Praktikum Ingenieurmäßige Software-Entwicklung

Palladio Component Model – Part III (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. Stochastic Expressions
   a. Constants, PMF, PDF, Parameter Characterisation
   b. Parametric Dependencies

Lecture 1 (1.a – 2.b)

Lecture 2 (2.c)

Lecture 3 (3.a – 3.b)

Uncertainties

- A situation is uncertain
  if the outcome is unknown in advance
- Probabilistic characterisations possible
- Examples
  - How will users interact with a system?
  - When do they arrive?
  - Which parameters do they pass in their calls?

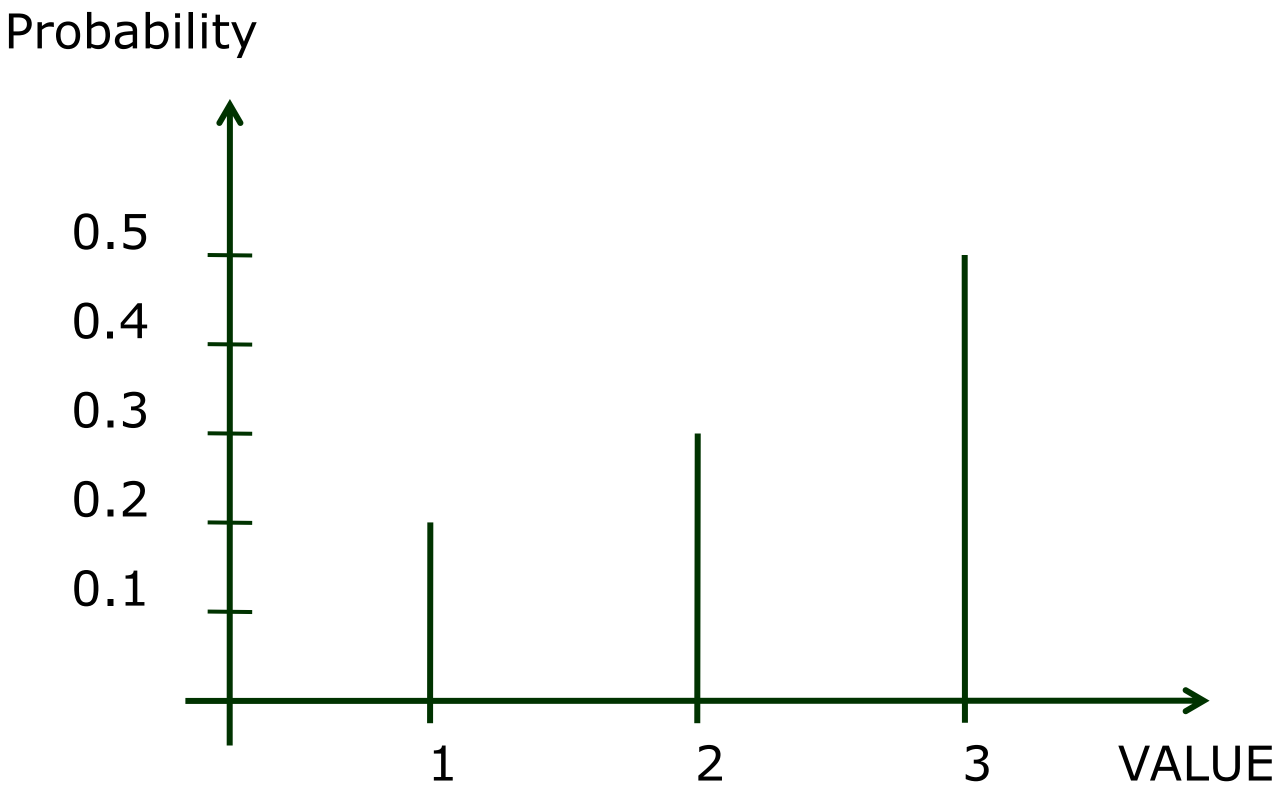
Random Variables

- Random variables describe uncertain events
- They may be described by their probability distribution
- Two kinds of random variables:
  - Discrete
    - Example: Iteration count of a loop
  - Continuous
    - Example: Passed time between the arrival of two jobs

Probability Mass Function

- PMF
- Distribution Function of a discrete variable
- Domain type depends on the model
  - Loop Iterations: Integer
  - Collection Structure: Enum
  - Actual Value: Any
  - ...
- PMF Literals
  - IntPMF[(1;0.1)(2;0.3)(5;0.6)]
  - EnumPMF[(„Sorted“;0.5)(„Unsorted“;0.5)]
- Constraint: Sum of probabilities has to be 1, be careful, this is still unchecked in the tools!

