VORLESUNG

Automatisierung industrieller Workflows

Teil C: Die Sprache SLX

- DISCO- Elemente -

Joachim Fischer

Inhalt C.5 (Wdh.)

- **Teil A**
  Aspekte von Modellierung und Simulation dynamischer Systeme
- **Teil B**
  Die Modellierungssprache UML
- **Teil C**
  Die ausführbare Modellierungssprache SLX
- **Teil D**
  Modellierung von Lieferketten

- **C.1**
  Einführung und Basissprache
- **C.2**
  Stochastische Prozesse in SLX
- **C.3**
  GPSS-Elemente
- **C.4**
  ODEMx-Elemente
- **C.5**
  Obektorientierung
- **C.6**
  DISCO-Elemente

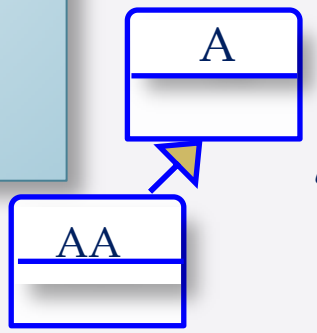
Augment: Vererbungsersatz (Wdh.)

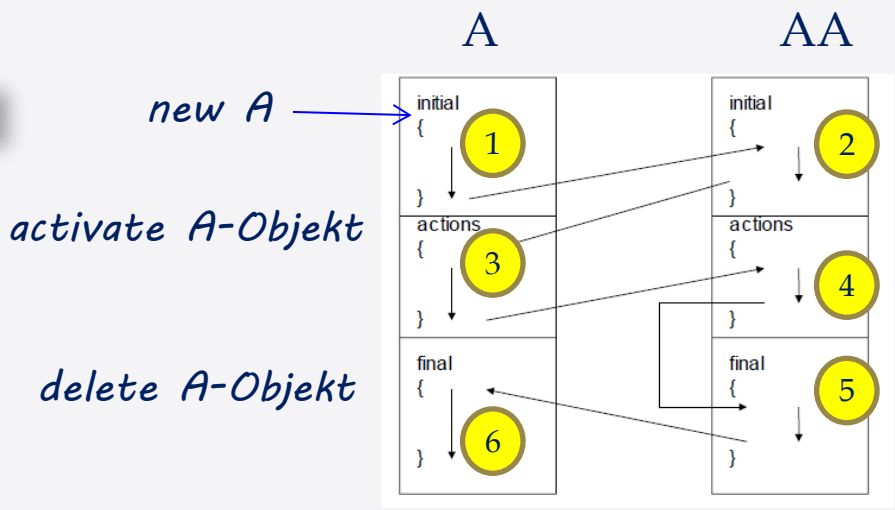
- Version 1.x (Konzept gibt es auch in anderen Sprachen: ObjectiveC)

```
class A {
      A1-attributes…
      A1-Methoden…
      A1-Properties…
};

augment A  {   //AA
      A2-attributes…
      A2-Methoden…
      A2-Properties…
};
```

A wird um Bestandteile erweitert
besser: A wird durch A + augment(A) ersetzt

…. und dies gegebenenfalls durchaus mehrmals

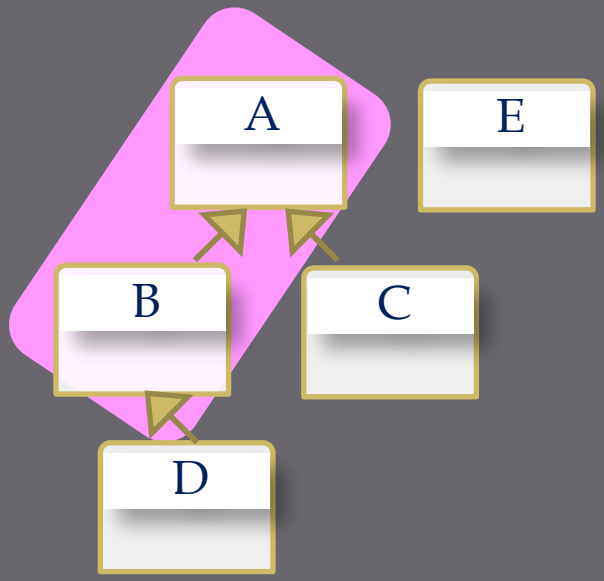




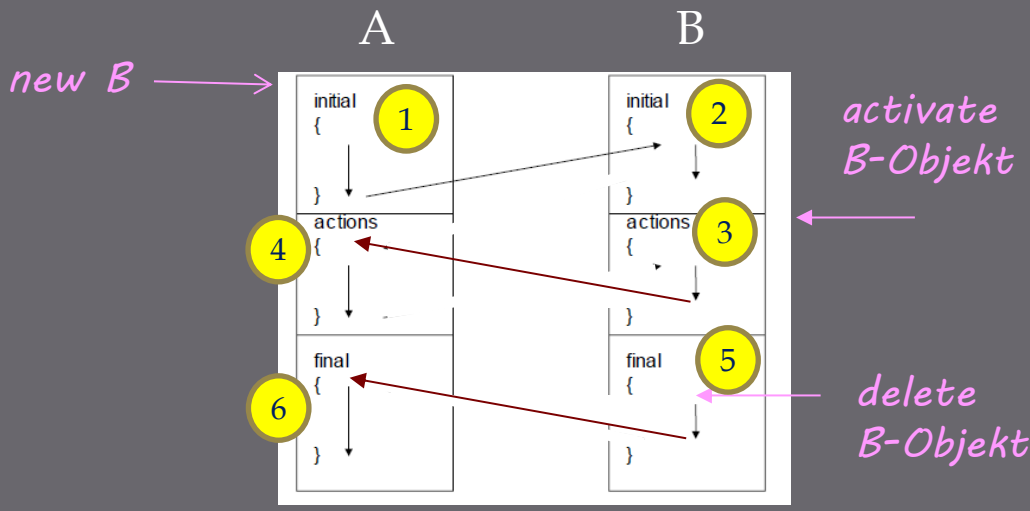
korrekte Anwendung liegt in der Hand des Nutzers

Aktionsreihenfolge für Augment-Klassen-Objekte

Vererbung, Virtuelle Methoden (Wdh.)



