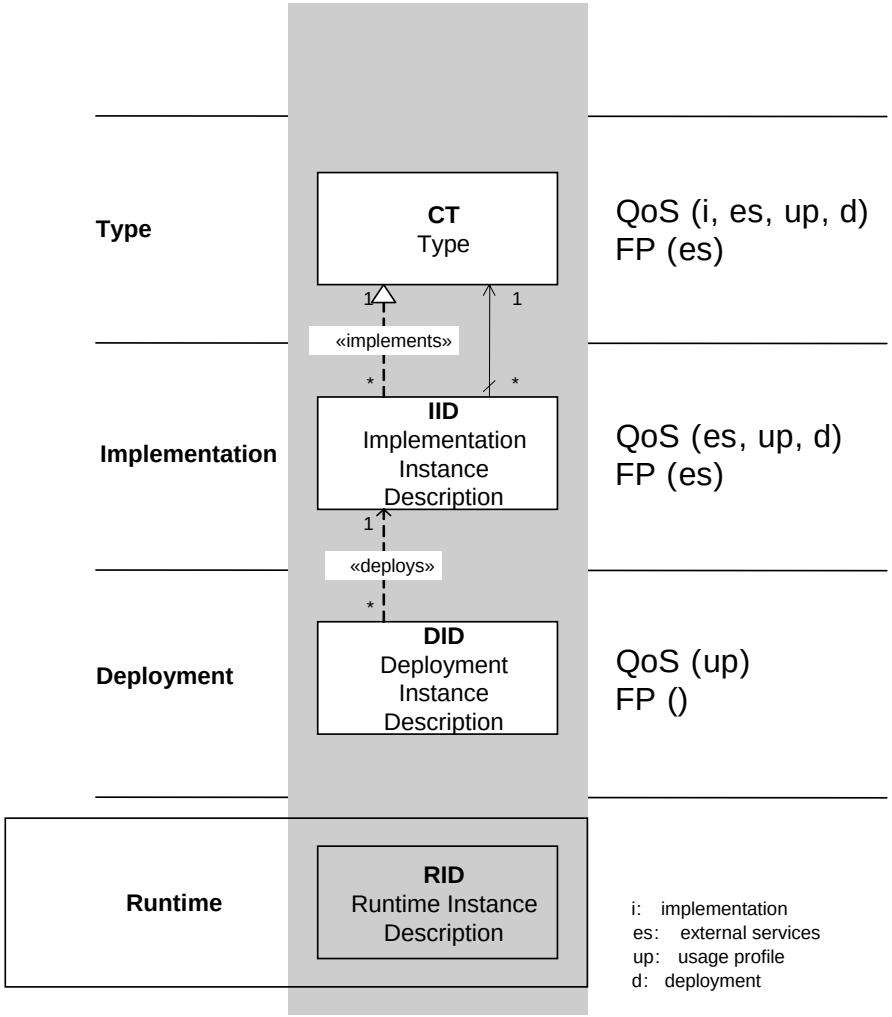
Overview

- Contracts and Parametric Contracts
- Protocol Adaptation with Parametric Contracts
- Prediction of Non-Functional Properties with Parametric Contracts
  - Reliability
  - Performance
- Conclusions

What is a component?

- "A component is a contractually specified building unit of software which can be readily composed or deployed."
  - "readily composed or deployed":
    - without having to understand the interna as a human
    - these are the two main things to be done with components
  - not necessarily "black-box": Information on interna can be available to tools.
- "Components are for composition, much beyond is unclear…" (Clemens Szyperski)

Different Abstraction of Components



Not considered within the Palladio ComponentModel

i: implementation
es: external services
up: usage profile
d: deployment

Contractual Use of Methods

- Design-by-contract (B. Meyer, 1992)
  "Supplier guarantees post-condition if pre-condition is fulfilled by client."
  [Bertrand Meyer, Object Oriented Software Construction, Prentice-Hall, 1997 2nd ed.]

$$\underbrace{\text{CustomerRecord}}_{\text{Post-condition}}\ \text{getCustomerRecord}\ (\underbrace{\text{String ID}}_{\text{Pre-condition}})$$

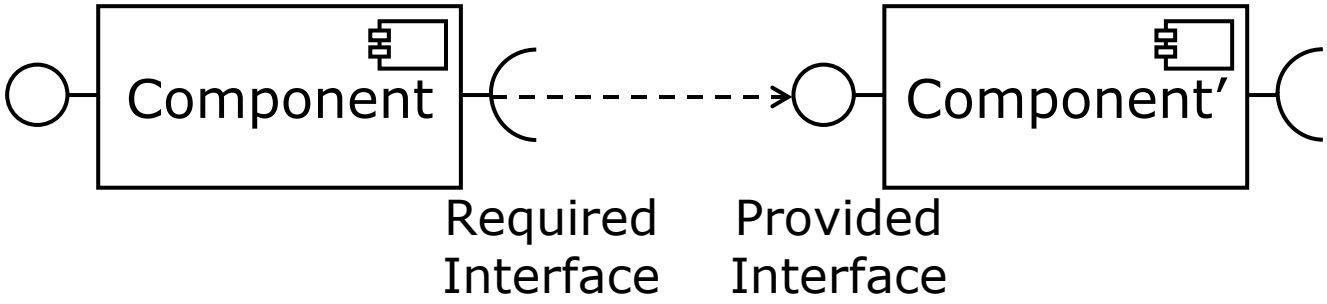
Design-by-Contract (Bertrand Meyer, 1992)

	Obligations	Benefits
Client	Satisfy precondition: (attached database)	From postcondition: Get customer returned or **`null`**
Supplier	Satisfy postcondition: Get right customer or return **`null`**.	From Precondition: Handle queries only if database is attached.

Benefits of Contracts

- Developing less error-prone software through:
  - Better documentation
  - Reduced error-handling code
  - Clear responsibilities
  - Results of run-time checking as a debugging aid

Contractual Use of Components



Use of Components

There are two different times of component usage:

1. At system run-time (== component run-time): call of component services
2. At system design-time or system configuration time: plugging in of components

Contractual Use of Components

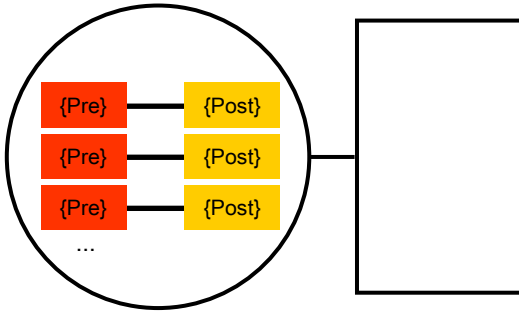
- Design-by-contract (B. Meyer, 1992)
  - "Supplier guarantees post-condition if pre-condition is fulfilled by client."
- Contractual Use of Components
  - (at system design- / configuration-time)
  - "Component guarantees provided services (as described in the provides-interface) if the environment offers all required services (as described in the requires-interface)."

Contractual Use of Components

- Pre-condition
  - Required services
  - Information concerning required services (sequences, QoS)
- Post-condition
  - Offered services
  - Information concerning offered services (sequences, QoS)
- Invariant
  - Relation between provided and required services (as this is independent from the usage context)

Contractual use of Components

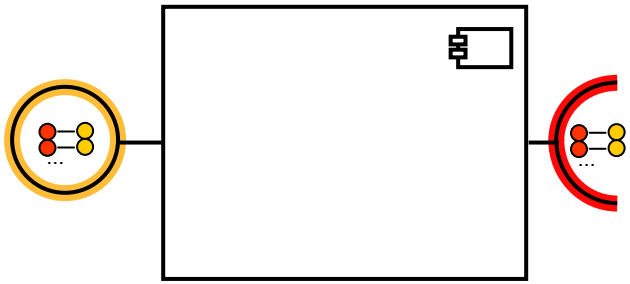
- Design-by-contract (B. Meyer, 1992)
  - " The service supplier guarantees the post-condition, if the client guarantees the precondition of the service."



- Contractual Use of Components

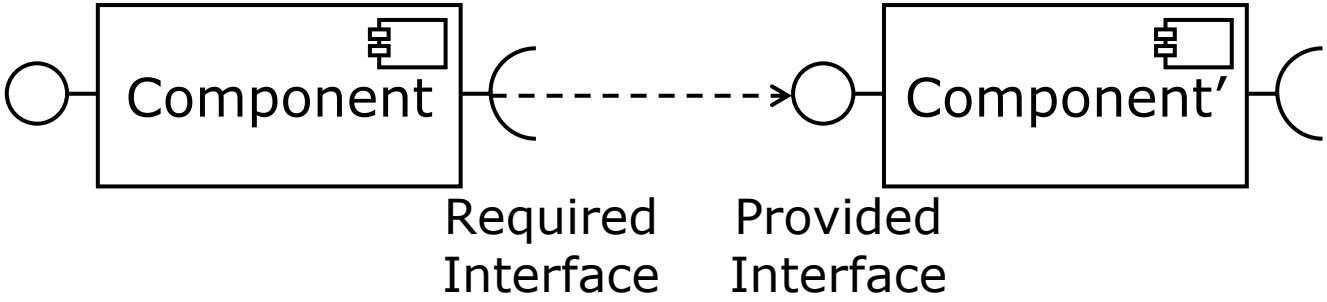
  (at system-(re-) configuration time)

  "The component offers the provided services (as specified in the provides interfaces), if the environment guarantees the required services (as specified in the requires interfaces)."



Contractual Use of Components

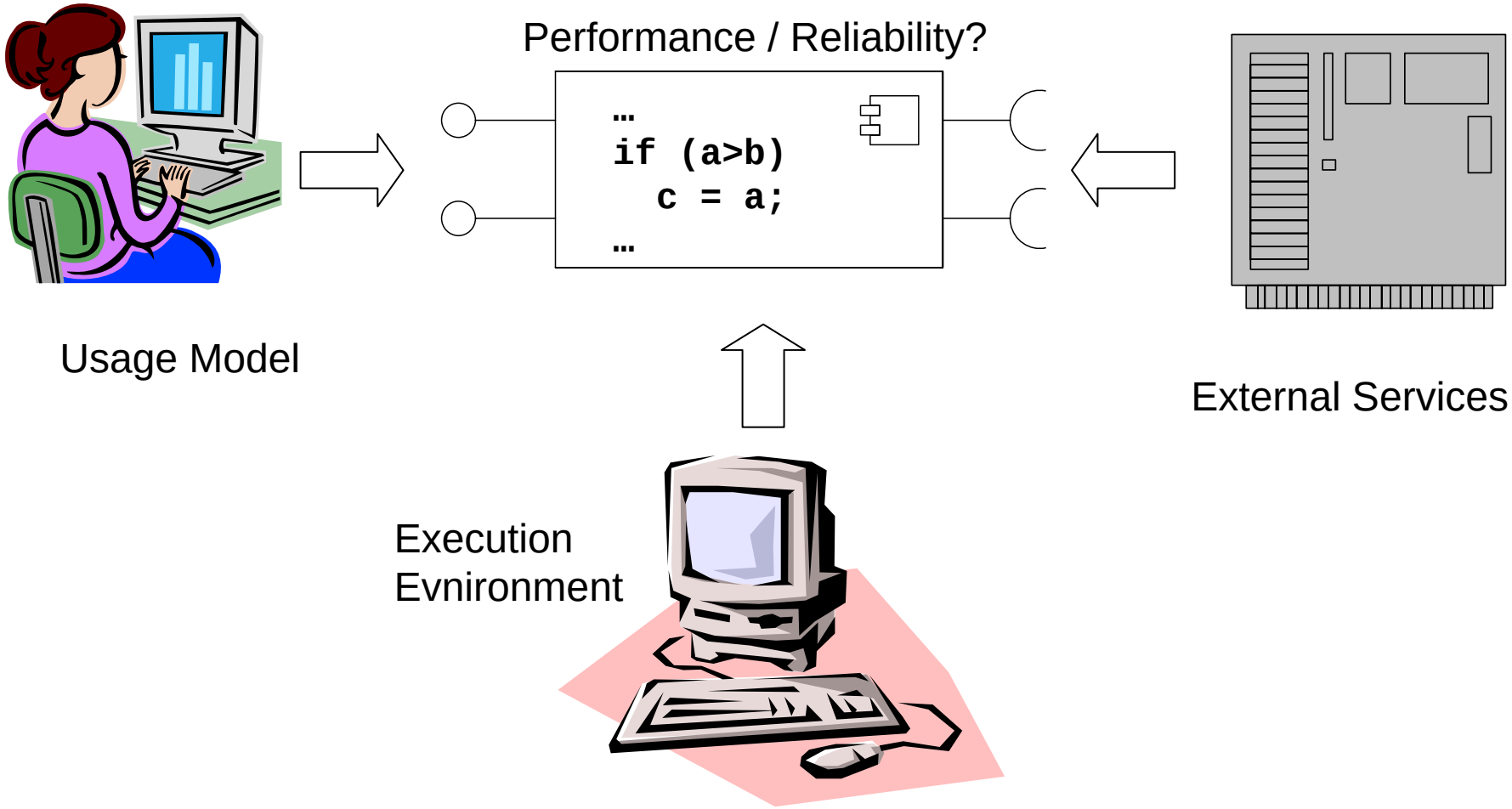
- Contractual use corresponds to interoperability checks: $R_C \subseteq P_{C'}$
- Static checks of practical relevance



Consequences (1)

- Interoperability checks during design- or reconfiguration time are possible by checking the pre-condition against the post-condition of the environment

Problem: Three external Influences on Quality (Performance / Reliability)



Consequences (2)

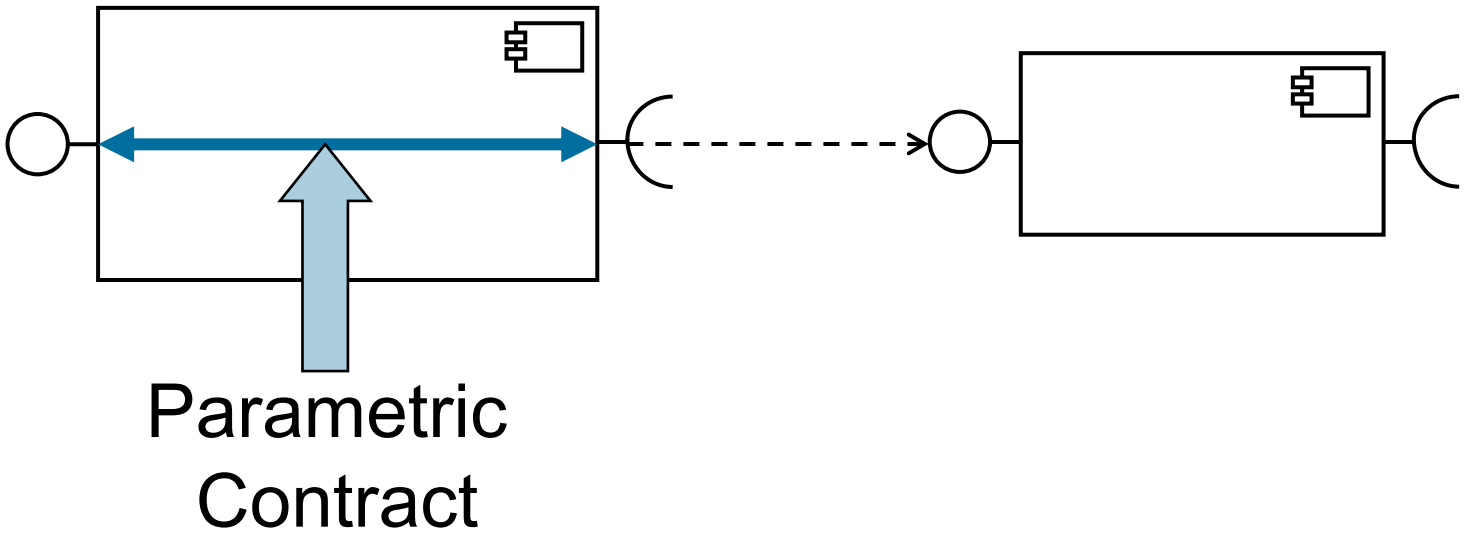
- But:
  - Fixed pre- and post-condition hinder adaptation
  - Pre- and post-condition are hard to fix at design-time
  - In particular, if non-functional properties are to be specified in interfaces.

Components and their contracts need to be adapted to context at deployment, which is after development!

Need for Parametric Contracts

- One single pair of pre- and postcondition insufficient:
  - restriction of reuse
  - need for transparent component adaptation
  - QoS-properties depend on component context and are not fixed.

Parametric Contracts for Components

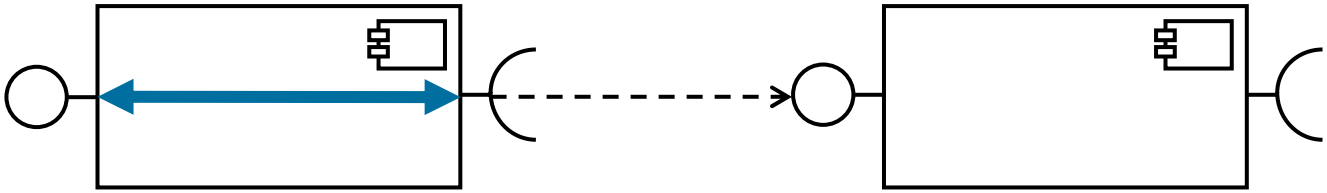


Parametric Contracts

- Link provides- and requires interface of the same component by bijective function p: $\{P\} \rightarrow \{R\}$
- Pre-condition computed out of post-condition (and vice versa)
- Mechanism of automatic component adaptation (besides adaptor generation)

Parametric Contracts for Components

- Computation of environment-dependent provides-interface
- Computation of use-dependent requires-interface ('wp-calculus')
- Fine-grain adaptation of large-grain components



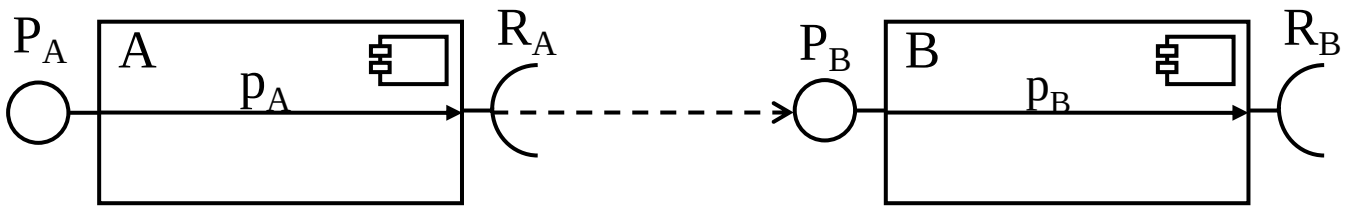
Computation of Parametric Contracts

- Computation of context-dependent provides-interface:

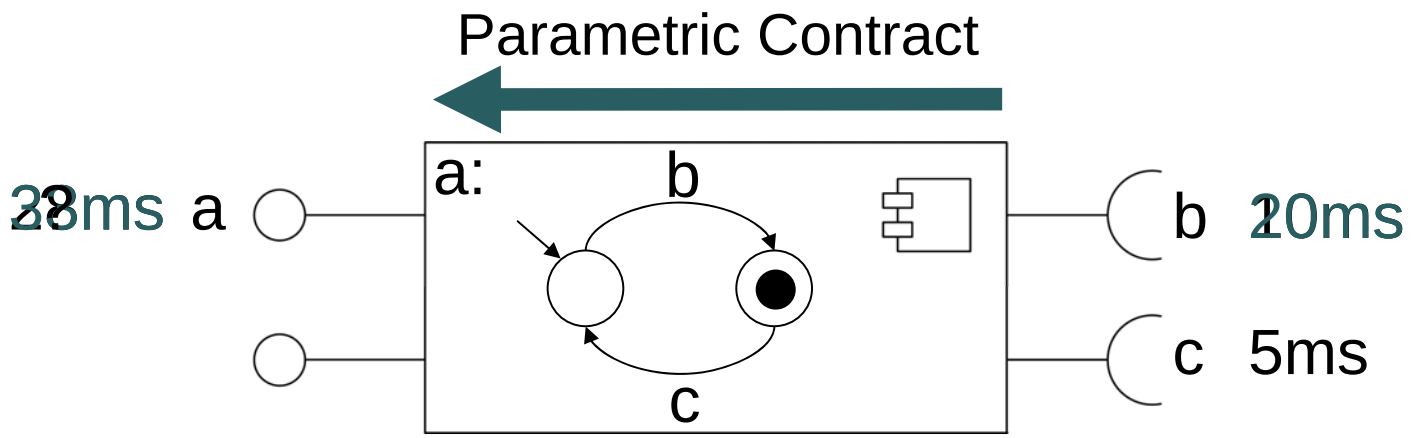
1. $(R_A = p_A(P_A))$
2. $R_{A'} := R_A \cap P_B$
3. $P_{A'} = p_A^{-1}(R_{A'})$