Probability Mass Function



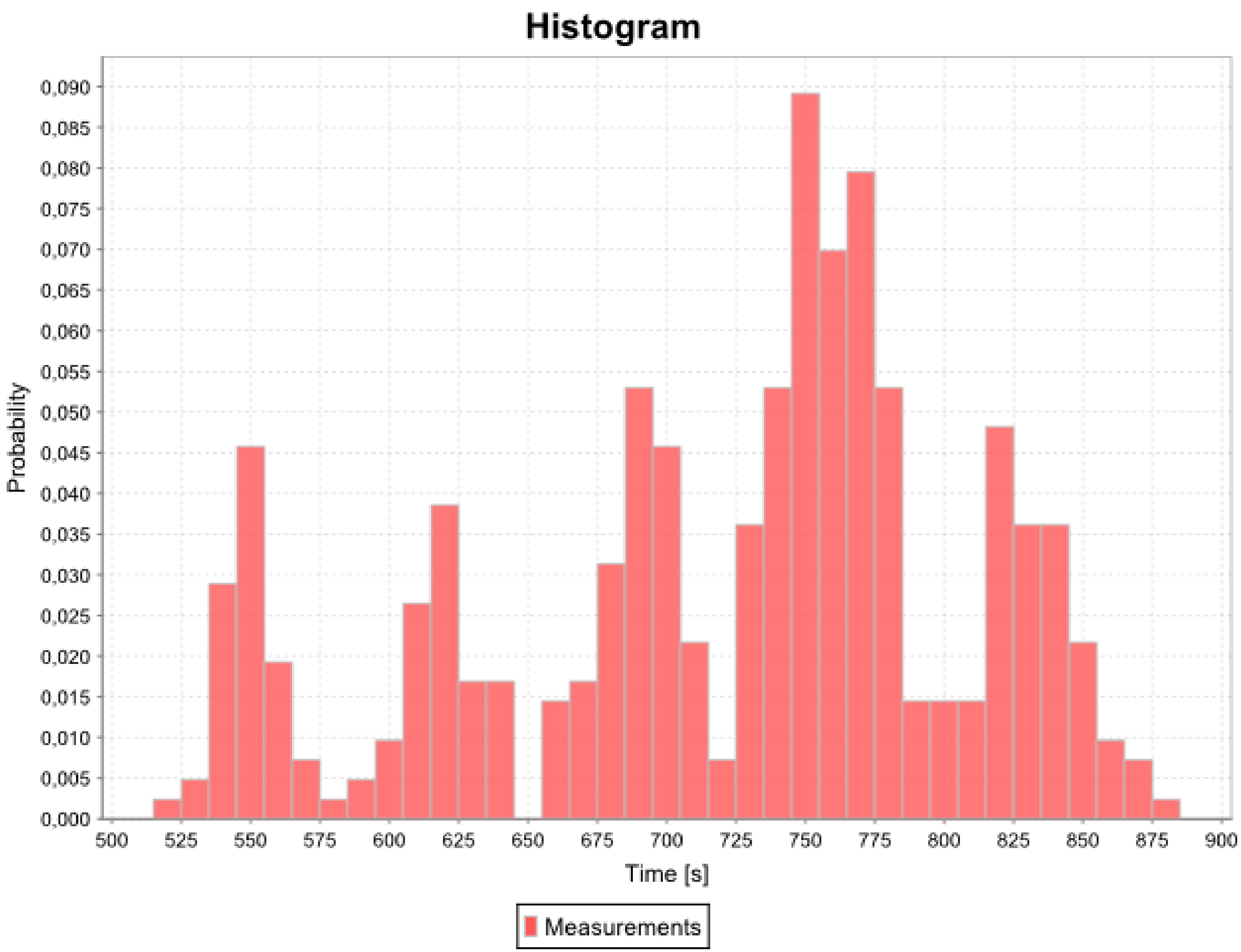
IntPMF[(1;0.2)(2;0.3)(3;0.5)]