Aktionsreihenfolge für Objekte abgeleiteter Klassen



overridable == virtual
override== redefines

Regel: einmal overridable,
danach immer overridable

OFFEN:

1. Wann genau werden die Pucks generiert?
2. Wie kommen sie in die MovingPuck-Liste?

Ablauf ← Nachtrag zur letzten Vorlesung

- Aktivierung eines Objektes einer Klasse C
  (3-stufiger Vererbung: A, B, C)

- Erzeugung eines Pucks C1/1 (steuert Actions von C)
  und Eintrag in LMP

- C1/1-Puck wird ausgeführt
  Bevor C-Actions startet, wird C1/2 (steuert Actions von B) erzeugt
  und in LMP eingetragen

- **Ann.**: C1/1 -> wait
  Fortsetzung mit C1/2
  Bevor B-Actions startet, wird C1/3 (steuert Actions von C) erzeugt
  und in LMP eingetragen

- FAZIT: Alle Actions der Vererbungskette haben eigene Pucks, die zum Generierungszeitpunkt (Modellzeit) gemeinsam in der LMP stehen könnten

ACHTUNG (Wdh.)

weitere Situation, die zur impliziten Generierung von Pucks führt

1. **bislang bekannt**
   - main-Start
   - **activate** *classId-objectNr-1*
   - **fork** *classId-objectNr-n*

2. **hinzu kommt**
   - bei Ausführung einer Action-Property wird zu Beginn ein Child-Puck für die Action-Property der (falls vorhanden) nächsten Basisklasse generiert

Bem.-1: bei Bedarf muss der nutzerdefinierte Puck-Pointer „my_pucK“ zu Beginn jeder Action-Property einer Vererbungskette neu gesetzt werden:

my_puck= **ACTIVE**;

bzw. jede Ableitungsstufe definiert seinen eigenen Puck-Pointer

Beispiel

```
Execution begins
Limo 1 Vehicle initial property
Limo 1 Car initial property
Limo 1 Limo initial property
0: main 1/1 has reached a breakpoint
```

Inheritance.slx - main 1/1

```
        final {
                print (ME)      "_ Car final property\n";
        }
}

set(Vehicle)            Vset;

procedure main() {
        pointer(Limo)   l;

        l = new Limo;

        activate l;
        advance (10.0);
        l= NULL;
        }
}
```

- Vehicle <- Car <-Limo

Log - Time 0

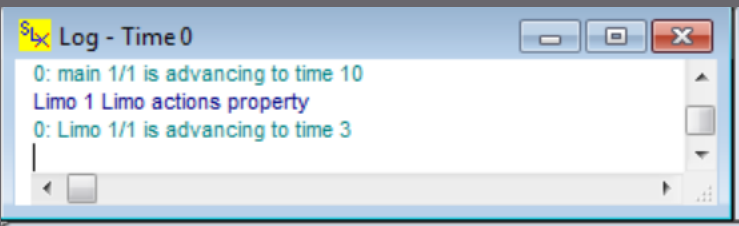
```
Limo 1 Limo initial property
0: main 1/1 has reached a breakpoint
0: main 1/1 is advancing to time 10
```

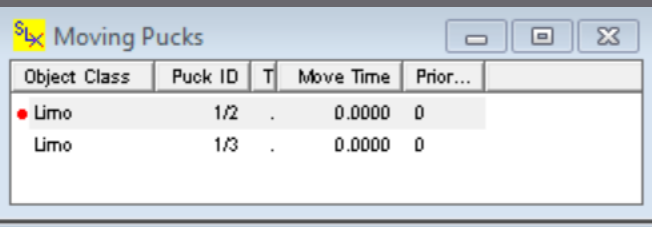
Moving Pucks

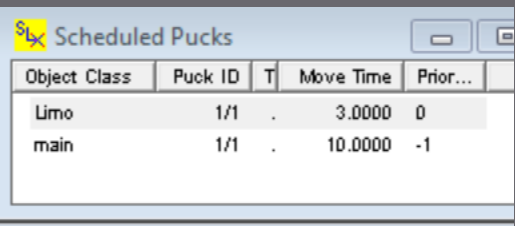
Object Class	Puck ID	T	Move Time	Prior...
Limo	1/1	.	0.0000	0
Limo	1/2	.	0.0000	0

Inheritance.slx - Limo 1/1

```
        string(10)      limoName;

        initial {
                print (ME)      "_ Limo initial property\n";
        }

        actions {
                c_puck= ACTIVE;
                print (ME)      "_ Limo actions property\n";
                advance(3.0);
                wait;
        }


        final {
                print (ME)      "_ Car final property\n";
        }
}

set(Vehicle)            Vset;

procedure main() {
        pointer(Limo)   l;

        l = new Limo;

        activate l;
        advance (10.0);
```