- Computation of context-dependent requires-interface:

1. $P_{B'} := R_A \cap P_B$
2. $R_{B'} = p_B(P_{B'})$

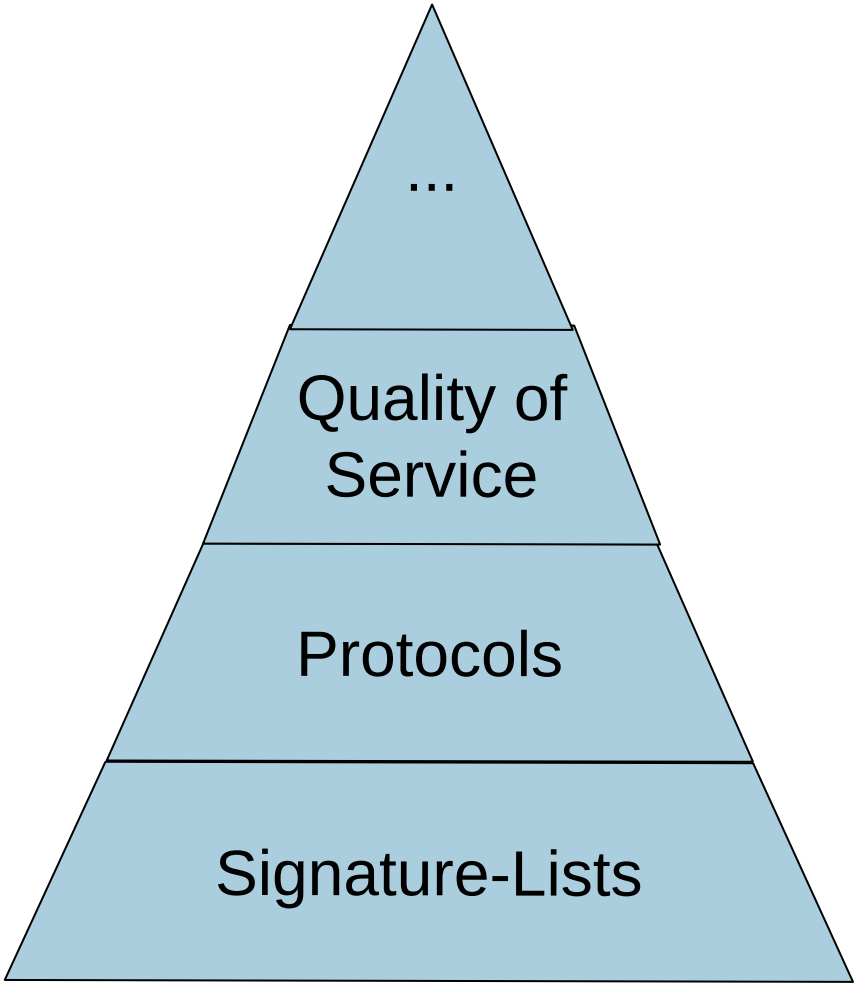


Parametric Contracts



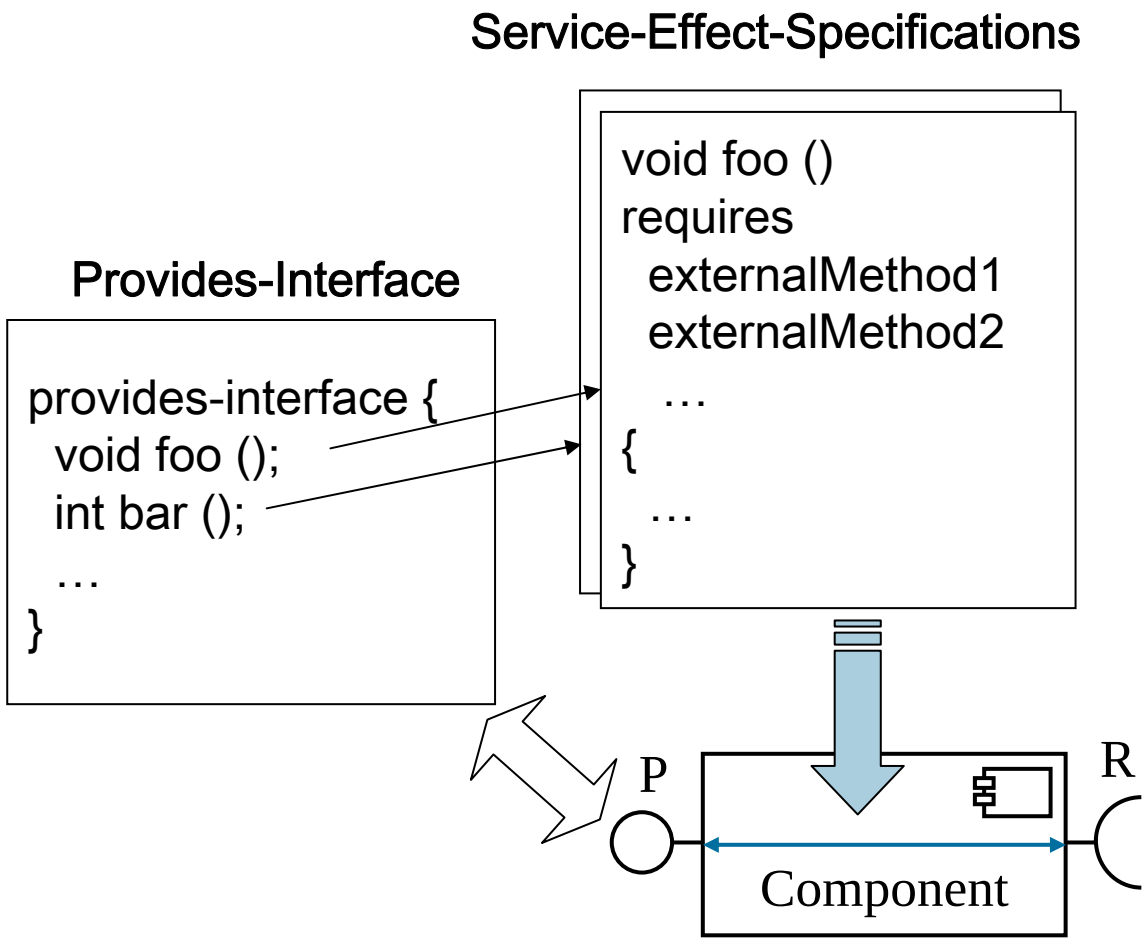
- Lifting the Design-by-Contract Principle to Software Components
- Linking the provided and required services of the same component

Hierarchie of Interface Models



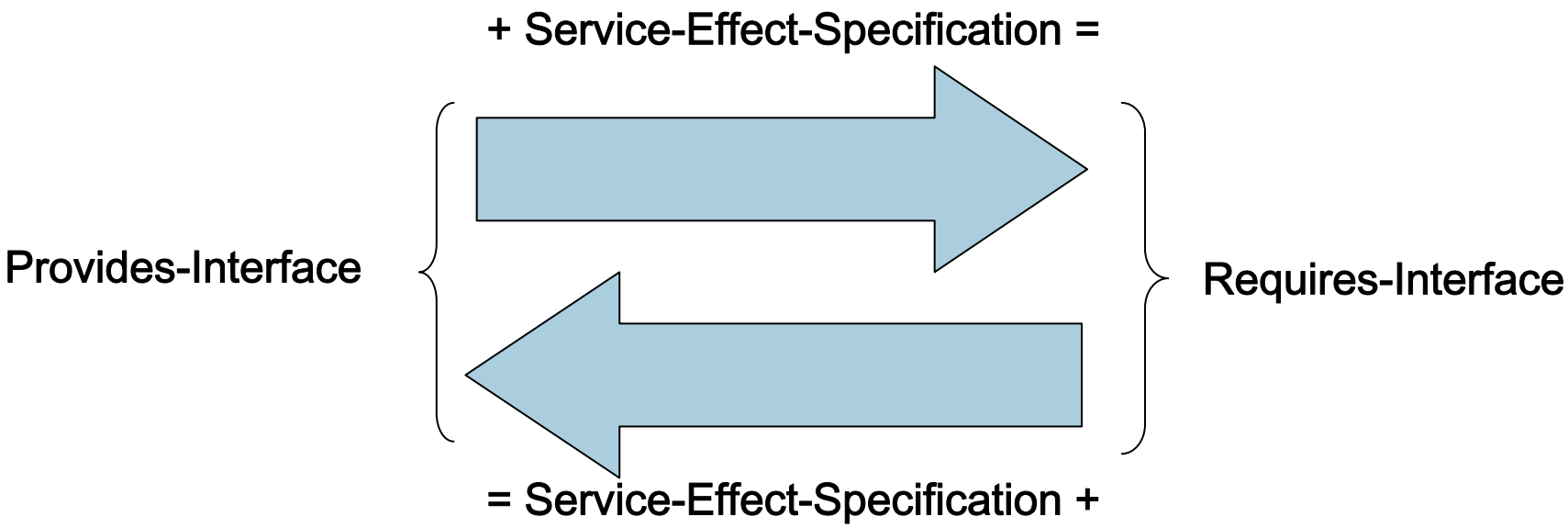
Parametric Contracts for Signature-list based Interfaces

Service Effect Specification: List of required services for each provided service



Parametric Contracts for Signature-list based Interfaces

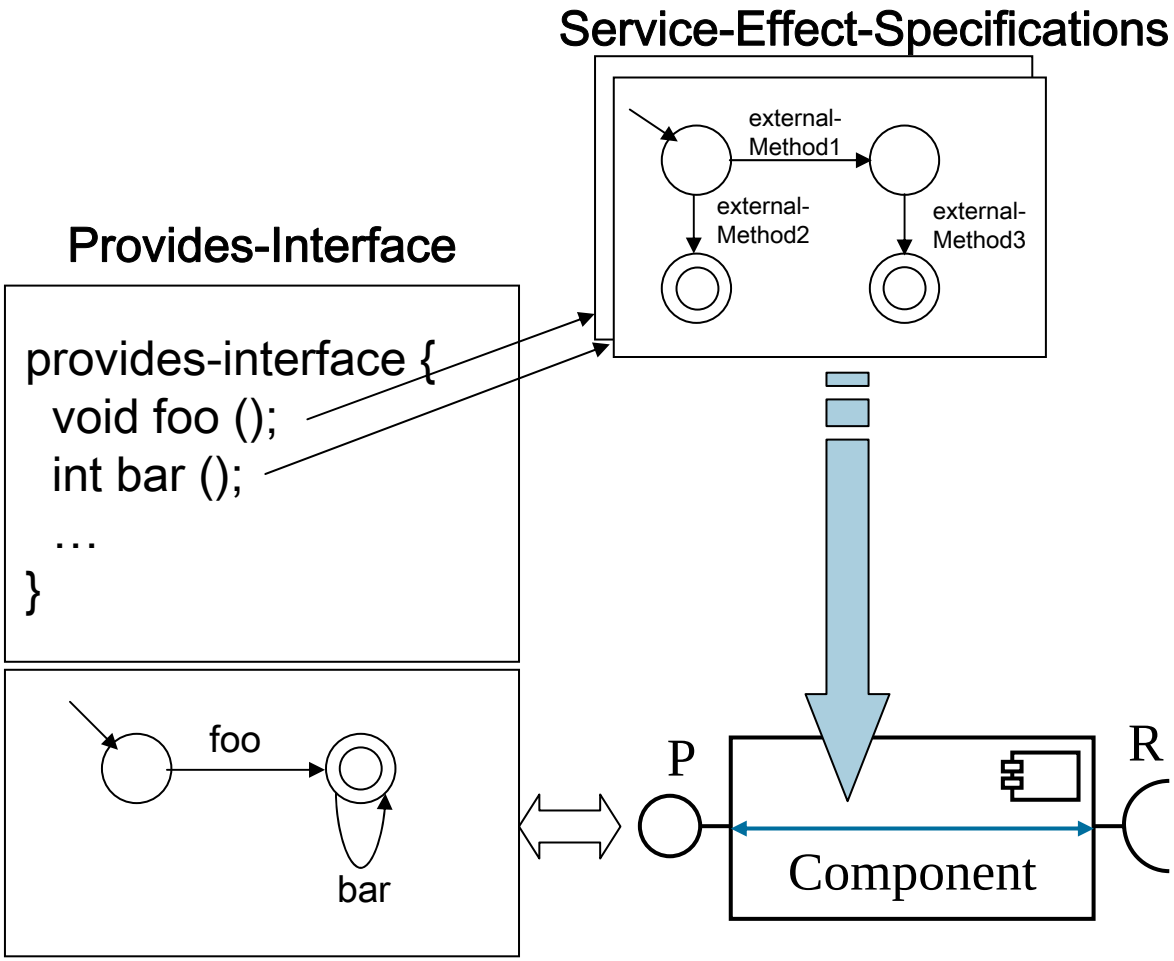
A method n is then and only then required,
if there exists (at least) one method m with n ∈ se(m).



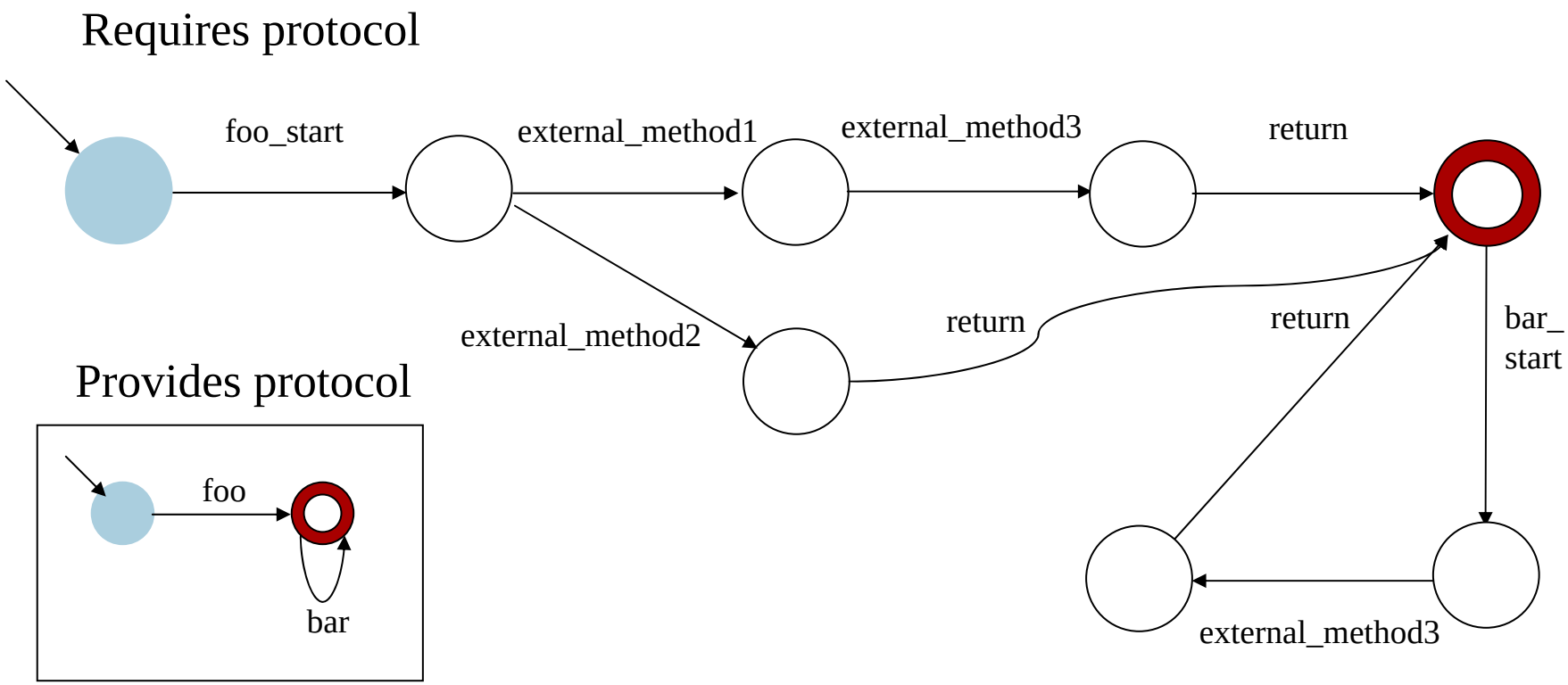
A method n is then and only then provided,
if all its required methods se(n) are available.

Parametric Contracts for Protocol-modeling Interfaces

Service Effect Specification: List of required call sequences of services for each provided service

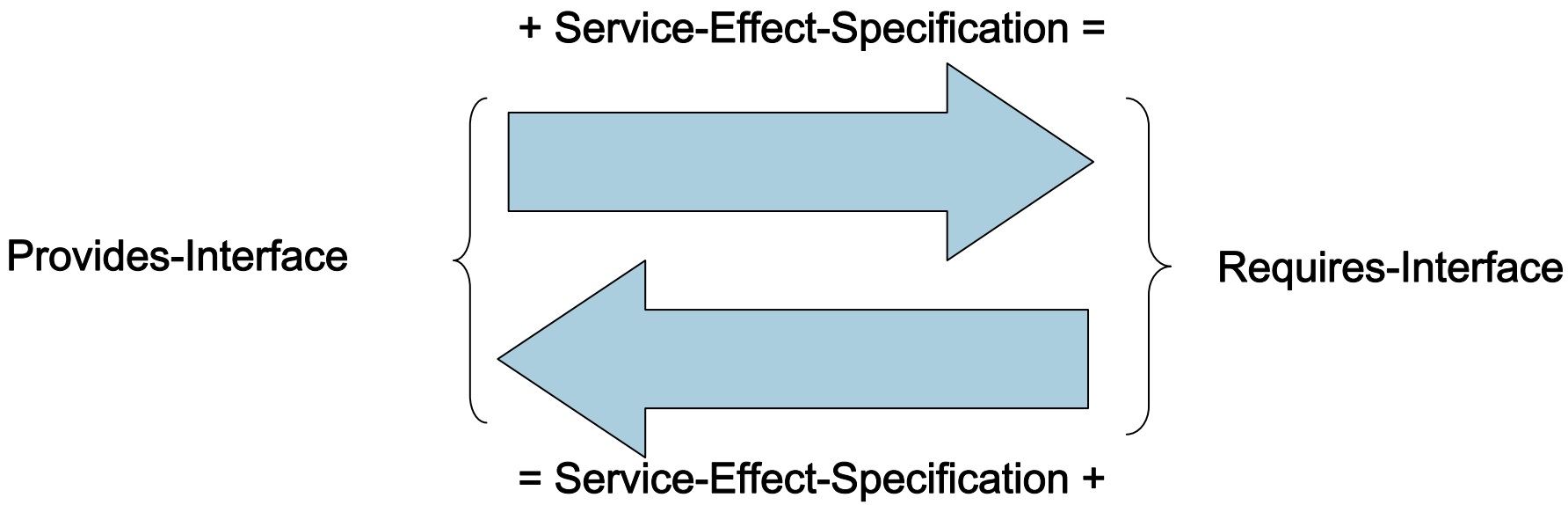


Protocol-Adaptation with Parametric Contracts



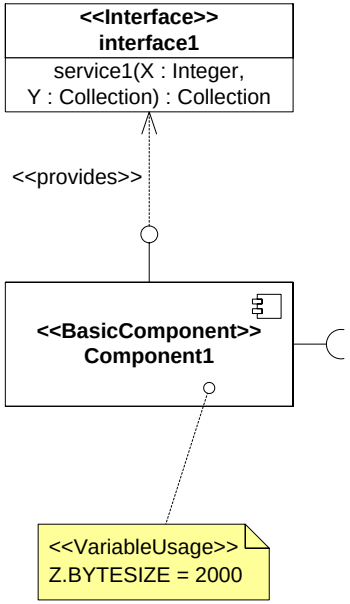
Parametric Contracts for Protocol-modelling Interfaces

A call sequence s is then and only then required,
if there exists (at least) one method m with s $\in$ se(m).