Probability Density Function

- PDF
- Dist. Function of a continuous random variable
- Domain is always double
- Hard to characterise as possibly infinite

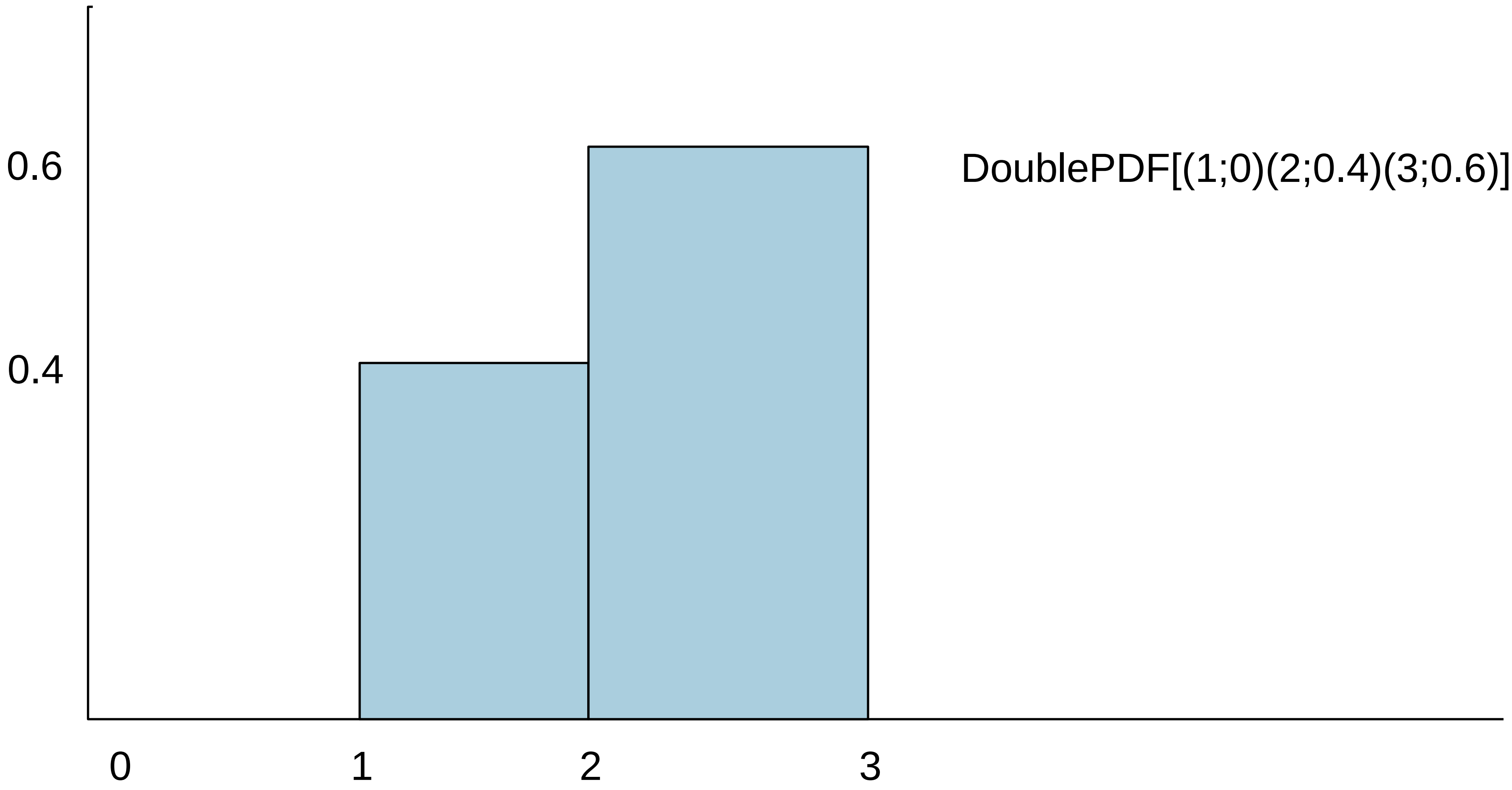
  → We use a derived discrete function: BoxedPDF
- Boxes sum up all events
  falling into their bounds
- Inner box distribution is uniform
- Depicted as histogram or CDF