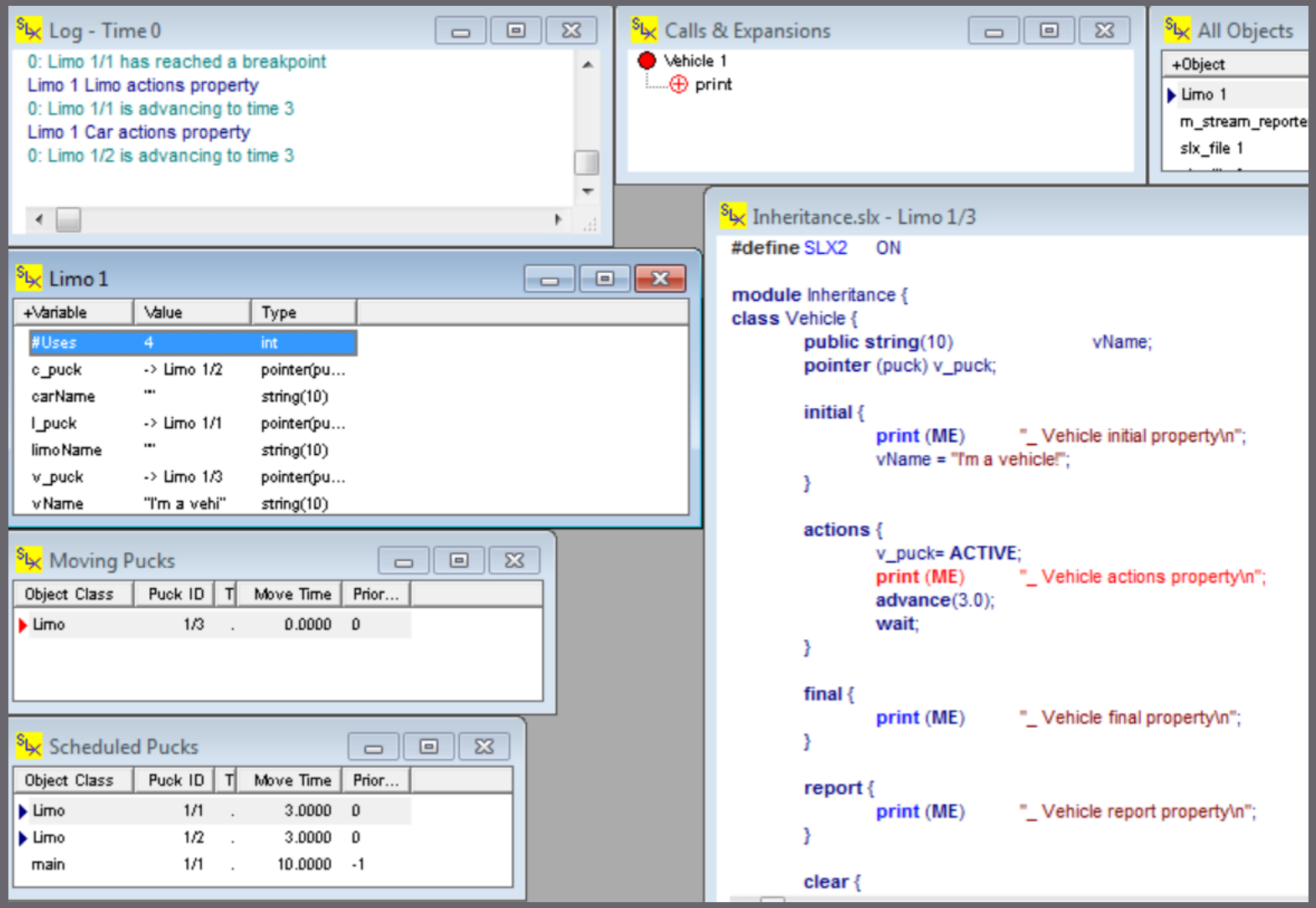






Inheritance.slx - Limo 1/2

```
        destroy {
                print (ME)      "_ Vehicle destroy property\n";
        }
};

class Car subclass(Vehicle()) {
        pointer (puck)  c_puck;
        string(10)      carName;

        initial {
                print (ME)      "_ Car initial property\n";
        }

        actions {
                c_puck= ACTIVE;
                print (ME)      "_ Car actions property\n";
                advance(3.0);
                wait;
        }


        final {
                print (ME)      "_ Car final property\n";
        }
}

class Limo subclass(Car()) {
        pointer (puck)  l_puck;
```


Log - Time 0
0: Limo 1/1 has reached a breakpoint
Limo 1 Limo actions property
0: Limo 1/1 is advancing to time 3
Limo 1 Car actions property
0: Limo 1/2 is advancing to time 3
Calls & Expansions
Vehicle 1
print
All Objects
+Object
Limo 1
m_stream_reporte
slx_file 1
Limo 1
+Variable Value Type
#Uses 4 int
c_puck -> Limo 1/2 pointer(pu...
carName "" string(10)
l_puck -> Limo 1/1 pointer(pu...
limoName "" string(10)
v_puck -> Limo 1/3 pointer(pu...
vName "I'm a vehi" string(10)
Moving Pucks
Object Class Puck ID T Move Time Prior...
Limo 1/3 . 0.0000 0
Scheduled Pucks
Object Class Puck ID T Move Time Prior...
Limo 1/1 . 3.0000 0
Limo 1/2 . 3.0000 0
main 1/1 . 10.0000 -1
Inheritance.slx - Limo 1/3
#define SLX2 ON
module Inheritance {
class Vehicle {
public string(10) vName;
pointer (puck) v_puck;
initial {
print (ME) "_ Vehicle initial property\n";
vName = "I'm a vehicle!";
}
actions {
v_puck= ACTIVE;
print (ME) "_ Vehicle actions property\n";
advance(3.0);
wait;
}
final {
print (ME) "_ Vehicle final property\n";
}
report {
print (ME) "_ Vehicle report property\n";
}
clear {

Inhalt C.6

- **Teil A** Aspekte dynamis
- **Teil B** Die Moc
- **Teil C** Die ausf SLX
- **Teil D** Modellie