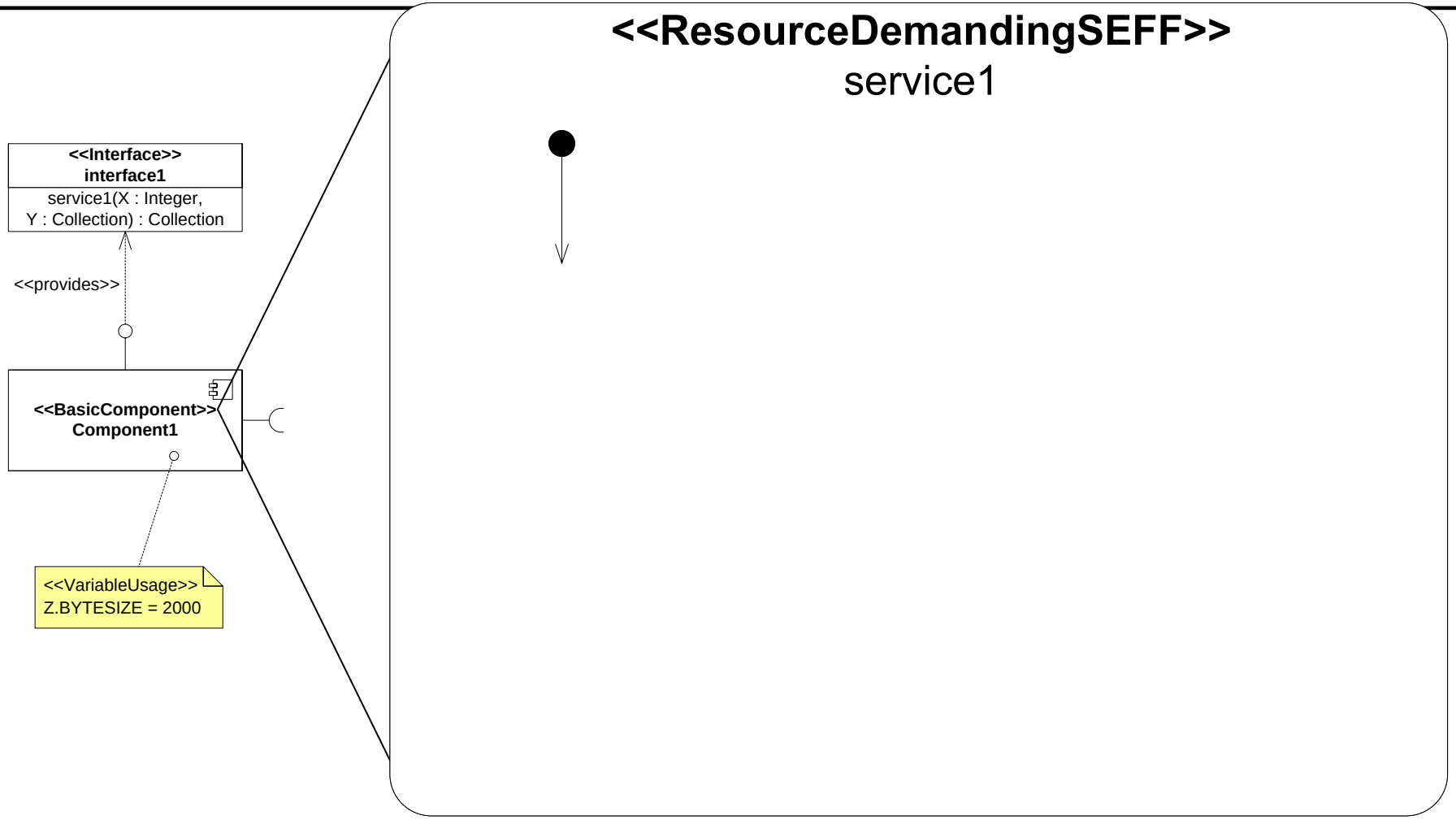


A method n is then and only then provided,
if all its required call sequences se(n) are available.