PDF

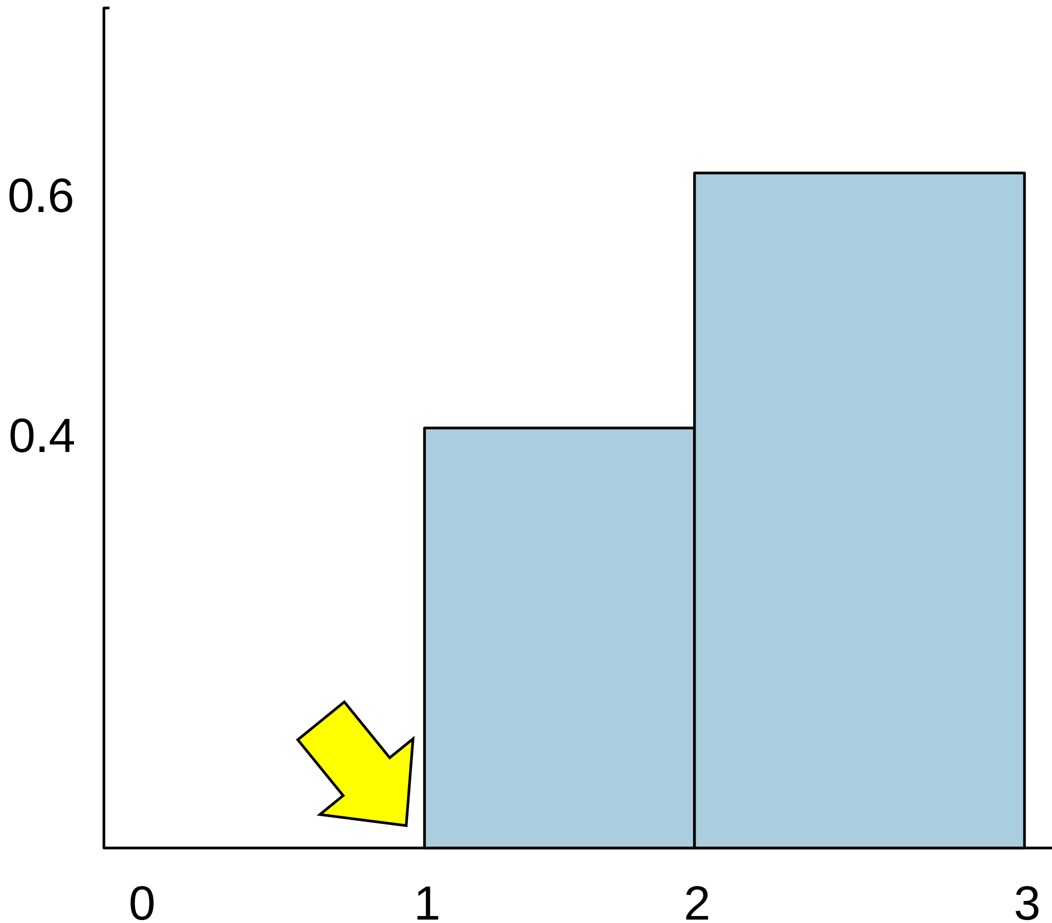


Histogram
0,090
0,085
0,080
0,075
0,070
0,065
0,060
0,055
0,050
0,045
0,040
0,035
0,030
0,025
0,020
0,015
0,010
0,005
0,000
Probability
500
525
550
575
600
625
650
675
700
725
750
775
800
825
850
875
900
Time [s]
Measurements