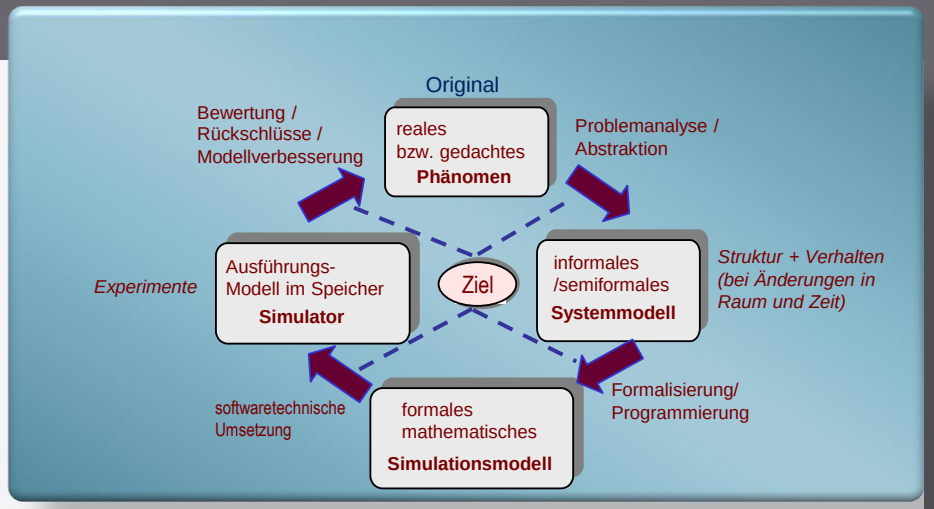
- **C.1** Einführung und Ba
- **C.2** Stochastische Pro
- **C.3** GPSS-Elemente
- **C.4** ODEMx-Elemente
- **C.5** Obektorientierung
- **C.6** DISCO-Elemente

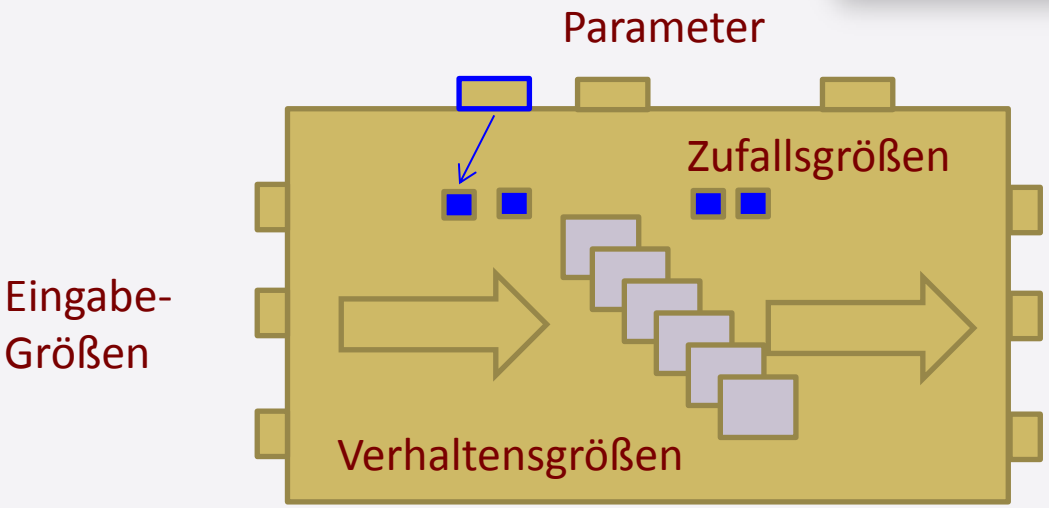
1. Simula-Bibliothek DISCO
2. SLX-Bibliothek Statistics
   - Modellgrößen, Bewertungsgrößen, Kennwertermittlung
   - Random_Variable, Histogram, Statistics
   - Einfache Simulationsläufe, Histogramm-Auswertung
   - Zeitintervall-basierte Beobachtung und Auswertung
   - Konfidenz-Intervall-Schätzung
   - Sequential Sampling
   - Vergleich zwischen simulierten Systemvarianten

Experimente und deren Auswertung (Wdh.)

Typischer Ablauf

- Sammlung von Beobachtungsdaten
  je ausgezeichnete Modellvariable
  (Ergebnisgröße)
  in einem Simulationslauf