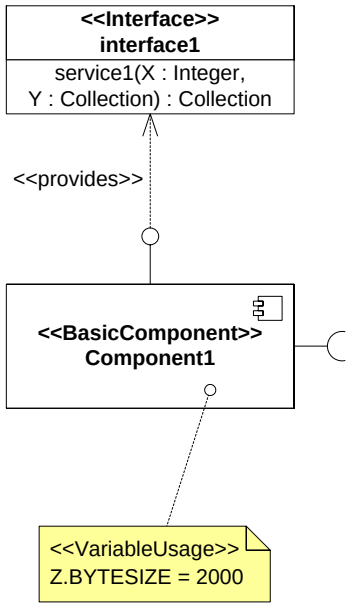
Service Effect Specification

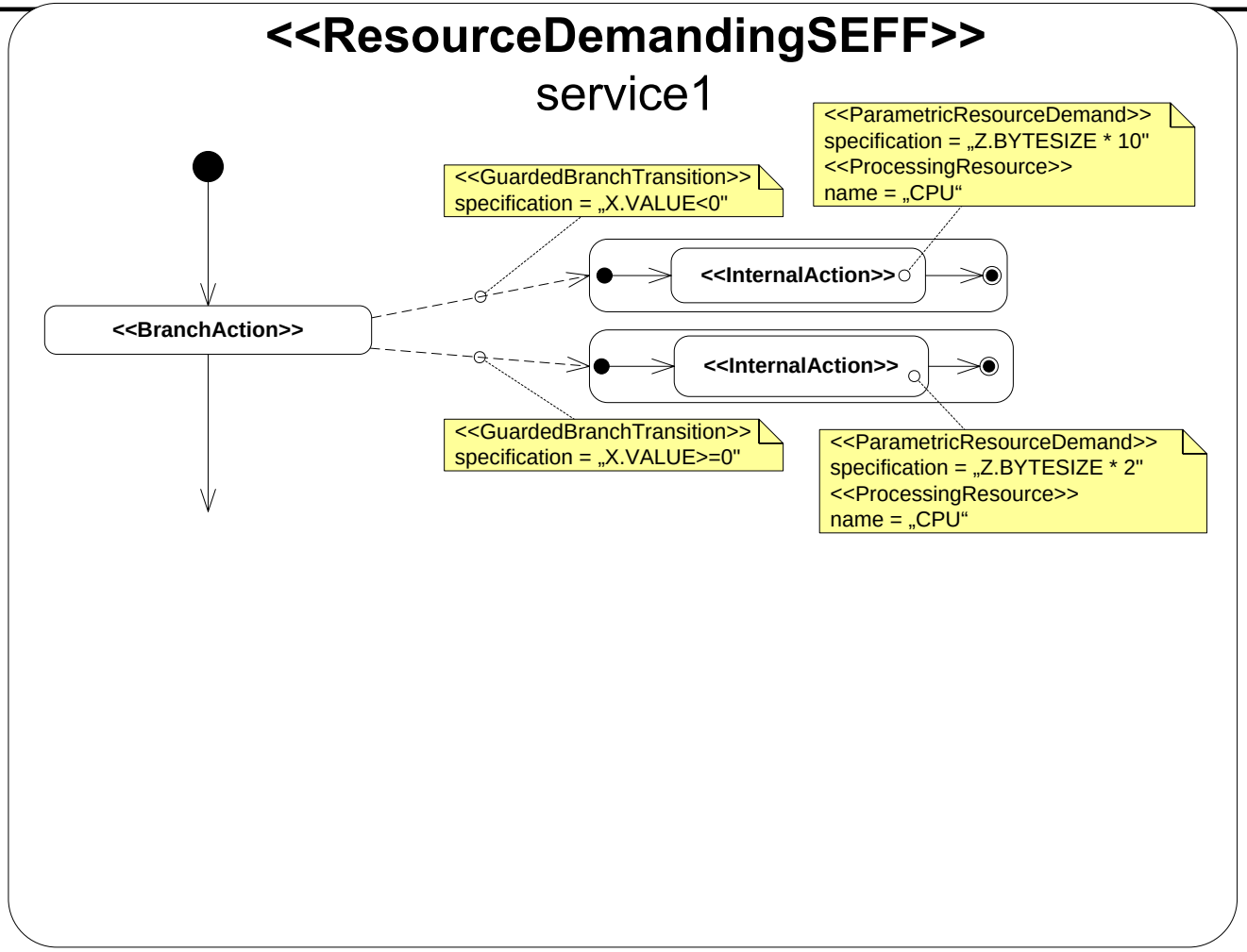


Service Effect Specification

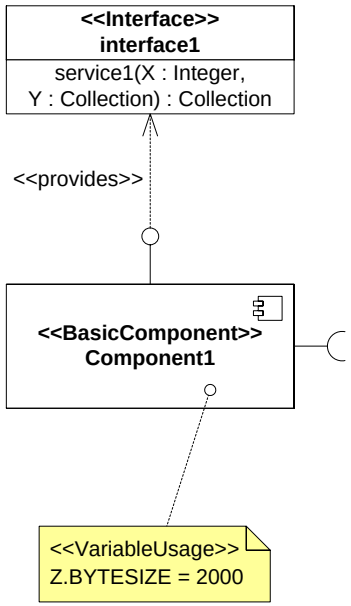


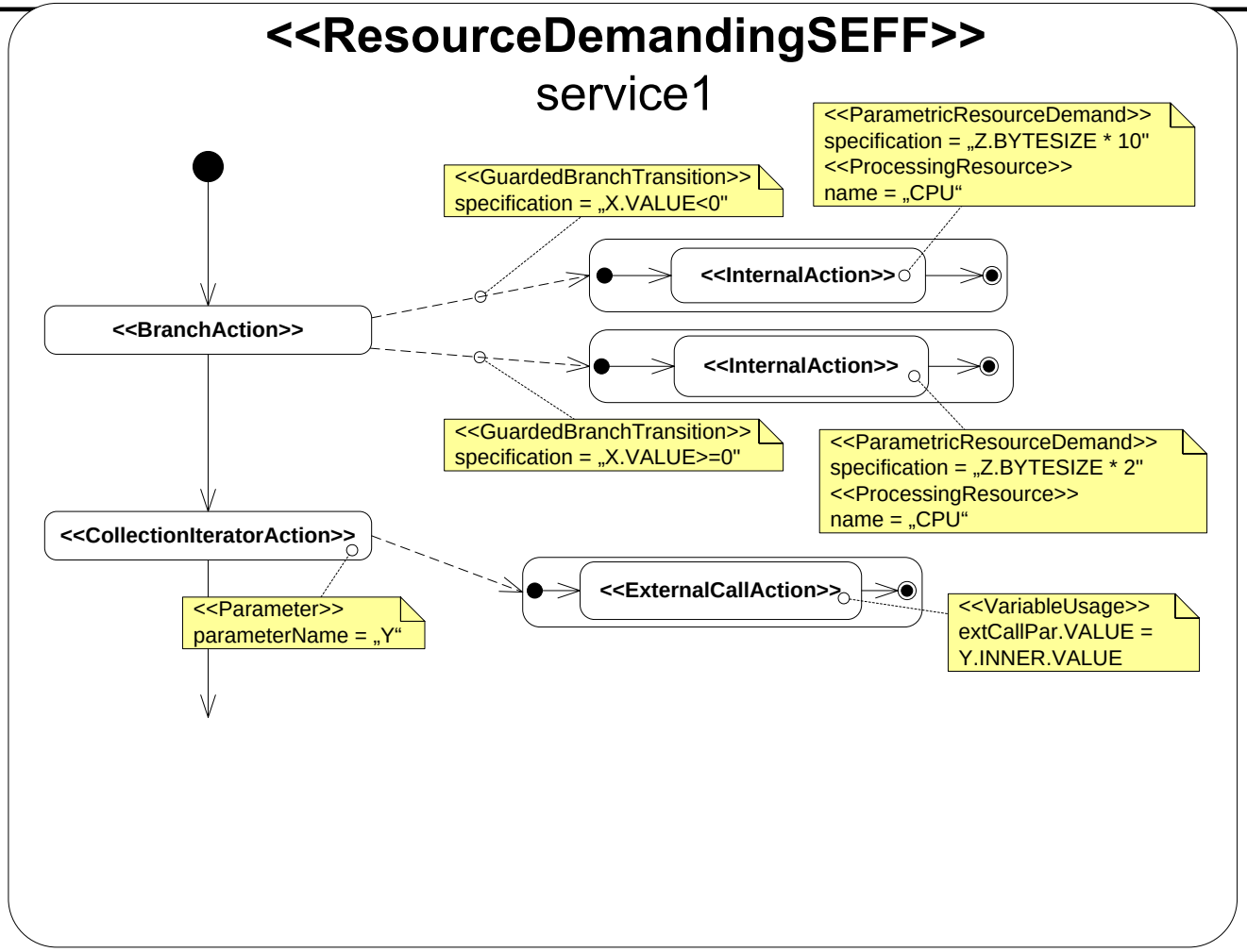
Service Effect Specification



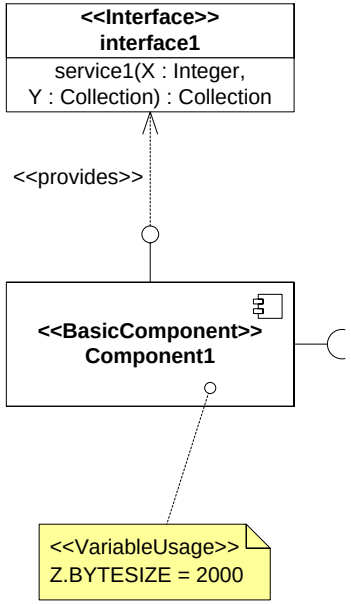


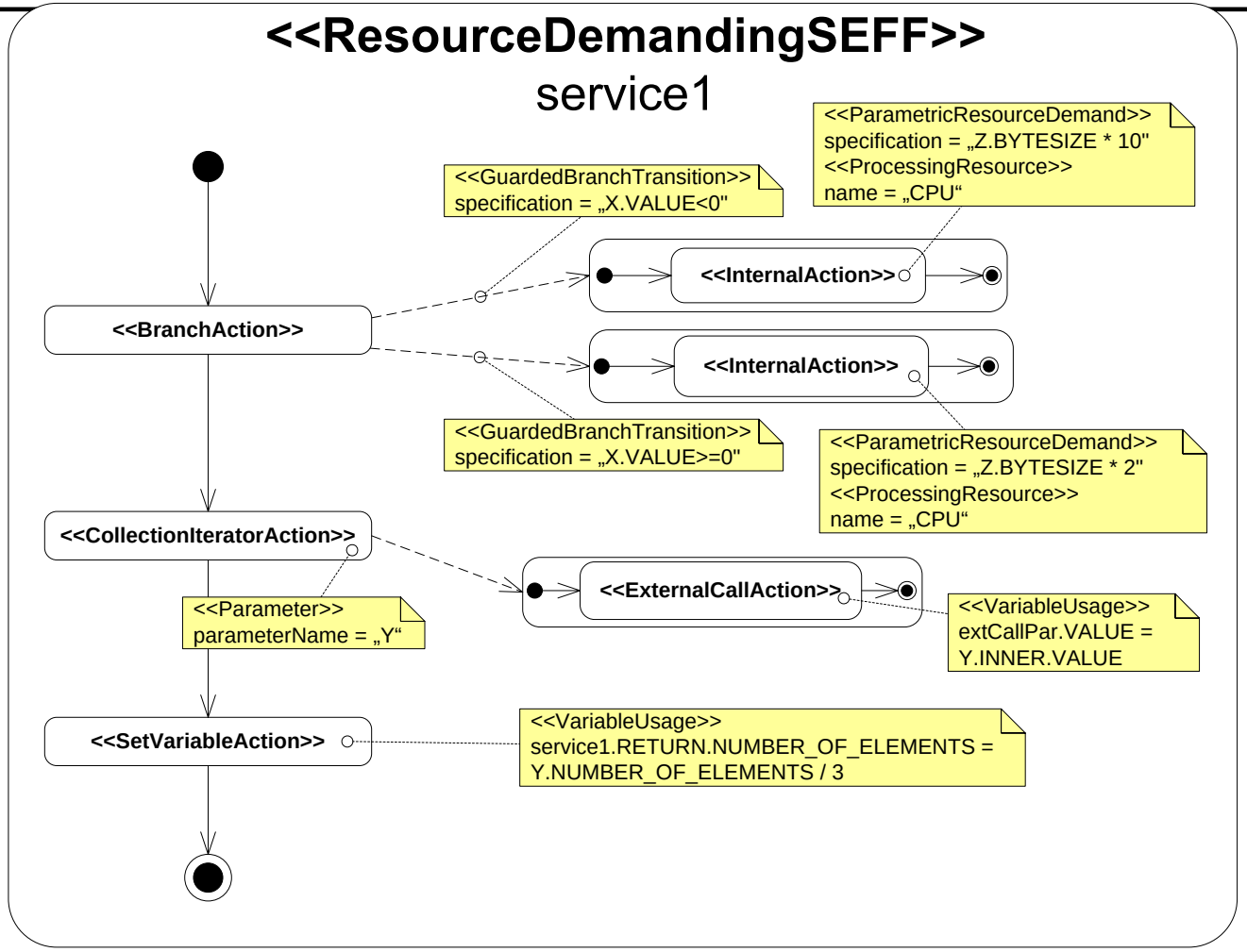
Service Effect Specification



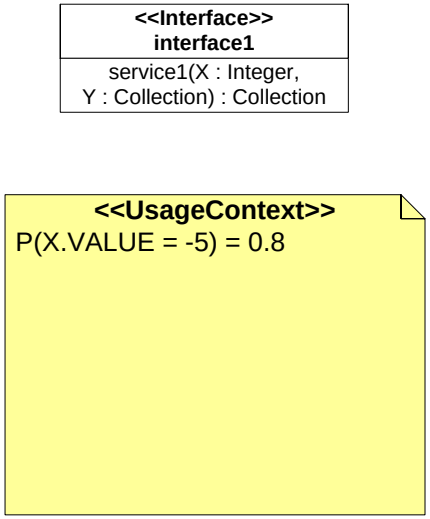


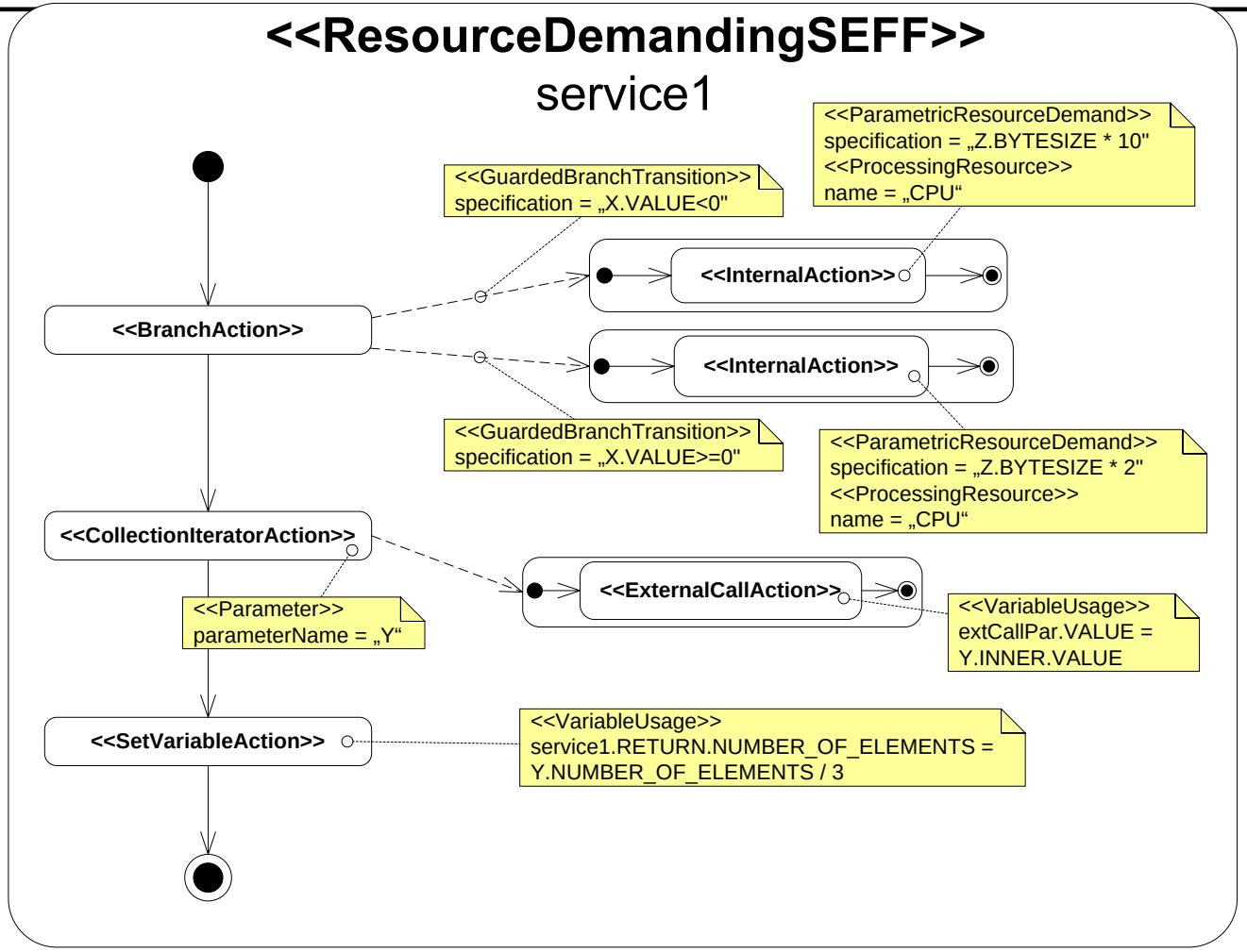
Service Effect Specification



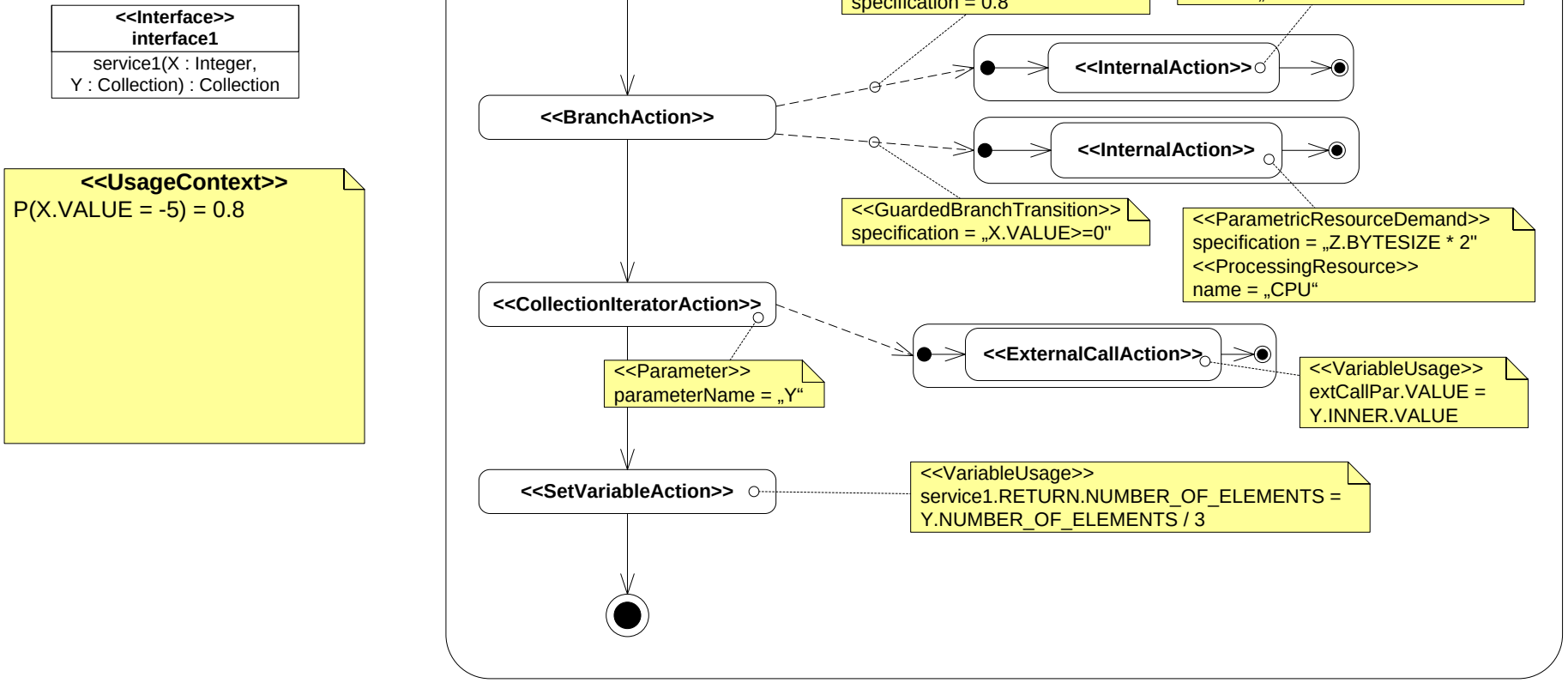


Service Effect Specification

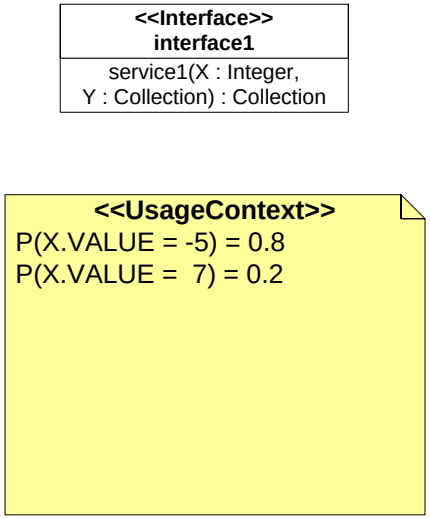


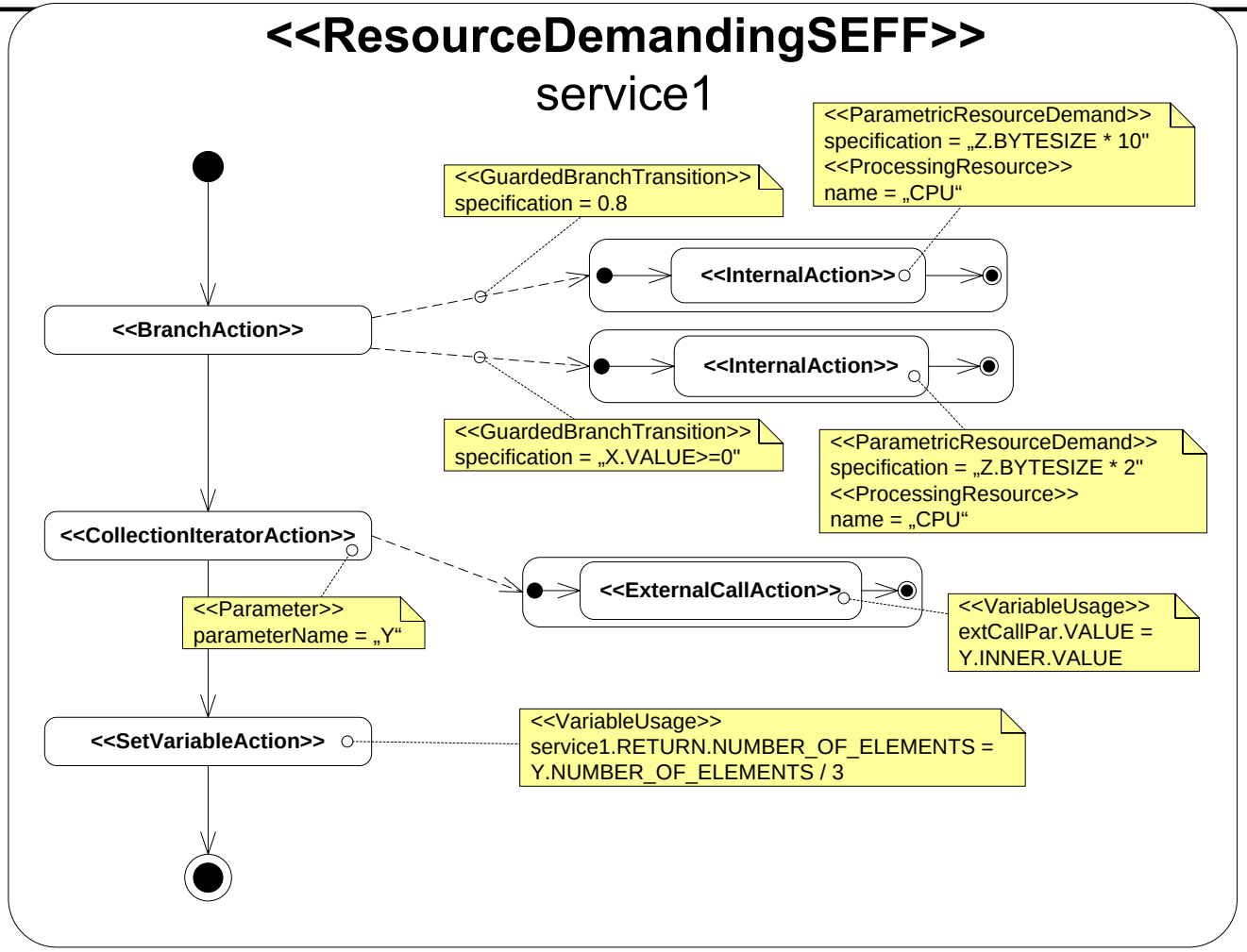


Service Effect Specification

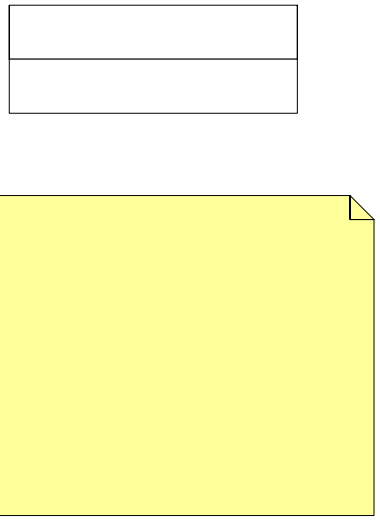


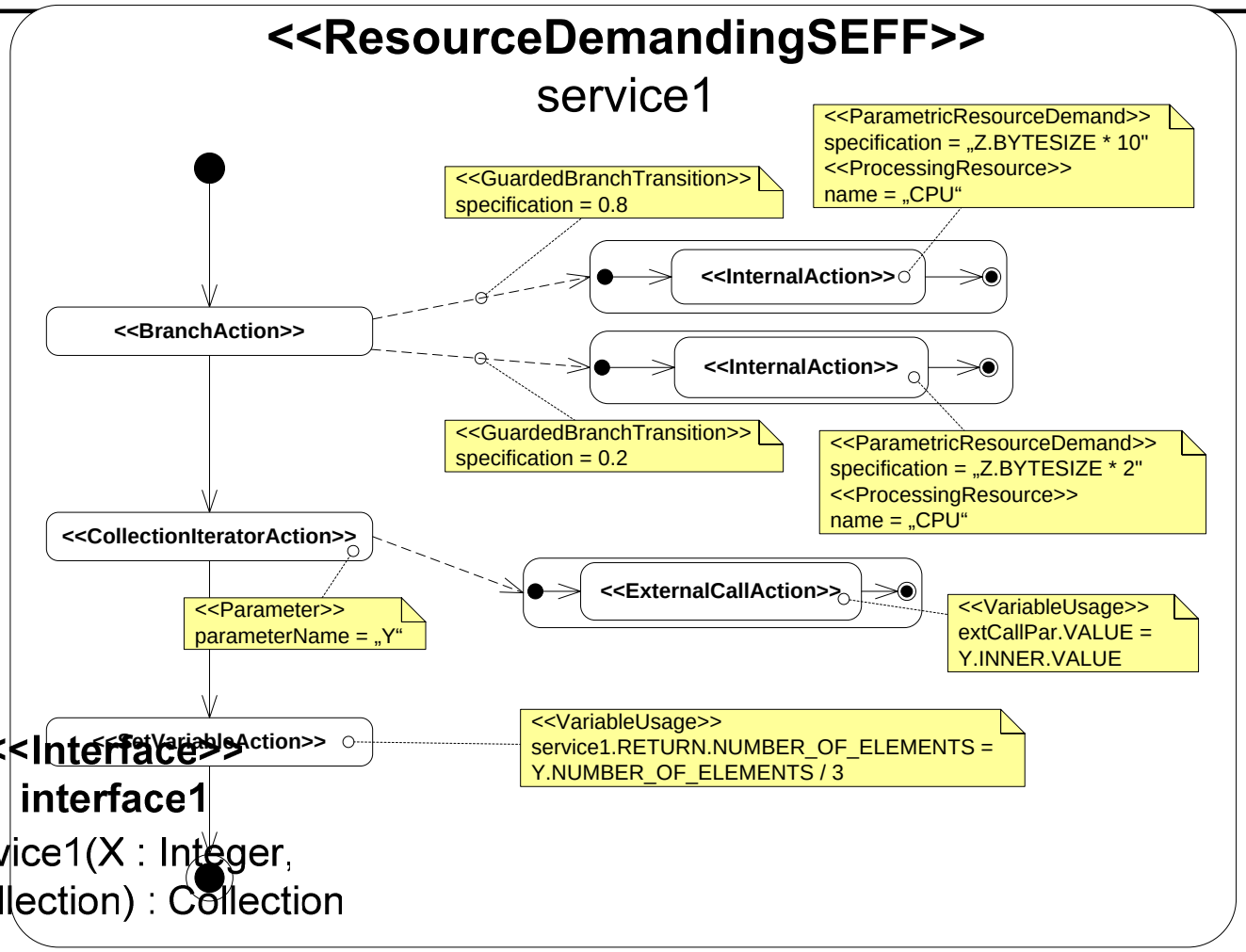
Service Effect Specification





Service Effect Specification