Specification



Specification

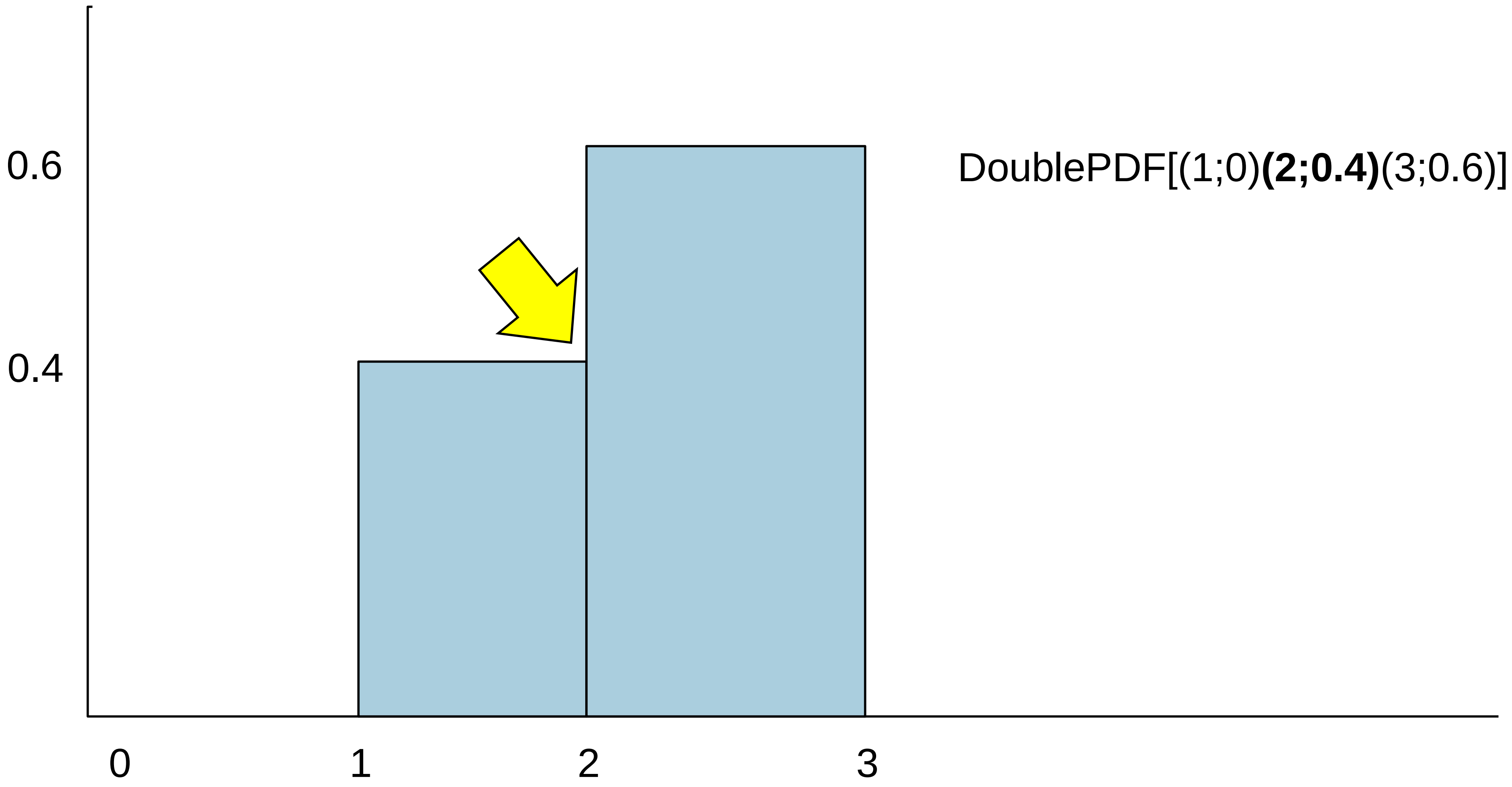


DoublePDF[**(1;0)**(2;0.4)(3;0.6)]