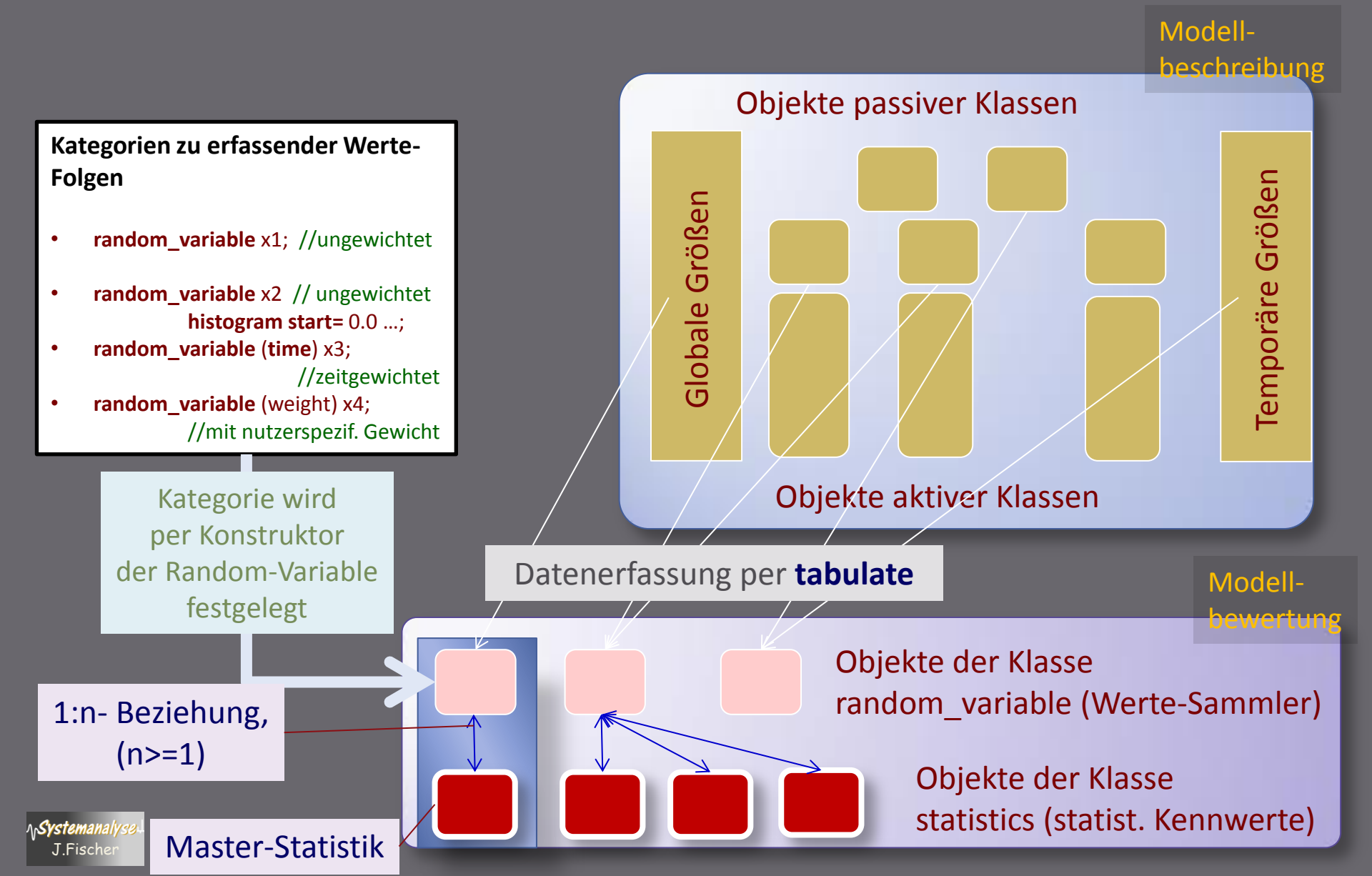




Beispiele
- Länge eines WS
- Antwortzeit

schwankende Werte
~ Beobachtungsdauer
~ Startwerte der ZZG

Erfassung von Beobachtungsdaten (Wdh.)



Erzeugung von random_variable-Instanzen (Wdh.)

- *Anweisung (Syntax):*

random_variable [**(time** | **weight)]** *random_variable_ident*
histogram start= *start_value* **width=** *width_value* **count=** *count_value*]
[**title=** *report_title*]

- *Beispiele*

```
random_variable x1;                    //ungewichtet
```

```
random_variable x2
        histogram start= 0.0 width= 0.5 count= 20; // ungewichtet
```

```
random_variable (time) x3;             //zeitgewichtet
```

```
random_variable (weight) x4;           //mit nutzerspezif. Gewicht
```

Erfassung der Beobachtungen (Wdh.)

- Anweisung (Syntax):

tabulate *random_variable_ident* **=** *observed_value*
[{ **weight =** *weight_value* } | { **count=** *count_increment* }]

- Beispiele

tabulate x1= **time** - active->mark_time;//ungewichtet

tabulate x2 = tanker->load; // ungewichtet mit Histogramm

tabulate x3 = tankerQ.length; // zeitgewichtet, bei Eintritt
tabulate x3 = tankerQ.length **count**= 0; // zeitgewichtet, bei Austritt

tabulate x4 = t **weight**= tank->pressure.state; //mit nutzerspezif. Gewicht

lokale Zeit

Histogramm ist dann ein Zeitdiagramm

Attribute von Statistics (Wdh.)

```
read_only class statistics(        pointer(random_variable) stat_root_rv,
                                   pointer(interval)        stat_interval)      {
        int         count;
        double      sum,
                    sum_of_weights,
                    sum_of_squares,
                    min_value,
                    max_value;
        // SOB variables
        double      SOB_low_sum,
                    SOB_high_sum,
                    SOB_mean,
                    SOB_svar_sum;
        int         SOB_batch_count,
                    SOB_sample_no;
```

Speicherplätze zur Profilbestimmung

- Anzahl der Beobachtungen
- Summe der bisher beobachteten Werte
- summe der bisher eingesetzten Gewichtswerte
- Summe der Quadrate der bisher beobachteten Werte
- Minimum
- Maximum

```
// upper half-batch sum of values
// current (recursively calculated) mean
```

pointer(histogram)
pointer(random_variable)
pointer(interval)

Berechnung erfolgt erst zur Abruf-Zeit

Mittelwert m

$$m = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Streuung/Varianz s^2

Standardabweichung s

$$s^2 = \frac{1}{n-1} \left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} x_i \right)^2 \right)$$

Dynamische Kennwertermittlung (Wdh.)