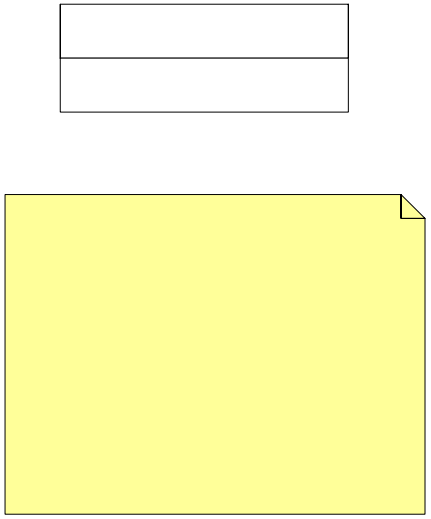


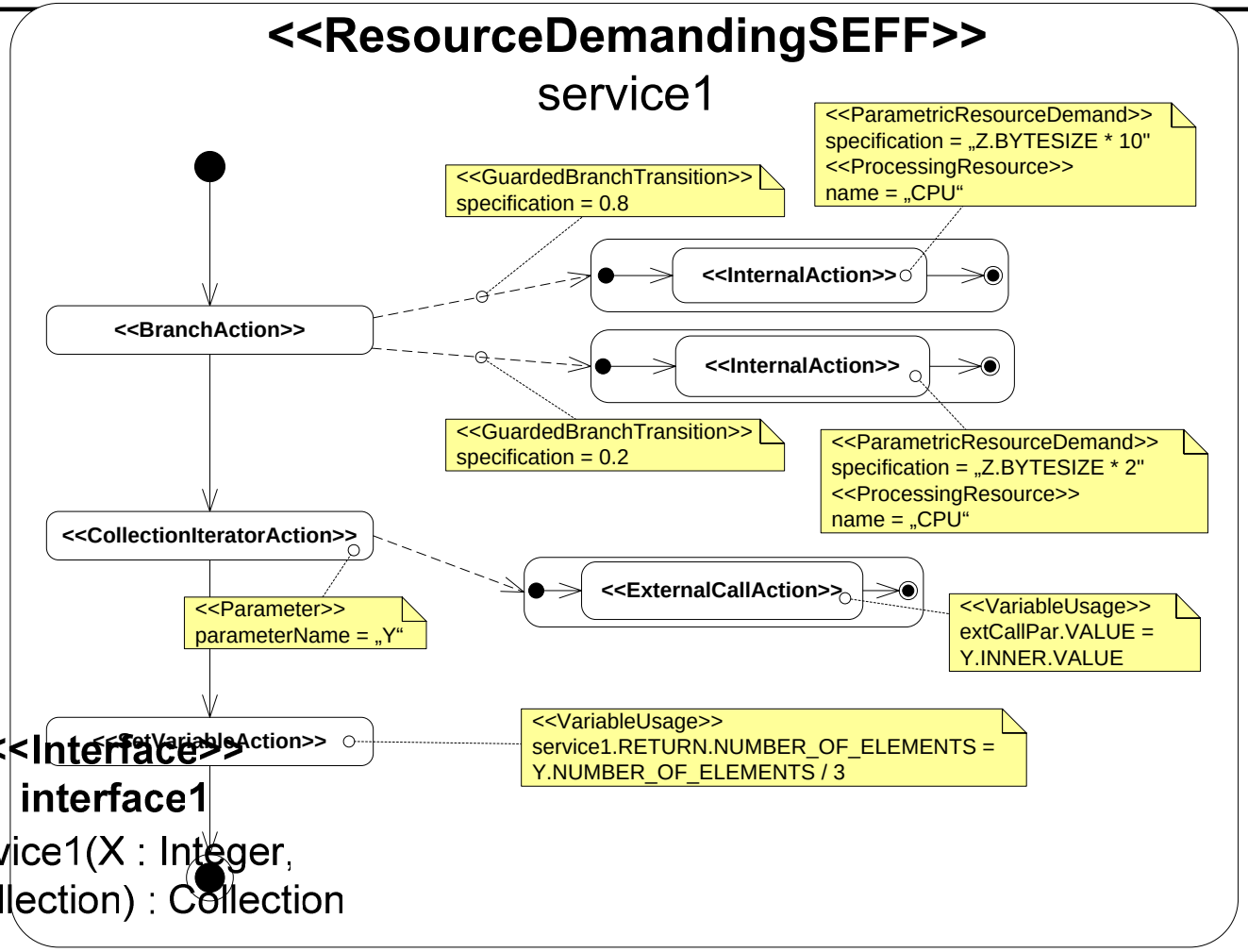


<<Interface>>
interface1
service1(X : Integer,
Y : Collection) : Collection

<<UsageContext>>
P(X.VALUE = -5) = 0.8

Service Effect Specification

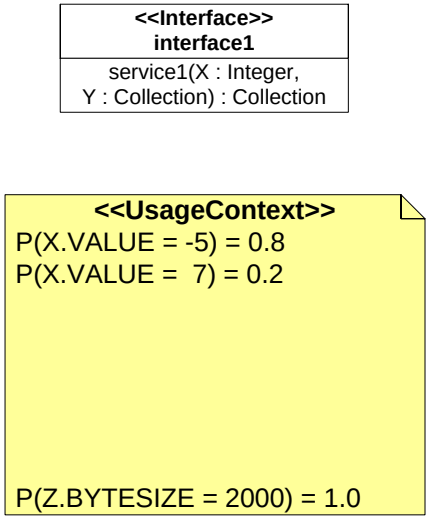


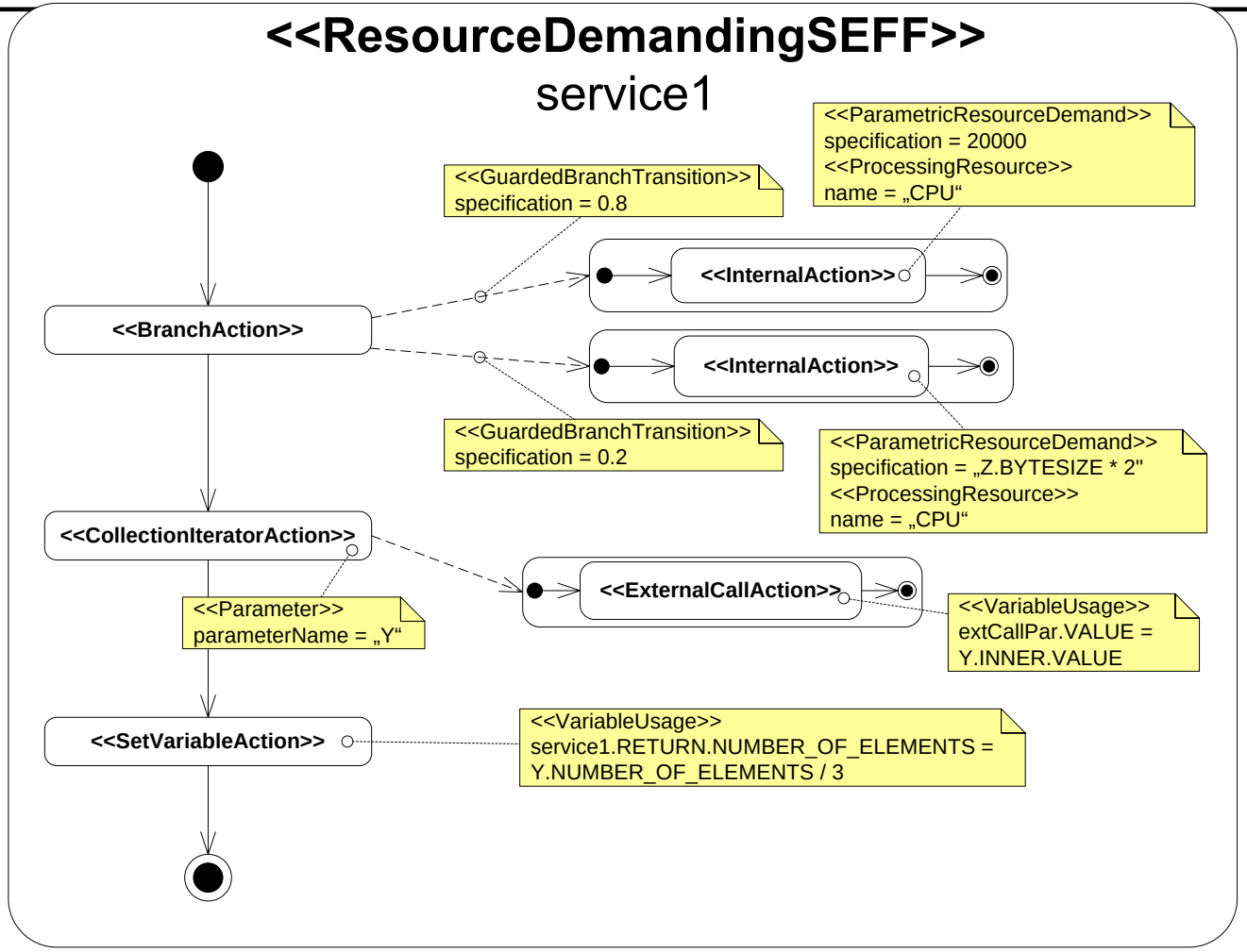


<<Interface>>
interface1
service1(X : Integer,
Y : Collection) : Collection

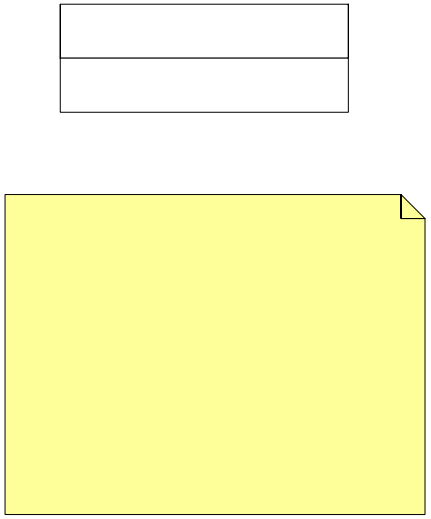
<<UsageContext>>
P(X.VALUE = -5) = 0.8

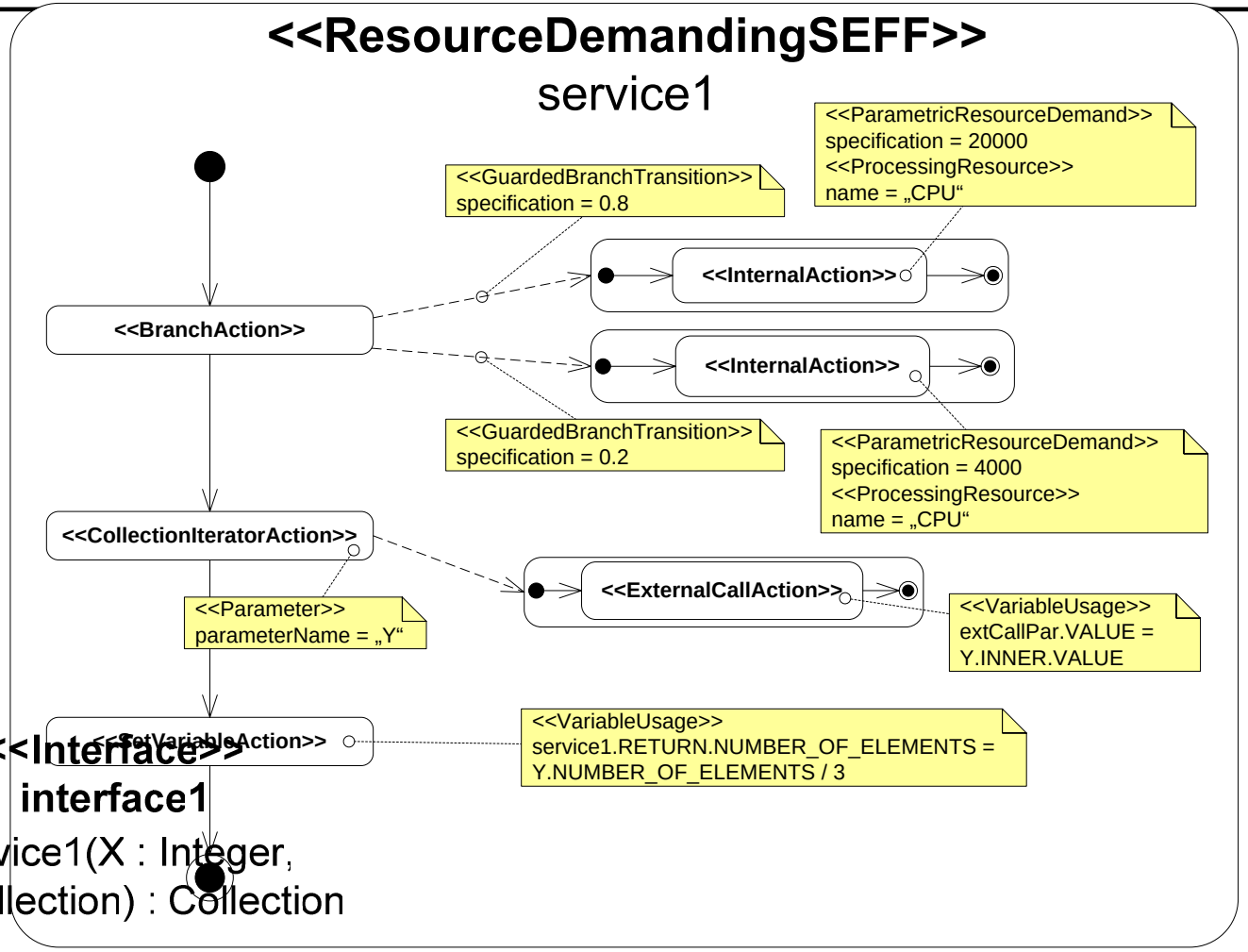
Service Effect Specification





Service Effect Specification

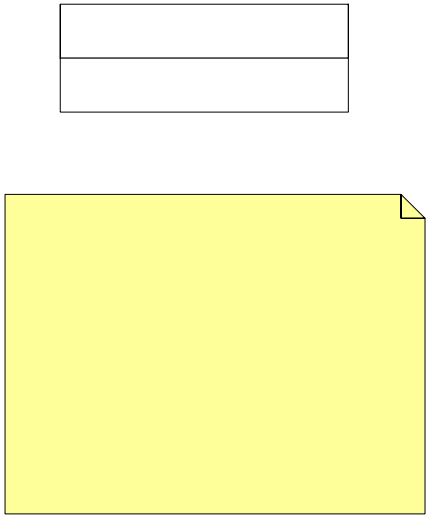


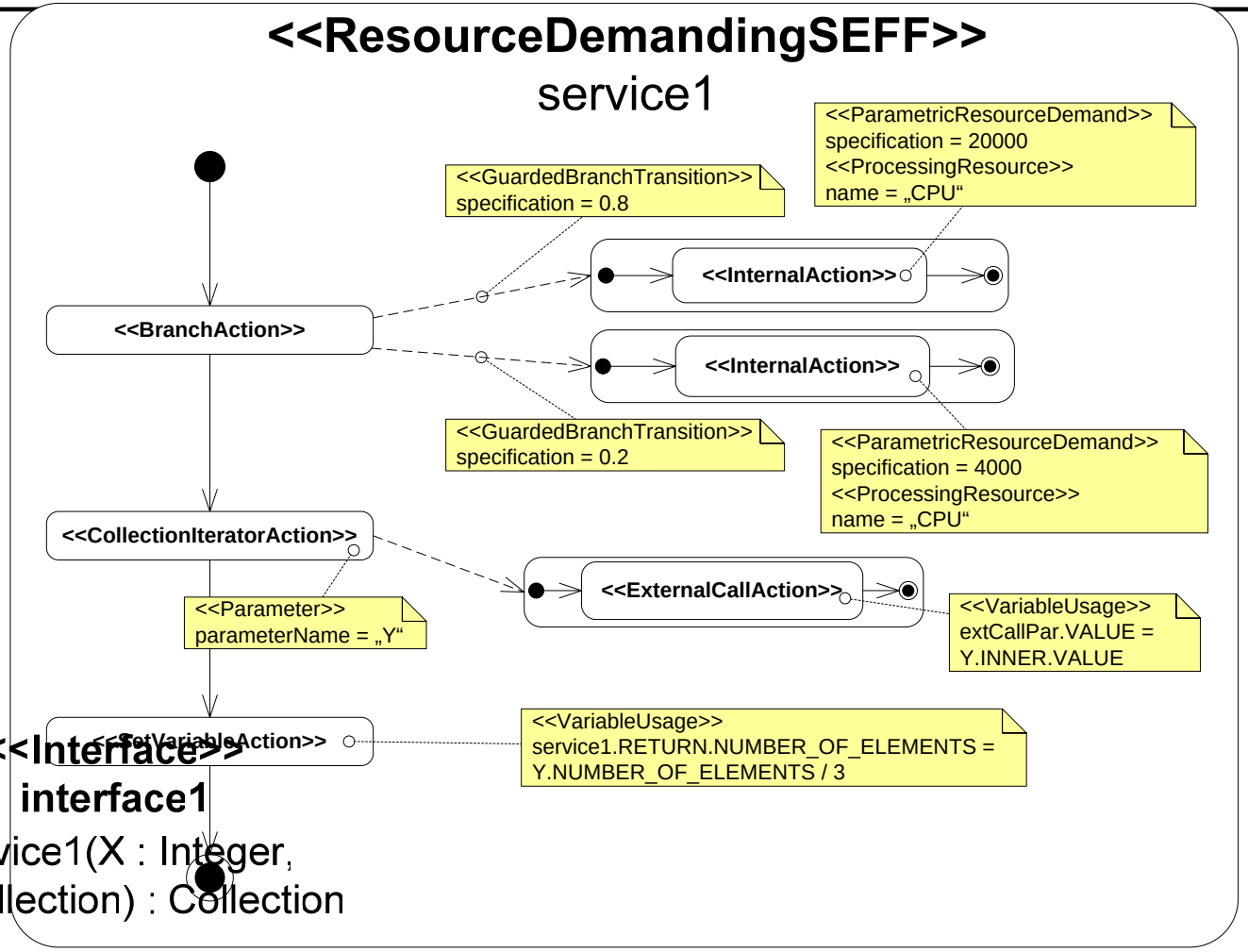


<<Interface>>
interface1
service1(X : Integer,
Y : Collection) : Collection

<<UsageContext>>
P(X.VALUE = -5) = 0.8

Service Effect Specification

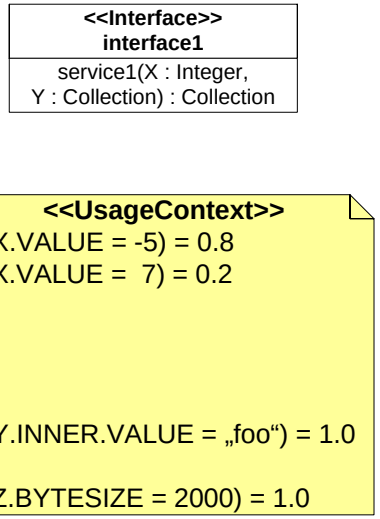