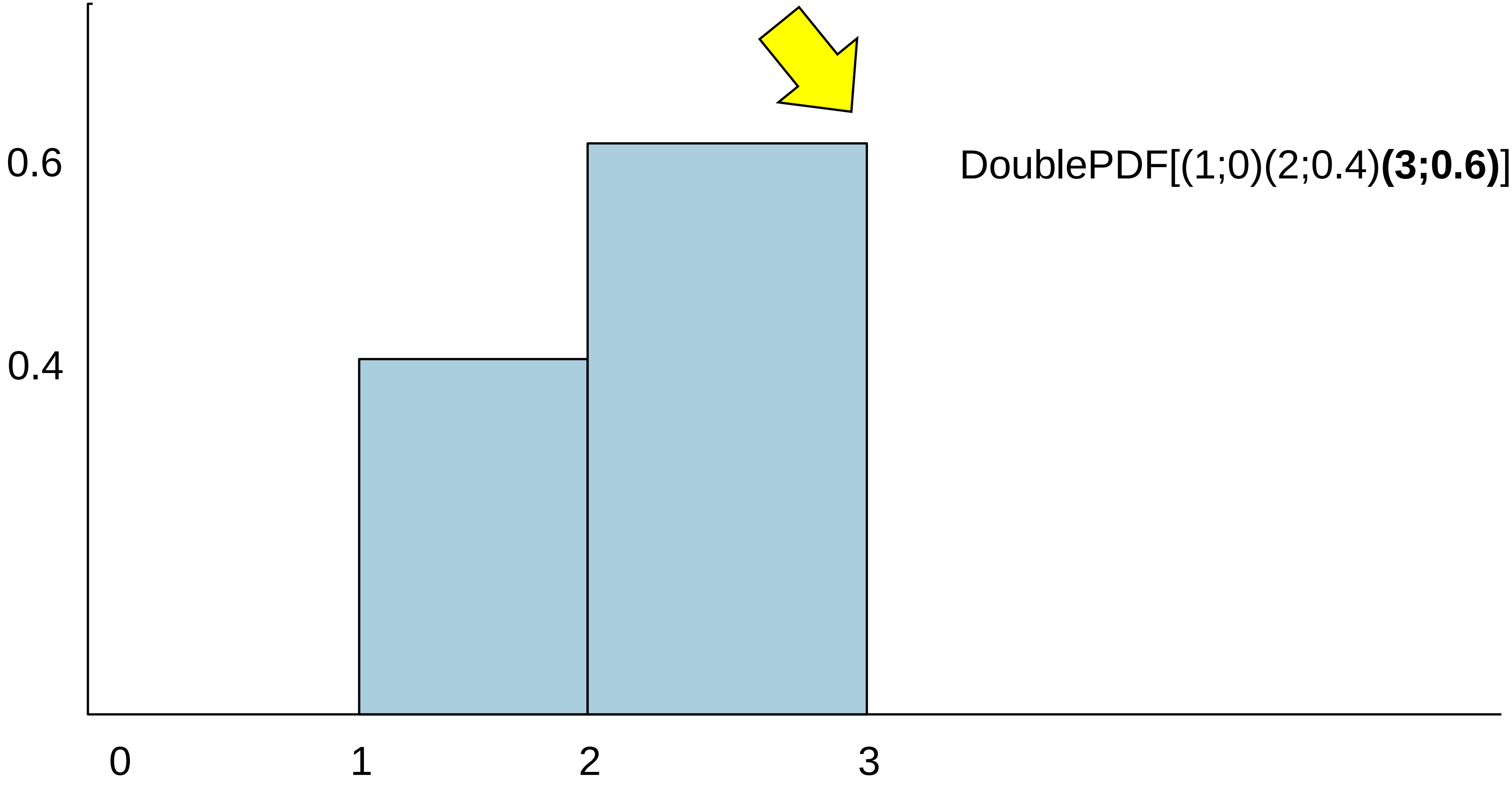
Note:

As first value, implicitly (0,0) is always assumed!