- SLX-Makros

Achtung: Der Nutzer kommt gar nicht in Kontakt mit Statistik-Objekten

sample_count (*rv* [**over** *interval*] **)**	**int**	Anzahl der Beobachtungen
sample_max (*rv* [**over** *interval*] **)**	**float**	Maximum
sample_min (*rv* [**over** *interval*] **)**	**float**	Minimum
sample_sum (*rv* [**over** *interval*] **)**	**float**	Zeitintegral über die Werte
sample_mean (*rv* [**over** *interval*] **)**	**float**	Erwartungswert
sample_variance (*rv* [**over** *interval*] **)**	**float**	Varianz
sample_stdev (*rv* [**over** *interval*] **)**	**float**	Standardabweichung
sample_time_per_unit (*rv* [**over** *interval*] **)**	**float**	Zeitintegral dividiert durch die Anzahl der Beobachtungen

Berechnung durch vorinstallierte Master-Statistik oder optionale Intervall-Statistik

Inhalt C.6

- **Teil A** Aspekte dynamis
- **Teil B** Die Moc
- **Teil C** Die ausf SLX
- **Teil D** Modellie

- **C.1** Einführung und Ba
- **C.2** Stochastische Pro
- **C.3** GPSS-Elemente
- **C.4** ODEMx-Elemente
- **C.5** Obektorientierung
- **C.6** DISCO-Elemente

1. Simula-Bibliothek DISCO
2. SLX-Bibliothek Statistics
   - Modellgrößen, Bewertungsgrößen, Kennwertermittlung
   - Random_Variable, Histogram, Statistics
   - Einfache Simulationsläufe, Histogramm-Auswertung
   - Zeitintervall-basierte Beobachtung und Auswertung
   - Konfidenz-Intervall-Schätzung
   - Sequential Sampling
   - Vergleich zwischen simulierten Systemvarianten

Beispiel-Anforderungen (Wdh.)

Bank mit
- 3 Schaltern
- 1 gemeinsame Warteschlange der Schalter
- Kundenstrom

Untersuchungsziel: statistische Kennwerte für
- Wartezeit (zusätzlich mit Histogramm)
- Warteschlangenlänge
- Verweilzeit

Randbedingung:
- Bank schließt nach 8 h
- Simulation ist beendet, wenn letzter Kunde die Bank verlassen hat

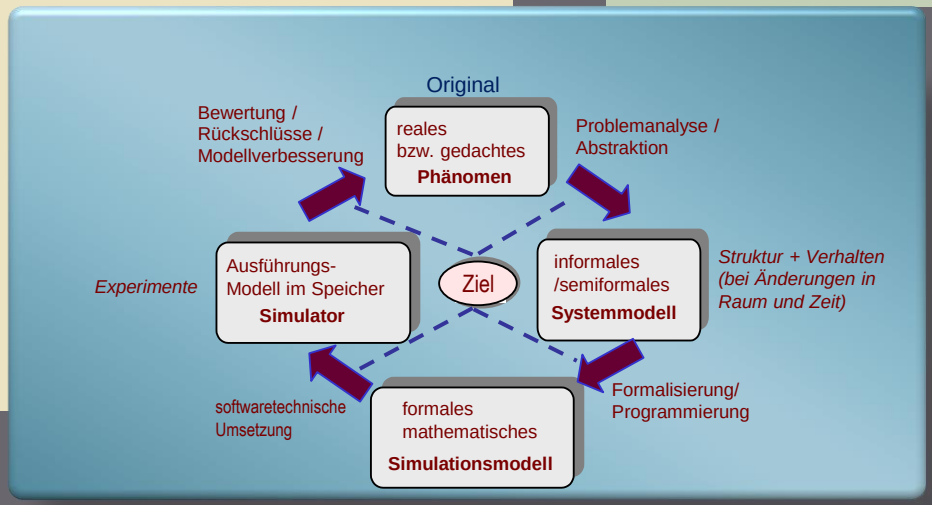
storage clerk

random_variable

rv_queuing_time

rv_queue_length

rv_elapsed_time



Beispiel (Wdh.)

Bank mit
- 3 Schaltern
- 1 gemeinsame Warteschlange der Schalter
- Kundenstrom

storage clerk

Untersuchungsziel: statistische Kennwerte für
- Wartezeit (zusätzlich mit Histogramm)
- Warteschlangenlänge
- Verweilzeit

random_variable

rv_queuing_time

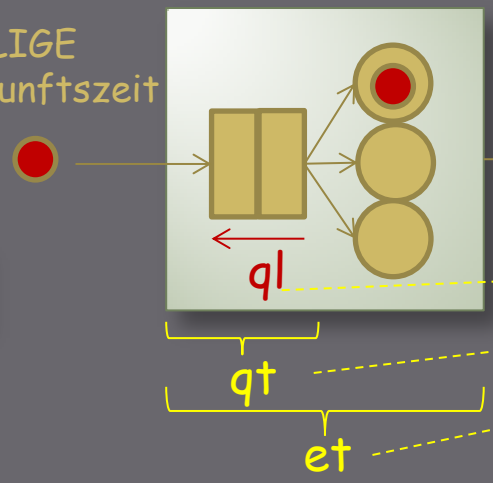
rv_queue_length

rv_elapsed_time

Randbedingung:
- Bank schließt nach 8 h
- Simulation ist beendet, wenn letzter Kunde die Bank verlassen hat

Zufallsgrößen und Statistik

(Zusammenfassung-1)



zu BEOBACHTENDE Zufallsgrößen (ZG)

- time-gewichtete ZG
- ungewichtete ZG mit Histogramm
- ungewichtete ZG

x: **random_variable**

Standard-Report

master

: statistics

Werte-Erfassung

tabulate x= … ;

tabulate x= … **count**= 0 ;