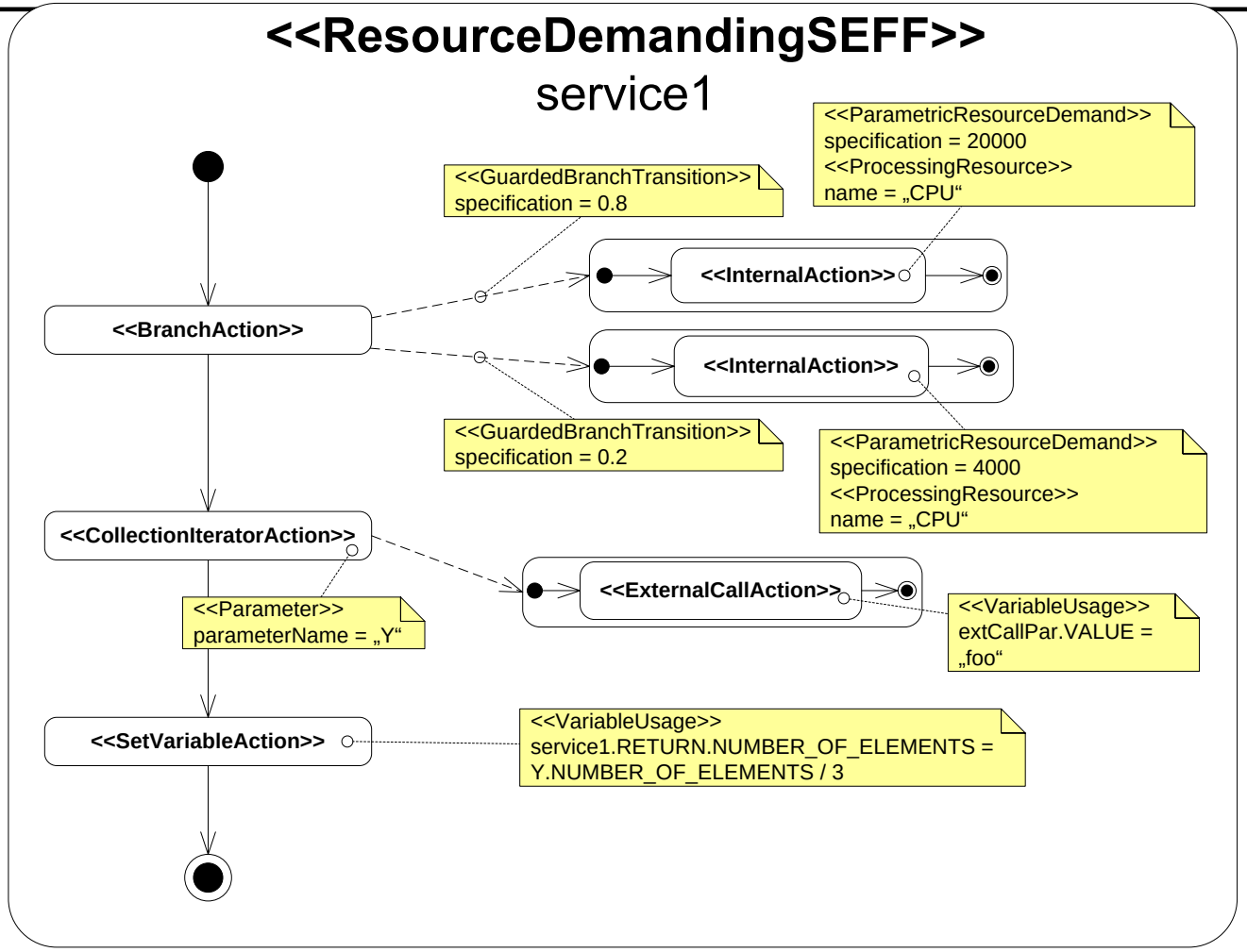


<<Interface>>
interface1
service1(X : Integer,
Y : Collection) : Collection

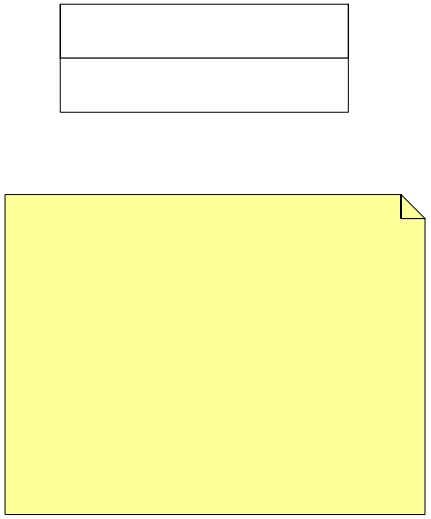
<<UsageContext>>
P(X.VALUE = -5) = 0.8

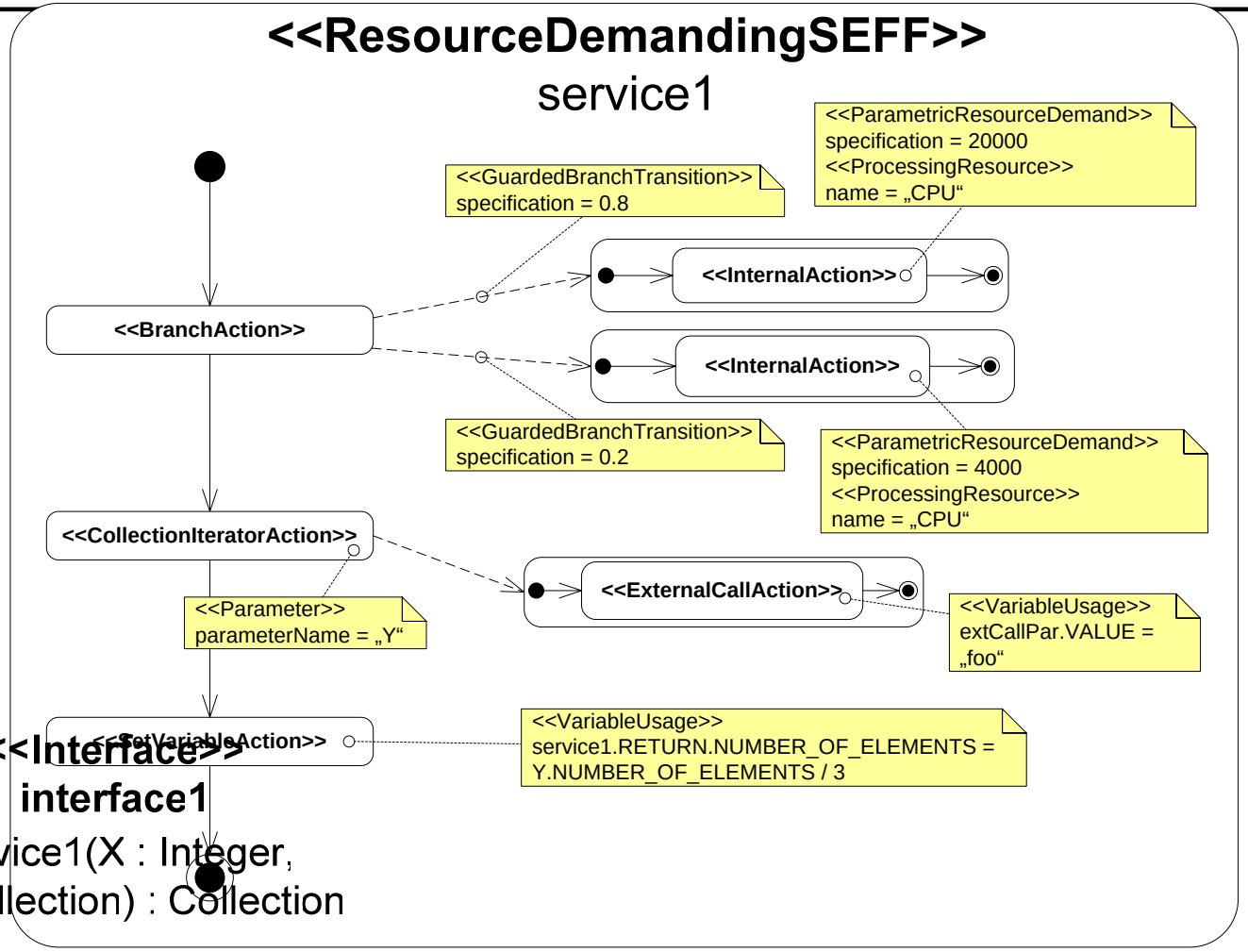
Service Effect Specification





Service Effect Specification

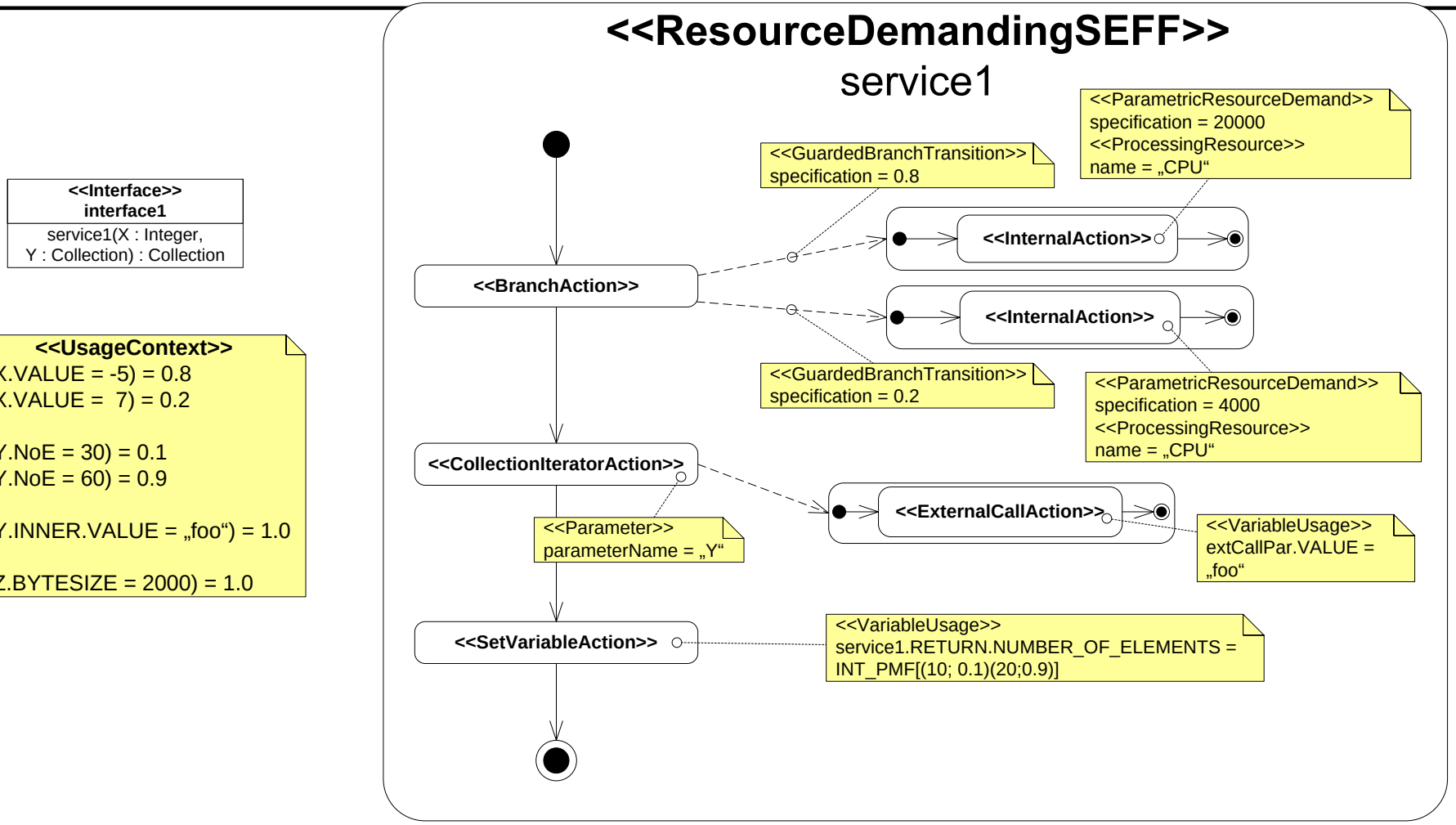




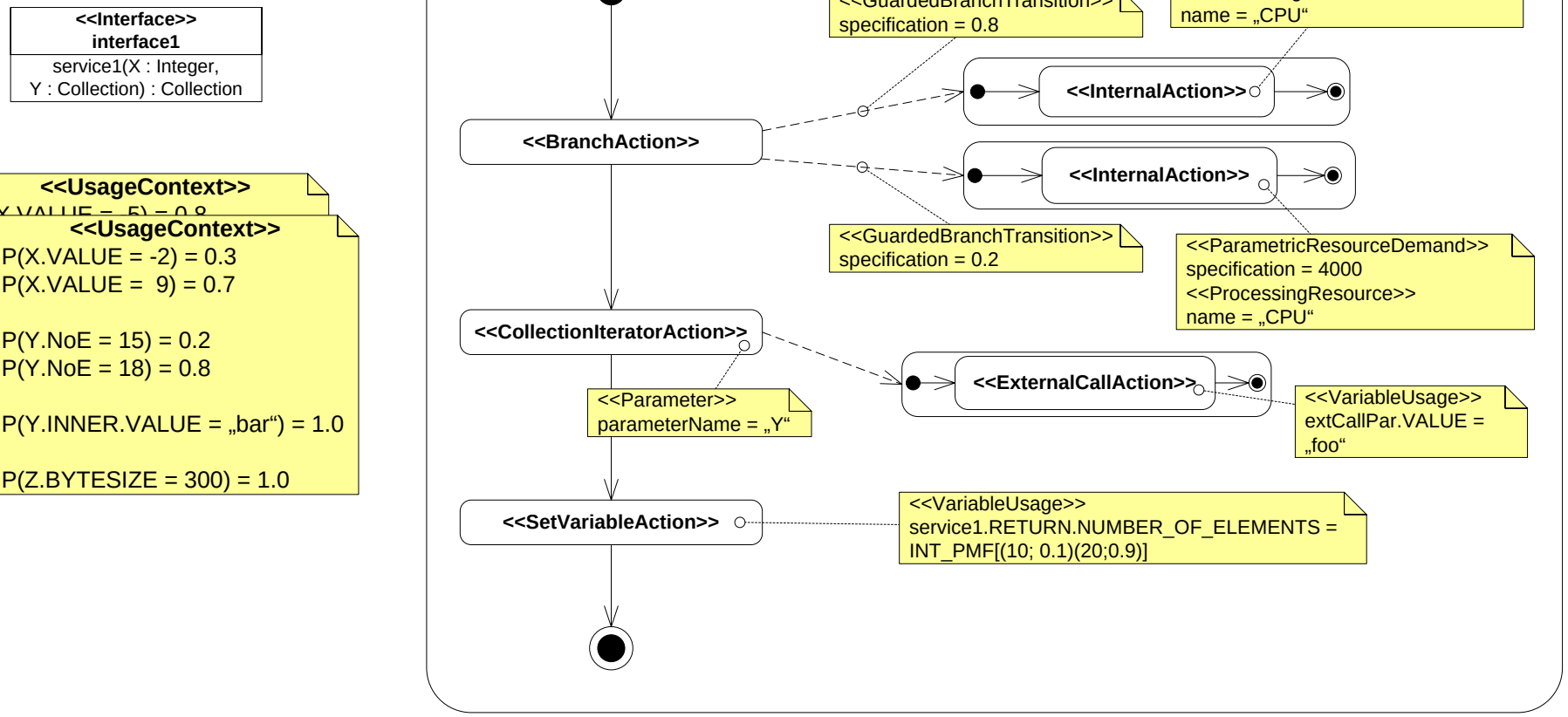
<<Interface>>
interface1
service1(X : Integer,
Y : Collection) : Collection

<<UsageContext>>
P(X.VALUE = -5) = 0.8

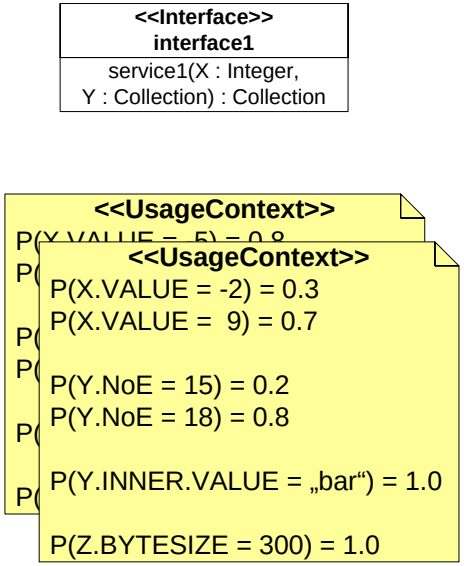
Service Effect Specification

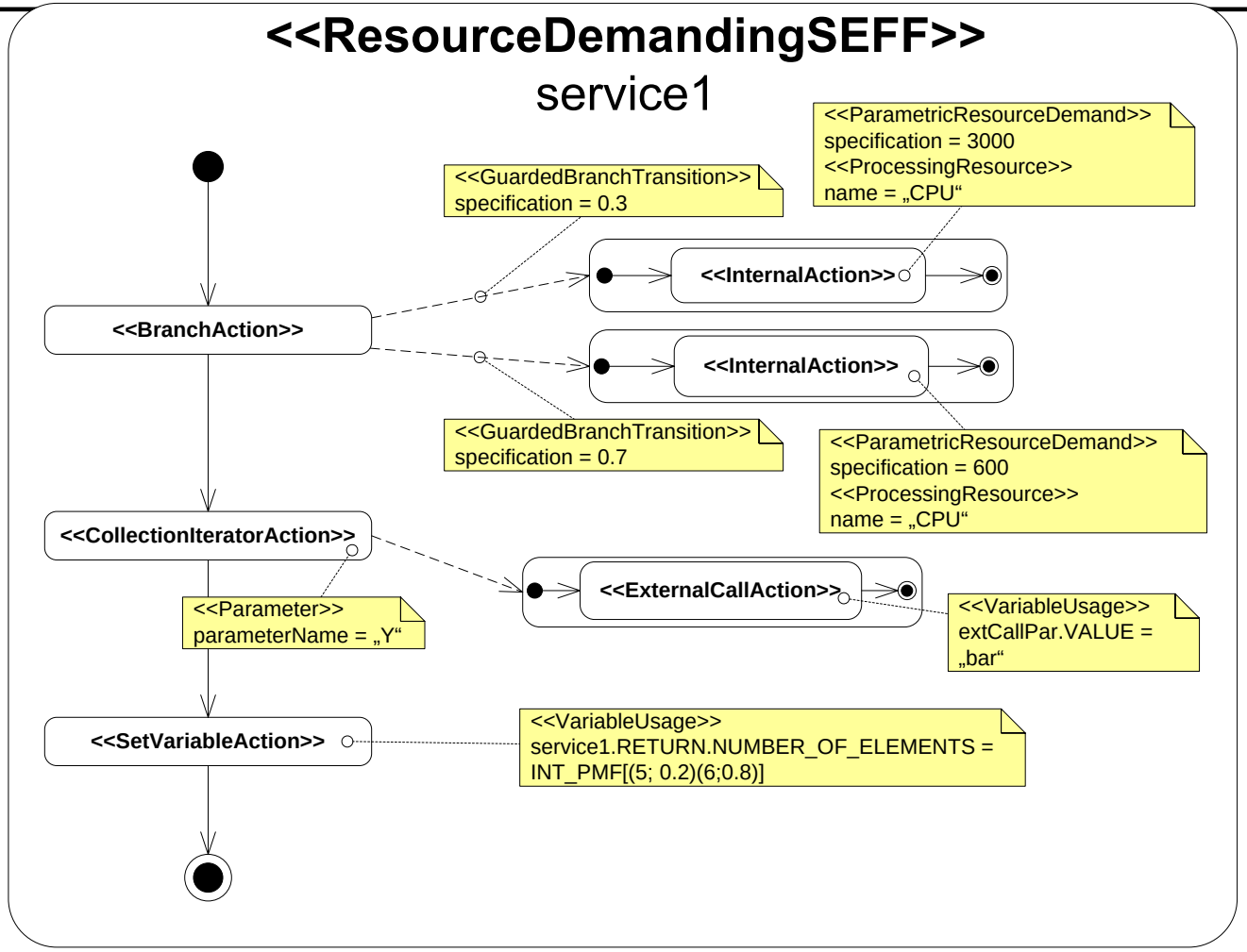


Service Effect Specification

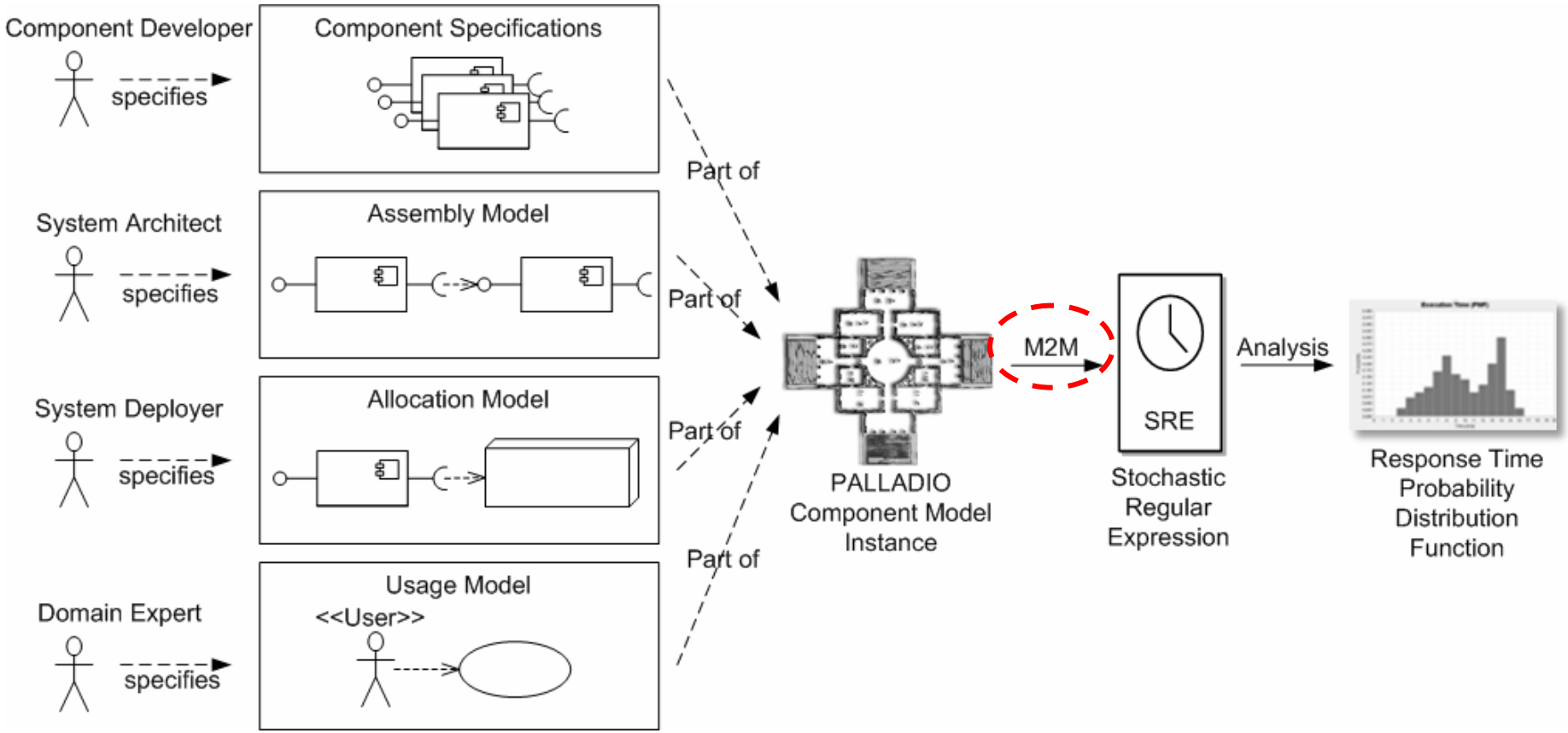


Service Effect Specification





Palladio Component Model



Stochastic Regular Expression

Sequence

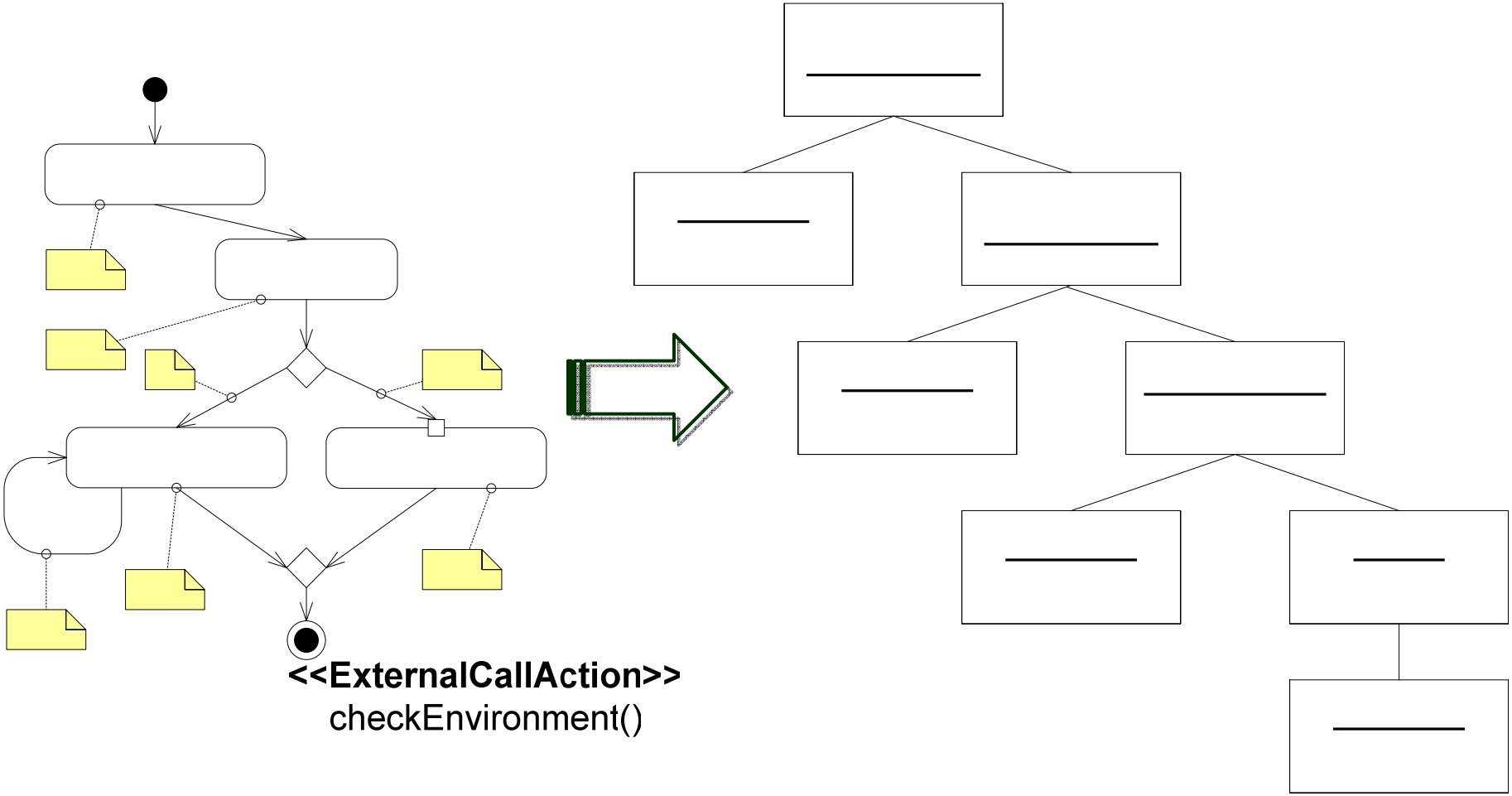
Loop Iterations (l)