Specification

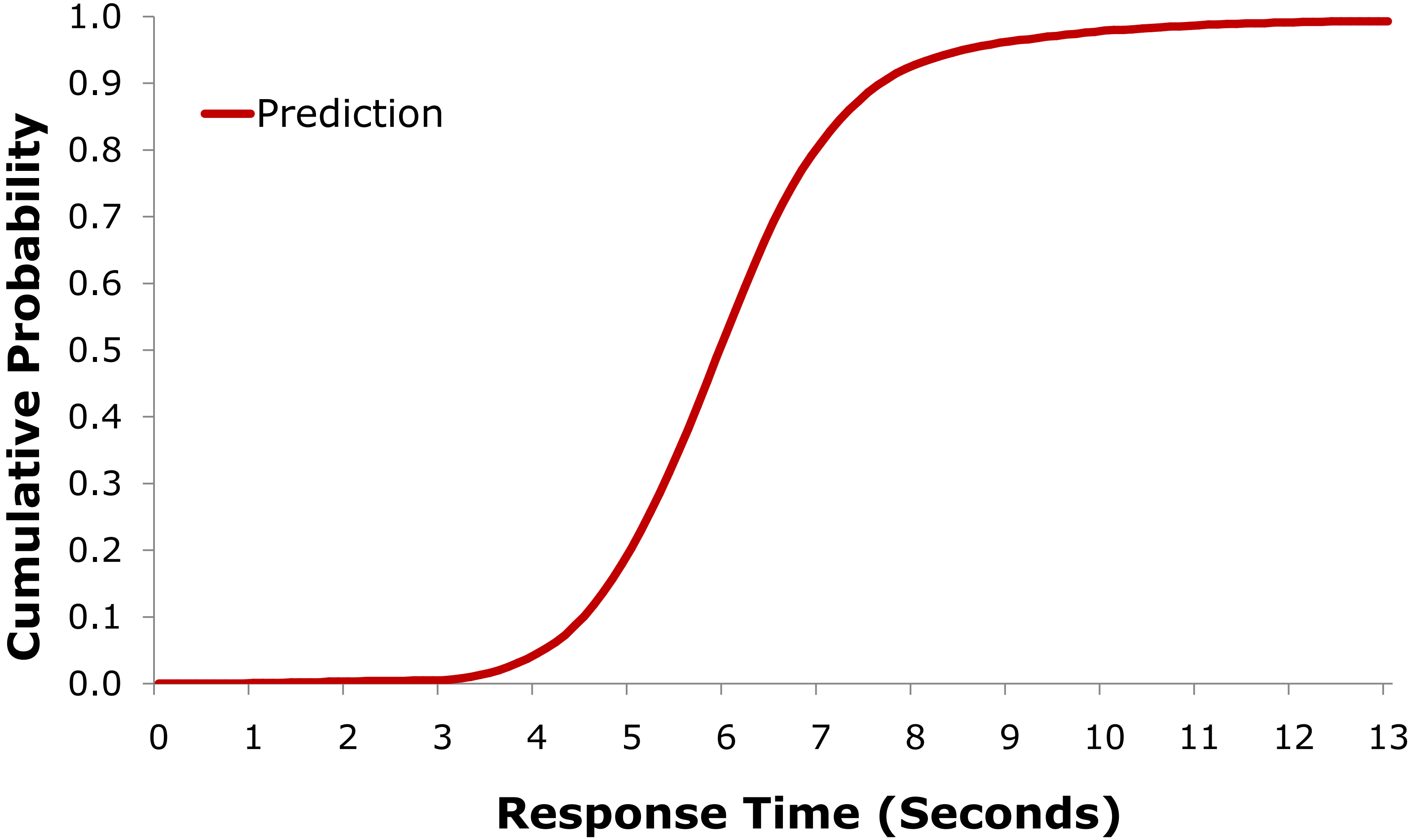


Specification



DoublePDF[(1;0)(2;0.4)**(3;0.6)**]

PDF



Cumulative Probability
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Prediction
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Response Time (Seconds)

Semantics

- X ~ DoublePDF[(1;0)(2;0.4)(3;0.6)]
  - P(0<=x<1) = 0
  - P(1<=x<2) = 0.4
  - P(2<=x<3) = 0.6
  - P(1<=x<1.5) = 0.2
  - …
- Y ~ IntPMF[(1;0.2)(2;0.5)(3;0.3)]
  - P(Y = 1) = 0.2
  - P(Y = 2) = 0.5
  - P(Y = 3) = 0.3
  - P(Y = n) = 0 for all n >=4

Functional dependent random variables

- X ~ IntPMF
- Y ~ IntPMF
- Z = X * Y
  - Z is also a Random Variable
  - Z ~ IntPMF
  - Z's distribution is derived automatically
- Operators: +, -, *, /, ^, <, >, …
- Resulting grammar is called Stochastic Expression (StoEx) in PCM

Using random variables for modelling

- Where can we use random variables?
  - Loop iterations
  - Branch conditions
  - Inter arrival time
  - Think time
  - For input parameter characterisations
  - For output/return parameter characterisations
  - For resource demands

Introducing variables…

- We can define our own variables to describe parameters
- They are set at the caller‘s side
- They are used at the called side
- Model performance relevant dependencies *only!*
  - Most parameters have no or only little influence on the performance
  - Omit these parameters from the specification!
  - Example: int ICalculator.add(int a, int b) Performance is not depending significantly on any parameter value!