Random Variable	#Observed	Mean	StD	Min	Max
x	…				

sample_count (x)

sample_max (x)

vordefinierte Makros

Beispiel: SLX-Umsetzung

```
//*****************************************************************************
// Example EX0022
//*****************************************************************************
import <stats>
import <h7>
module basic  {
        rn_stream               arrive,
                                service ;

        random_variable         rv_elapsed_time,
                                rv_queueing_time histogram start=0.0 width=0.5 count = 20;
        random_variable (time)  rv_quelength;

        storage clerk capacity=3;
        control int in_customer, out_customer;
        int que_length;
        constant float close_time=8*60, service_time=1.3 ;
        boolean door_closed;

        class Customer {
        actions {
                in_customer ++; // increment customer counter
                que_length++; // increment queue length
                tabulate rv_quelength=que_length; // tabulate
                enter clerk; // try to catch a clerk
                que_length --; // decrement queue length
                tabulate rv_quelength=que_length count = 0;
                tabulate rv_queueing_time= time - ACTIVE->mark_time;
                advance rv_expo ( service , service_time ); // service time
                leave clerk;
                out_customer ++;
                tabulate rv_elapsed_time = time - ACTIVE->mark_time;
        } //actions
        } // Customer
```

```
        procedure run_model () {
                float intensity=2.0;
                fork  { // arriving customer
                        forever {
                                advance rv_expo ( arrive , 1/intensity );
                                activate new Customer;
                                if ( door_closed ) terminate;
                        }
                }
                fork { // Controlling the bank
                        advance close_time;
                        door_closed = TRUE;
                        terminate;
                }

                wait until ( (time > close_time) && ( in_customer == out_customer ));
                report ( system );
        }

        procedure main() {
                run_model();
        }// main
}
```

EX-0022 (Bank).slx: SLX-64 UL211 Lines: 7,466 Errors: 0 Warnings: 0 Lines/Second: 278,752 Memory: 4 MB

Execution begins

System Status at Time 484.9196

Random Stream	Sample Count	Initial Position	Current Position	Antithetic Variates	Chi-Square Uniformity
arrive	924	200000	200924	OFF	0.43
service	924	400000	400924	OFF	0.63

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_elapsed_time	924	2.531	2.038		0.01	12.04
rv_quelength	924	2.362	3.213		0.00	13.00
rv_queueing_time	924	1.240	1.559		0.00	7.65

Lower	Upper	Frequency	Percent	
0.0	0.5	415	44.913	**
0.5	1.0	117	12.662	**********
1.0	1.5	114	12.338	**********
1.5	2.0	73	7.900	********
2.0	2.5	46	4.978	*****
2.5	3.0	30	3.247	***
3.0	3.5	29	3.139	***
3.5	4.0	18	1.948	**
4.0	4.5	34	3.680	****
4.5	5.0	15	1.623	*
5.0	5.5	9	0.974	*
5.5	6.0	10	1.082	*
6.0	6.5	6	0.649	
6.5	7.0	4	0.433	
7.0	7.5	3	0.325	
7.5	8.0	1	0.108	

Storage	Capacity	Total %Util	Avail %Util	Unavl %Util	Entries	Average Time/Puck	Current Status	%Avail	Average Contents	Current Contents	Maximum Contents
clerk	3	82.00			924	1.29	AVAIL	100.00	2.46	0	3

Execution complete

Objects created: 46 passive and 925 active Pucks created: 928 Memory: 4 MB Time: 0.13 seconds

Jetzt geht es weiter !

Inhalt C.6

- **Teil A** Aspekte dynamis
- **Teil B** Die Moc
- **Teil C** Die ausf SLX
- **Teil D** Modellie

- **C.1** Einführung und Ba
- **C.2** Stochastische Pro
- **C.3** GPSS-Elemente
- **C.4** ODEMx-Elemente
- **C.5** Obektorientierung
- **C.6** DISCO-Elemente

1. Simula-Bibliothek DISCO
2. SLX-Bibliothek Statistics
   - Modellgrößen, Bewertungsgrößen, Kennwertermittlung
   - Random_Variable, Histogram, Statistics
   - Einfache Simulationsläufe, Histogramm-Auswertung
   - Zeitintervall-basierte Beobachtung und Auswertung
   - Konfidenz-Intervall-Schätzung
   - Sequential Sampling
   - Vergleich zwischen simulierten Systemvarianten

Disco - Inhalt

1. Simula-Bibliothek DISCO

2. SLX-Bibliothek Statistics
   - Modellgrößen, Bewertungsgrößen, Kennwertermittlung
   - Random_Variable, Histogram, Statistics
   - Einfache Simulationsläufe, Histogramm-Auswertung
   - Zeitintervall-basierte Beobachtung und Auswertung
   - Konfidenz-Intervall-Schätzung
   - Sequential Sampling
   - Vergleich zwischen simulierten Systemvarianten

Histogramm-Auswertung

- angelegte Histogramme lassen sich weiter verarbeiten (gewisse Häufigkeitsklassen)

- Erweiterung des Bankschalter-Beispiels

a) Wie groß ist die Wahrscheinlichkeit, dass die Wartezeit der Kunden kleiner gleich 5 ZE ist ?

b) Wie groß ist die maximale Wartezeit für 50%, 90% und 95% der ankommenden Kunden?

zur Umsetzung ist ein Zugriff auf **histogram- Attribute** notwendig

ad Erweiterung a)

Wie groß ist die Wahrscheinlichkeit, dass die Wartezeit der Kunden kleiner gleich 5 ZE ist ?

1. Bestimmung der Häufigkeitsklasse (Index) des berechneten Histogramms, zu der Vergleichswert (5.0) gehört
2. Bestimmung der kumulativen absoluten Häufigkeit bis zu diesem Index
3. Bestimmung der Wahrscheinlichkeit

Umsetzung in einer Nutzer-Routine special_report_1() mit Aufruf in run_model()

```
procedure run_model () {
    float intensity=2.0;
    fork  {// Arriving Customer
      ...
    }
    fork { // Controlling the bank
      ...
    }
    wait until ( (time > close_time) &&
                                ( in_customer == out_customer ));
    report ( system );
    special_report_1 (rv_queuing_time.histo, 5.0);
}
```