$$P := a \;\mid\; P \cdot Q \;\mid\; P +_{\pi} Q \;\mid\; P^{*(l)}$$

Symbol

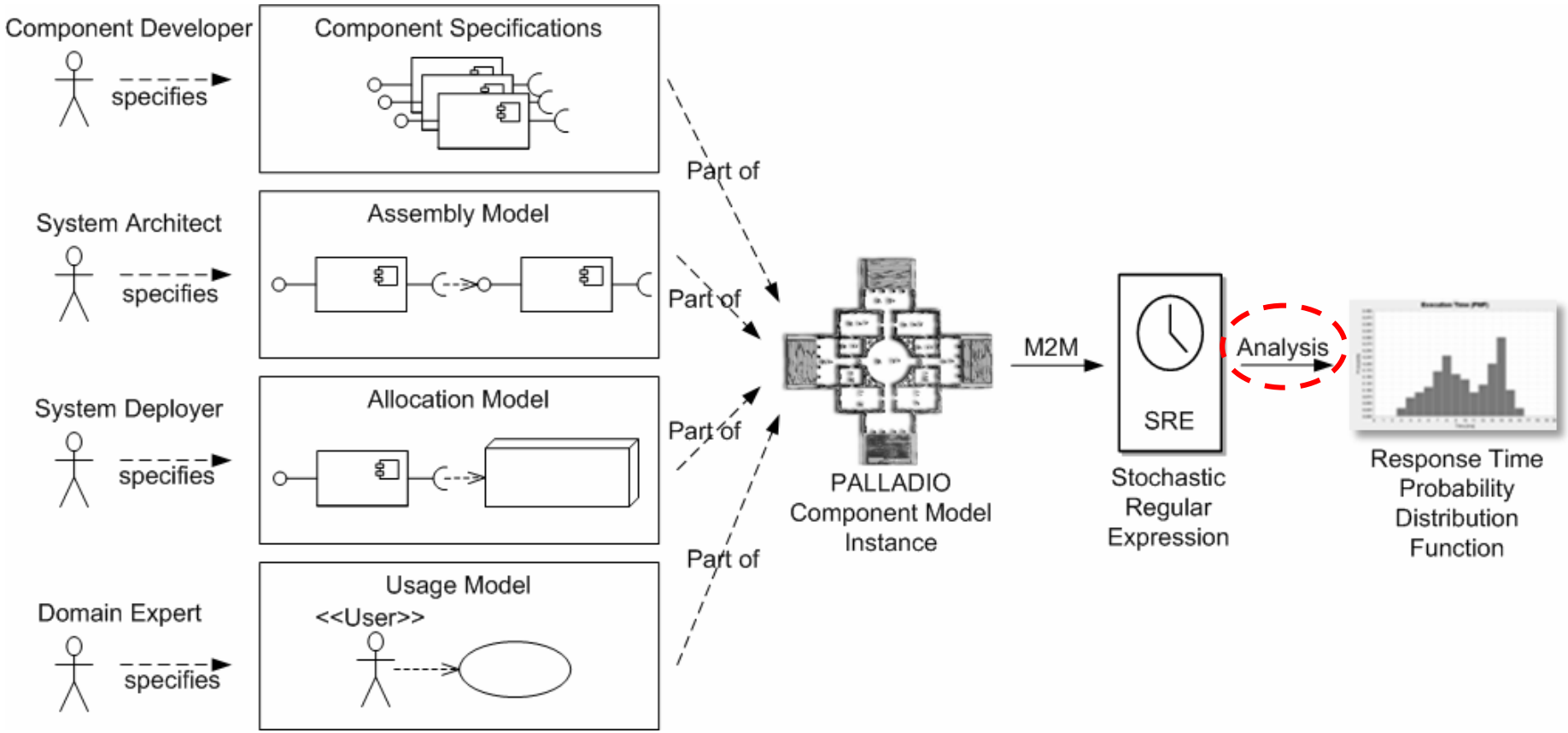
Alternative Probability π, 1-π

Transformation

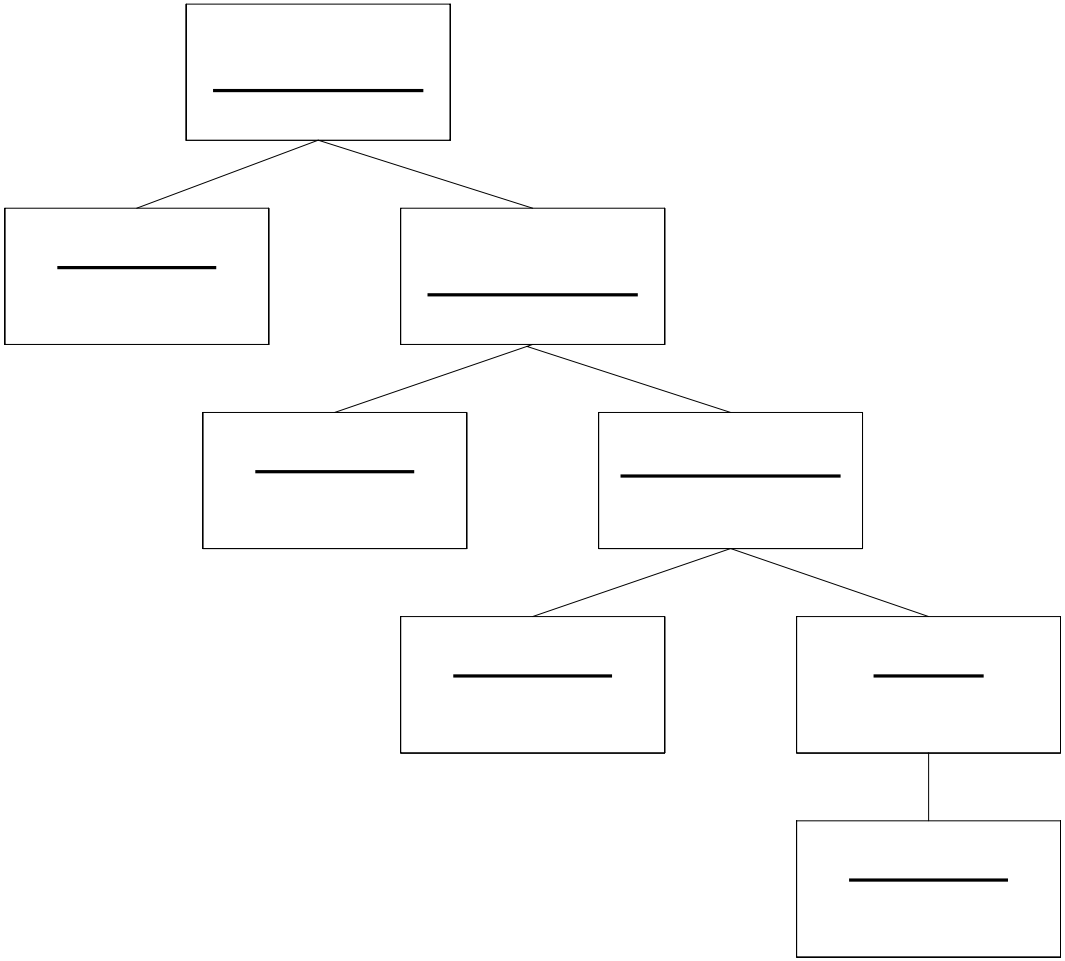


:Sym
time =

Palladio Component Model



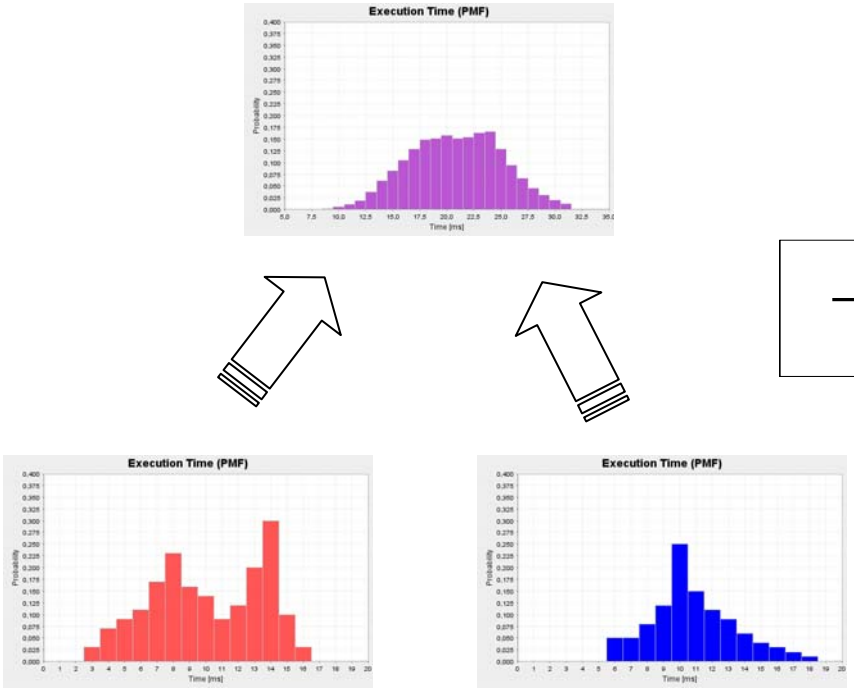
Model Solution

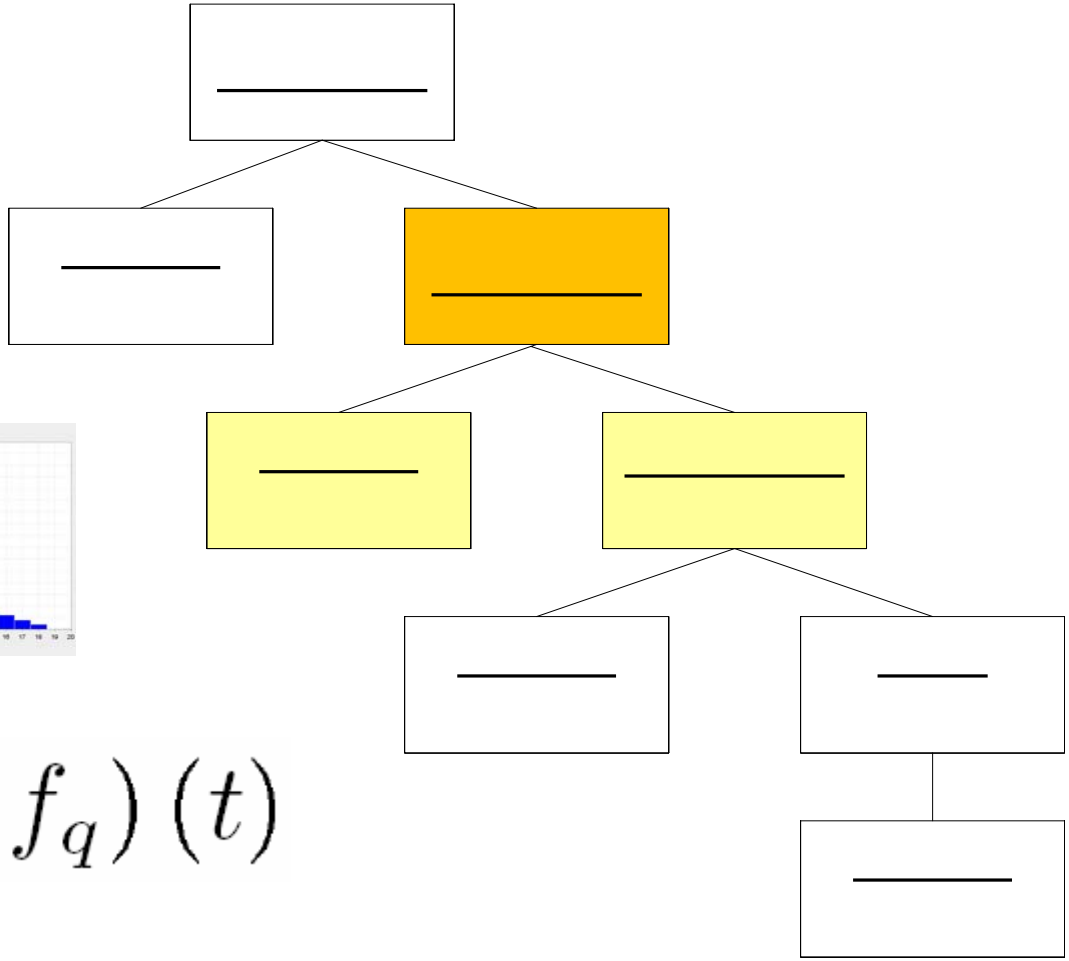


:Sym

time =

Model Solution



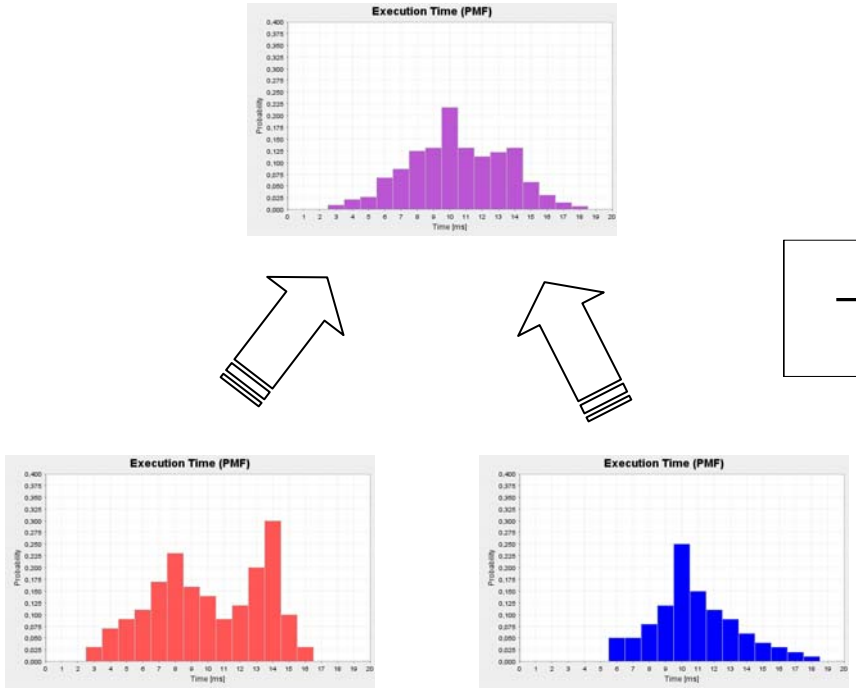


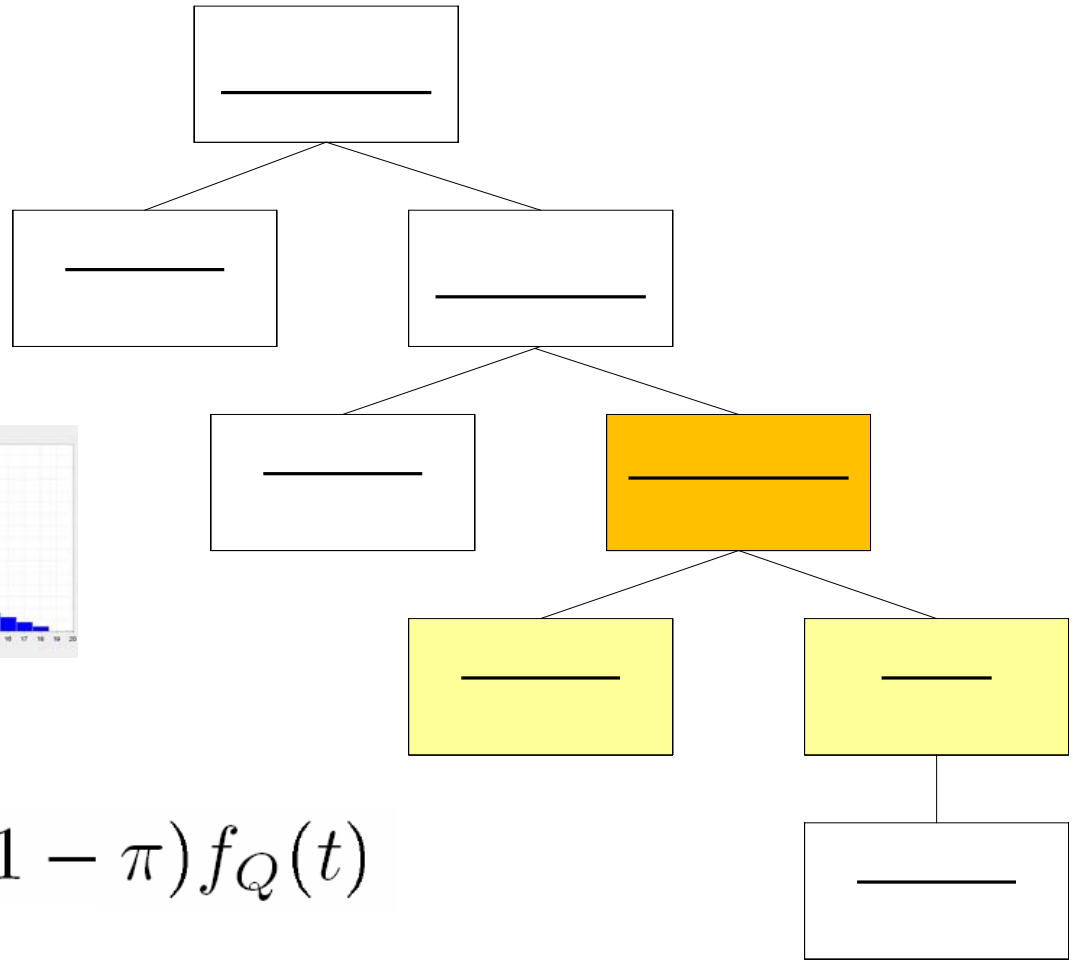
$$f_{P \cdot Q}(t) = (f_p \circledast f_q)(t)$$

:Sym

time =

Model Solution



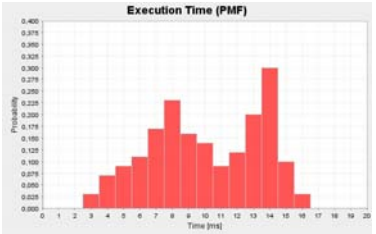


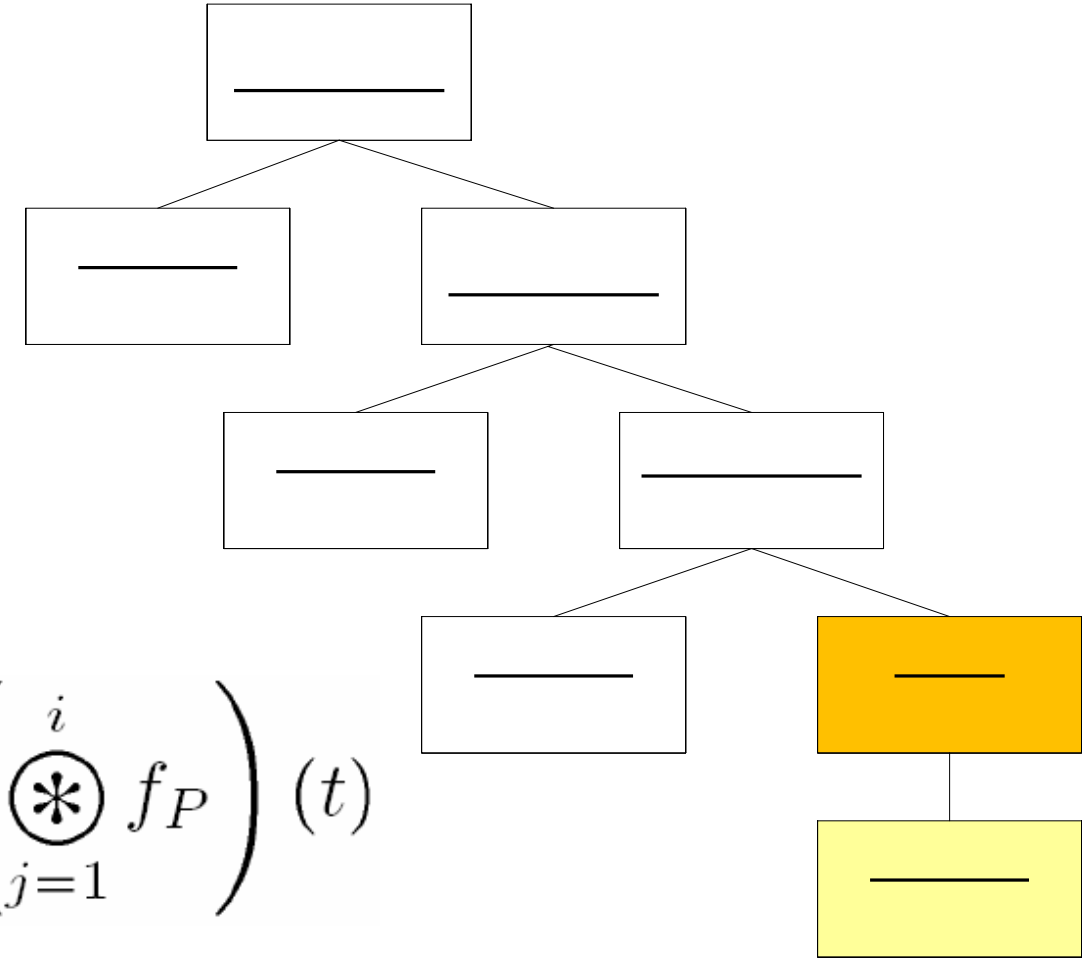
$$f_{P+_\pi Q}(t) = \pi f_P(t) + (1-\pi) f_Q(t)$$

:Sym

time =

Model Solution



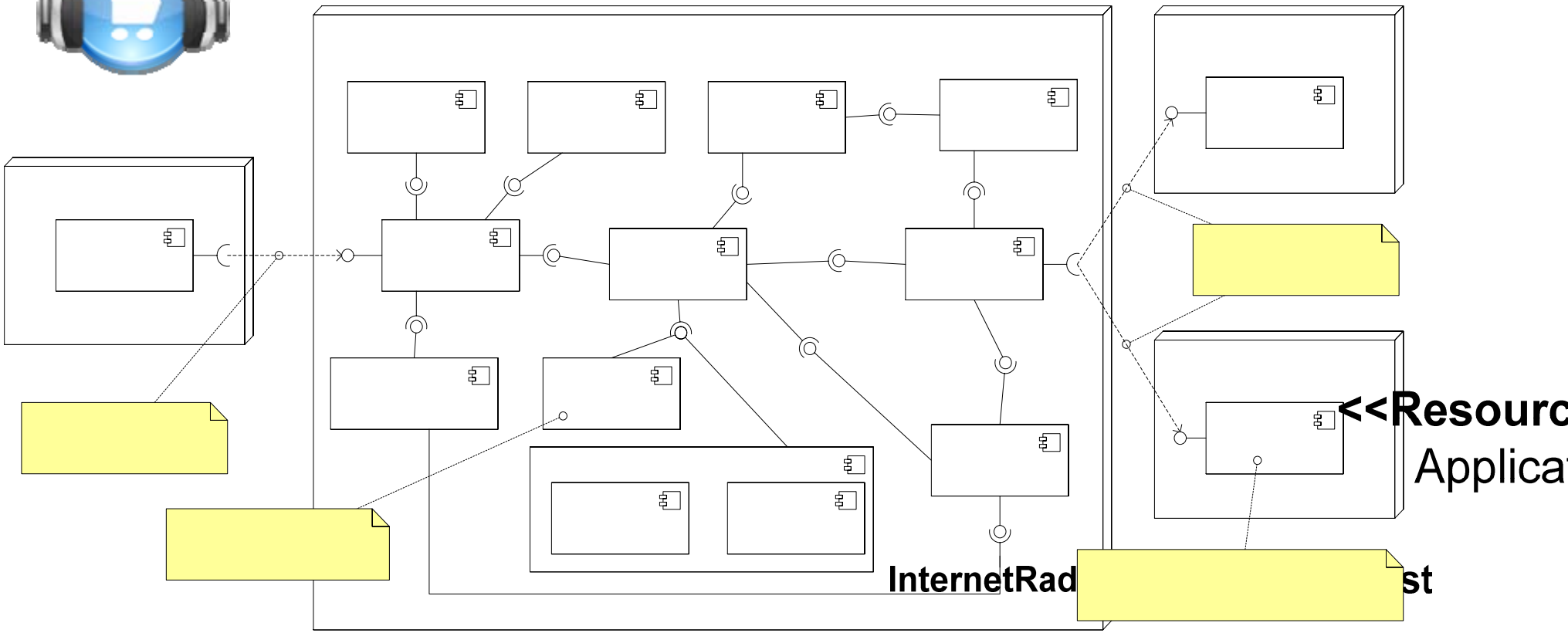


$$f_{P*(l)}(t) = \sum_{i=0}^{N} p_l(i) \left(\mathop{\circledast}_{j=1}^{i} f_P \right)(t)$$