Parameter Abstractions

- We normally do not model parameter values but performance abstractions
- The following types are available
  - BYTESIZE: Memory footprint of a parameter
  - VALUE: The actual value of a parameter for primitive types
  - STRUCTURE: Structure of data, like „sorted“ or „unsorted“
  - NUMBER_OF_ELEMENTS: The number of elements in a collection
  - TYPE: The actual type of a parameter (vs. the declared type)

Examples

Void aMethod(int a, int[] b, MyFigure c)

Caller Specifies:

a.BYTESIZE = 4

a.VALUE = IntPMF[(10;0.2)(30;0.4)(100;0.4)]

b.NUMBER_OF_ELEMENTS = 100

c.TYPE = EnumPMF[(„circle";0.4)(„rectangle";0.6)]

Example cont.

Void aMethod(int a, int[] b, MyFigure c)

Use in the SEFF of aMethod

aLoop.Iterations = a.VALUE

anAction.ResourceDemand = b.NUMBER_OF_ELEMENTS * 100

aBranch.Condition = c.TYPE == „circle"

Special Keywords

- INNER
  - Refers to the elements of a collection
  - Describes the contents of the collection
- RETURN
  - Refers to the return value of the current SEFF
  - Characterises the result
- Namespace of variables
  - Characterise inner elements of composed data types

Examples

Void aMethod(int a, int[] b, MyFigure c)

b.INNER.BYTESIZE = 4

b.INNER.VALUE = 42

b.INNER.VALUE = IntPMF[(42;0.5)(43;0.5)]

c.color.VALUE = EnumPMF[(„red";0.1)(„green";0.9)]

Editor Support

- „StoEx-Dialog"
- Offers syntax highlighting, code completion, online help and basic syntax checking
- Often available on double click of the corresponding model element

Semantic difference
Loop and Collection Iterator

- In a Loop *all* characterisations are evaluated any time they occur (stochastical independence)

```
// a.INNER.BYTESIZE=IntPMF[(1;0.5)(10;0.5)]
Object[] a = ...
for (int i=0; i < 10; i++) {
  // a.INNER.BYTESIZE can be 1 in doSth
  doSth(a[i]);
  // a.INNER.BYTESIZE can be 10 in doSthElse
  doSthElse(a[i]);
}
```

Semantic difference Loop and Collection Iterator

- In a Collection Iterator *all* characterisations are evaluated any time they occur (stochastical independence) execpt the INNER characterisations of the iterator parameter

```
// a.INNER.BYTESIZE=IntPMF[(1;0.5)(10;0.5)]
Object[] a = ...
for (Object o:a) {
  // a.INNER.BYTESIZE can be 1 in doSth
  doSth(o);
  // a.INNER.BYTESIZE is also 1 in doSthElse
  doSthElse(o);
}
```

Semantics: Dependant Branches

```
// x.VALUE=IntPMF[(1;0.5)(6;0.3)(12;0.2)]
if (x > 5) {
  if (x > 10) {
  } else {
  }
}
```

If you would have to model this with probabilistic branch transitions, what would be the probabilities? (Tip: Bayes Theorem!!!)

Semantics: Dependant Branches

```
// x.VALUE=IntPMF[(1;0.5)(6;0.3)(12;0.2)]
if (x > 5) { // p = 0.5
  // x.VALUE is always 6 or 12 here!
  if (x > 10) { // p = 0.4
     // x.VALUE is always 12 here!
  } else { // p = 0.6
     // x.VALUE is always 6 here!
  }
} else { // p = 0.5
  // x.VALUE is always 1 here!
}
```

Our tools respect this automatically, you don't have to calculate on your own!

Now: Exercises in the Tool

- Switch to Eclipse!

Lessons Learned Today

- What is uncertainty?
- How is it modelled in PCM?
- Random Variables
- Random Variables in the PCM
  - Loop Iterations
  - Branch Conditions
  - Resource Demands
  - Parameter characterisations
  - Usage model details