:Sym

time =

MediaStore - Architecture



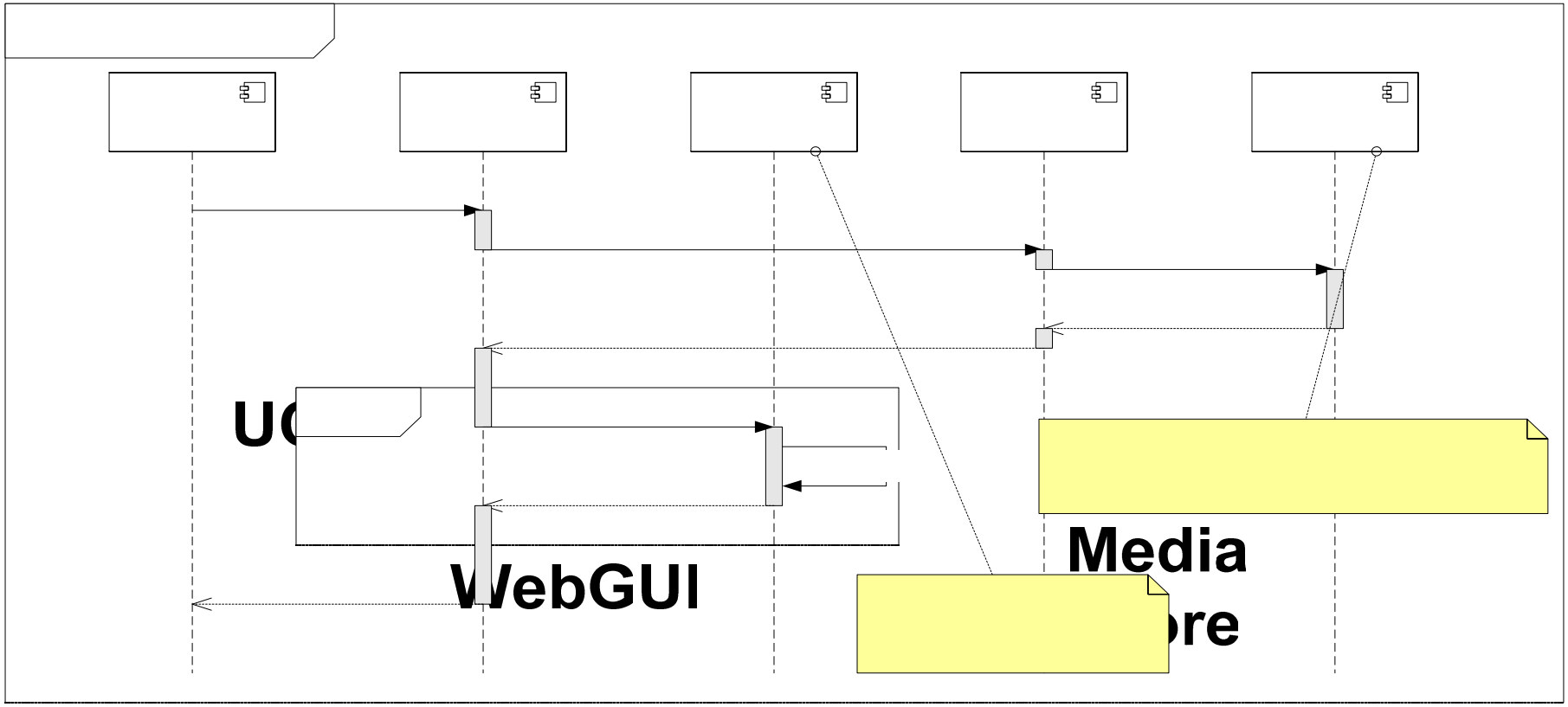
<<ResourceContainer>>
Client

WebBrowser

WebGUI

MediaSt

Download - Use Case



dowload(Files)

Wat
mark

queryDB(

Case Study: Usage Profiles

Parameter Characterisation

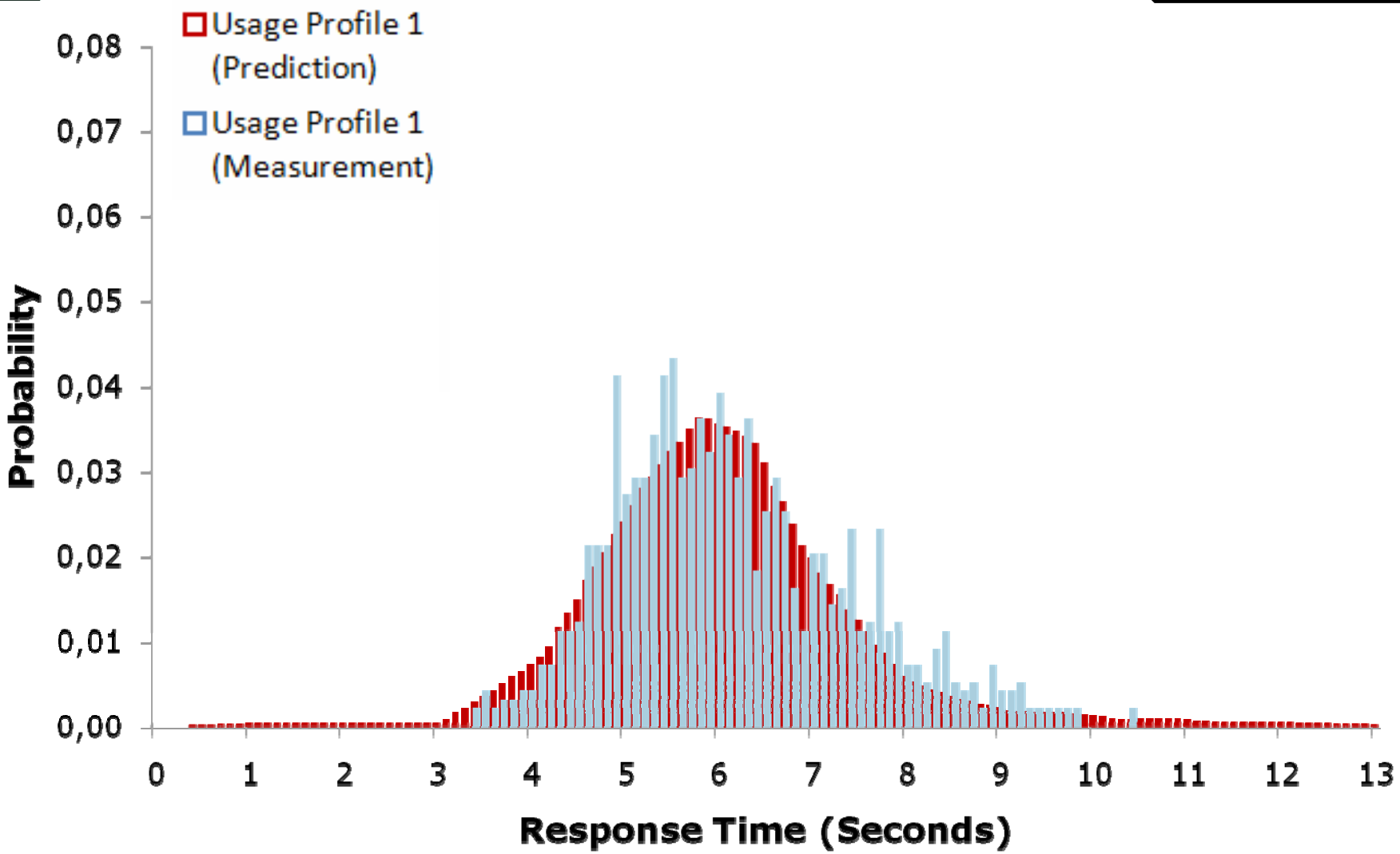
==========

DesiredFiles.NUMBER_OF_ELEMENTS

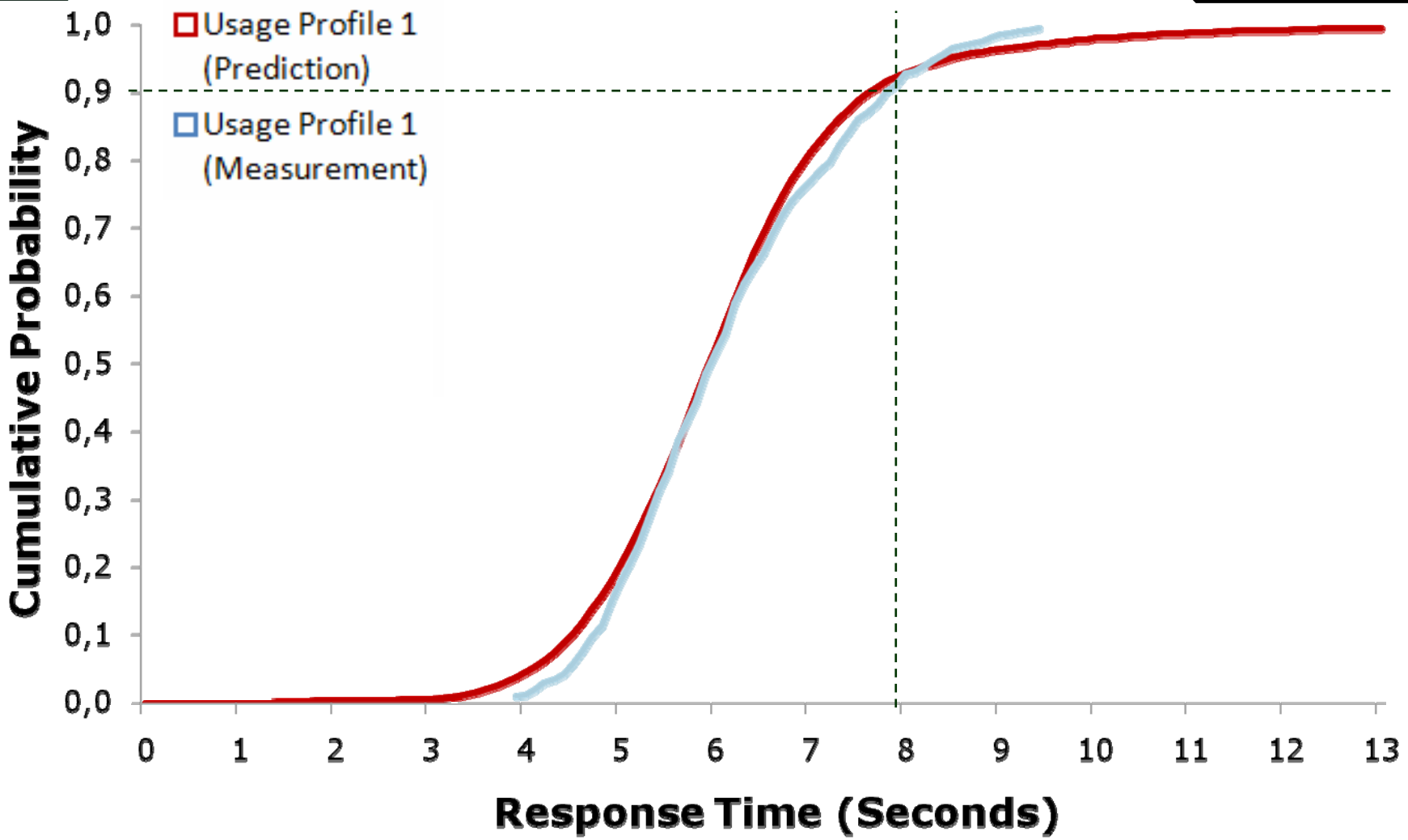
StoredFiles.NUMBER_OF_ELEMENTS

StoredFiles.INNER_BYTESIZE

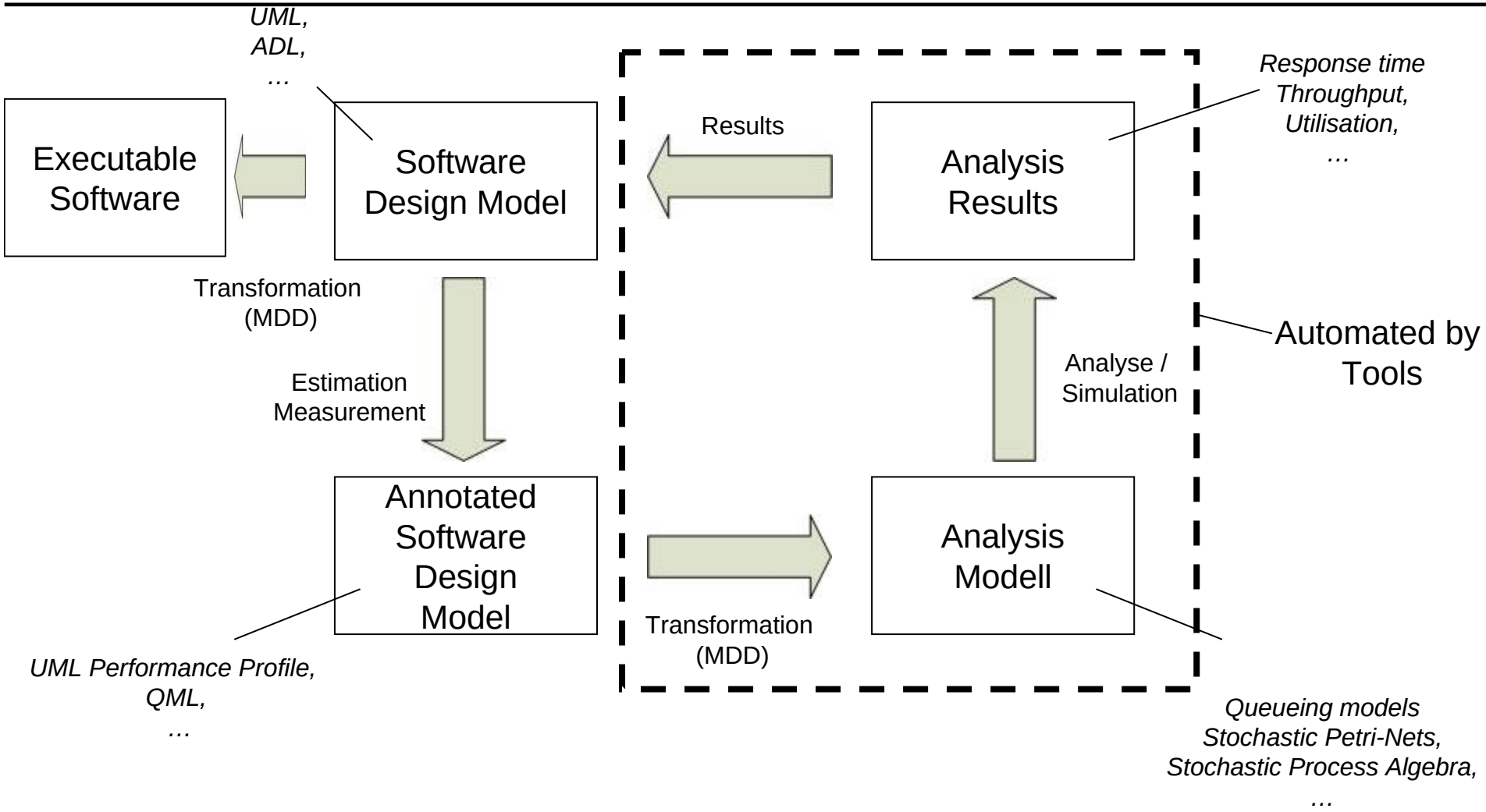
Results


Usage Profile 1 (Prediction)
Usage Profile 1 (Measurement)
Probability
0,08
0,07
0,06
0,05
0,04
0,03
0,02
0,01
0,00
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Response Time (Seconds)

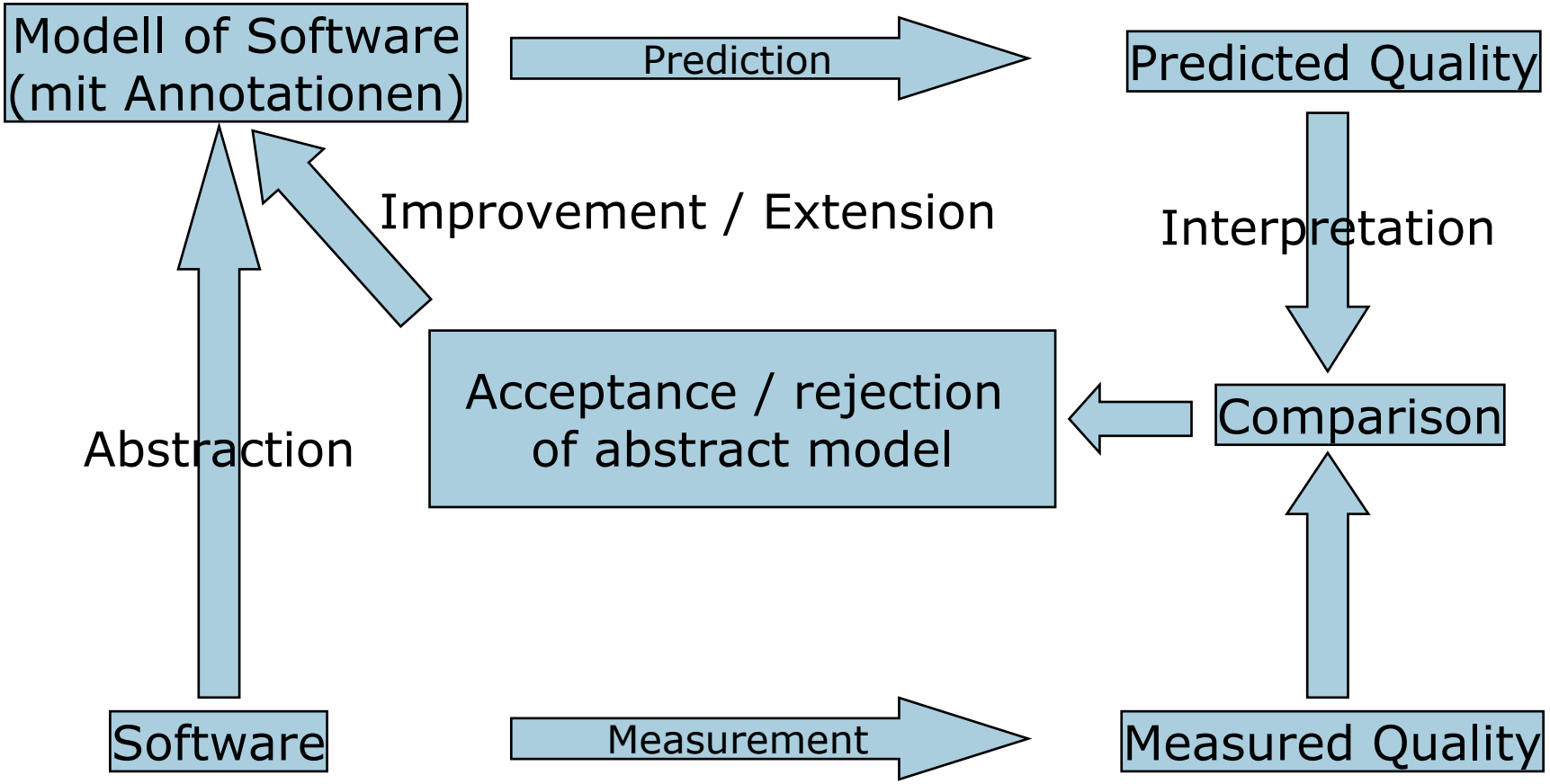
Results


Usage Profile 1 (Prediction)
Usage Profile 1 (Measurement)
Cumulative Probability
1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Response Time (Seconds)

Model-based Prediction of Quality



Scientific Approach



Summary

- Contractual Use of Components
  - Requires-interfaces as pre-conditions
  - Provides-interfaces as post-conditions
  - Enabling (static) interoperability checks
- Parametric Contracts
  - Compute pre-condition in dependency of post-condition and vice versa
  - Prediction of properties in non-cyclic architectures by propagating interface information
  - Formal models of parametric contracts for predicting quality properties are created in a process oriented toward natural sciences.