Beispiel: EX-0022-a

```
procedure special_report_1 (in pointer(histogram) histo,
                            in double s_val )          {
// probability for waiting time <= s_val
        float   prob,
                sum;
        int     index,
                i;
// in the overflow area
        if ( s_val > histo->upper_bound ) {
                prob = 1.0;
       }
       else {
// determine the index in the frequency table
                index = ( s_val - histo->lower_bound ) / histo->class_width ;

// calculate the cumulative frequency
                for ( i = 0; i<=index; i++ )
                        sum+= histo->frequency [i] ;
// probability
                prob = sum* 100 / histo->sum_of_weights ;
      }
      print ( s_val ,prob ) "\n Special Report 1 \n"
                "maximum of __._ min waiting time is true for __.__ percent of the customers \n";
}
```

1. Bestimmung der Häufigkeitsklasse (Index) des berechneten Histogramms, zu der Vergleichswert (5.0) gehört
2. Bestimmung der kumulativen absoluten Häufigkeit bis zu diesem Index
3. Bestimmung der Wahrscheinlichkeit

Ausgabe-Erweiterung

```
Special Report 1
maximum of  5.0  min waiting  time for   97.40  percent of the customers
```

ad Erweiterung b)

Wie groß ist die maximale Wartezeit für 50%, 90% und 95% der ankommenden Kunden?

1. Berechnung der kumulativen Häufigkeiten aus den absoluten Häufigkeiten
2. … bis kumulative Häufigkeit den Vergleichswert einschließt

Umsetzung in einer Nutzer-Routine special_report_2() mit Aufruf in run_model()

```
procedure run_model () {
    float intensity=2.0;
    fork  {
       ...
    }
    fork { // Controlling the bank
       ...
    }
    wait until ( (time > close_time) &&
                                ( in_customer == out_customer ));
    report ( system );
    special_report_1 (rv_queuing_time.histo, 5.0);
    special_report_2 (rv_queuing_time.histo, 0.50);
    special_report_2 (rv_queuing_time.histo, 0.90);
    special_report_2 (rv_queuing_time.histo, 0.95);
}
```

Beispiel: EX-0022-b

```
procedure special_report_2 (in pointer(histogram) histo, in double s_prob) {
   // p ( max waiting_time ) = s_prob
   float value,sum;
   int index ;

   // calculate cumulative probability
   for ( index = 0; TRUE ; index++ ) {
       sum+= ( histo->frequency [index]/
                  histo->sum_of_weights);
       if ( sum >= s_prob )
                  break;
   }

  // related value
  value = ( index * histo->class_width ) + histo->lower_bound ;
  print ( s_prob*100 , value )"\n Special Report 2 \n"
   " ___._ percent of the customer waiting not longer than __.__ min \n";
}
```

1. Berechnung der kumulativen Häufigkeiten aus den absoluten Häufigkeiten
2. … bis kumulative Häufigkeit den Vergleichswert einschließt

Beispiel: EX-0022-b

```
procedure special_report_2 (in pointer(histogram) histo, in double s_prob) {
   // p ( max waiting_time ) = s_prob
   float value,sum;
   int index ;

   // calculate cumulative probability
   for ( index = 0; TRUE ; index++ ) {
      sum+= ( histo->frequency [index]/
                histo->sum_of_weights);
      if ( sum >= s_prob )
               break;
   }

  // related value
  value = ( index * histo->class_width ) + histo->lower_bound ;
```

1. Berechnung der kumulativen Häufigkeiten aus den absoluten Häufigkeiten
2. … bis kumulative Häufigkeit den Vergleichswert einschließt

```
//*************************************************
*****
// Example EX0022-a
//*************************************************
*****
…
procedure main() {
                run_model();
                special_report_1 (rv_queueing_time.histo, 5.0);
                special_report_2 (rv_queueing_time.histo, 0.5);
                special_report_2 (rv_queueing_time.histo, 0.9);
                special_report_2 (rv_queueing_time.histo, 0.95);
} // main
```

Ausgabe-Erweiterung

```
… __.__ min \n";
```

Special Report 1
Maximum of 5.0 min waiting for 97.40 percent of the customer

Special Report 2
50.0 percent of the customer waiting not longer than 0.50 min

Special Report 2
90.0 percent of the customer waiting not longer than 3.50 min

Special Report 2
95.0 percent of the customer waiting not longer than 4.50 min

Inhalt C.6

- **Teil A**
  Aspekte
  dynamis
- **Teil B**
  Die Moc
- **Teil C**
  Die ausf
  SLX
- **Teil D**
  Modellie

- **C.1**
  Einführung und Ba
- **C.2**
  Stochastische Pro
- **C.3**
  GPSS-Elemente
- **C.4**
  ODEMx-Elemente
- **C.5**
  Obektorientierung
- **C.6**
  DISCO-Elemente

1. Simula-Bibliothek DISCO
2. SLX-Bibliothek Statistics
   - Modellgrößen, Bewertungsgrößen, Kennwertermittlung
   - Random_Variable, Histogram, Statistics
   - Einfache Simulationsläufe, Histogramm-Auswertung
   - Zeitintervall-basierte Beobachtung und Auswertung
   - Konfidenz-Intervall-Schätzung
   - Sequential Sampling
   - Vergleich zwischen simulierten Systemvarianten

Zeitintervallbezogene Auswertungen

- Motivation:
  z.B.: Anlaufphasen,
  stabile Nutzungsphasen,
  Schicht-Regime in Produktionsabläufen
  verlangen separate Bewertungen

- **interval**-Objekt wird mit Objekten von **random_variable** verbunden
  (ein **random-variable**-Objekt ist durch mehreren Statistik-Intervallen zugeordnet)

- für jedes Intervall (**interval**-Objekt) wird eine eigene Statistik für die zugeordneten **random_variable**-Objekten geführt

Interval-Instanzerzeugung

Anweisung (Syntax):

interval *interval_ident* [**title=** *report_title*]

Beispiel

```
interval moorning, noon, afternoon;
```

Zuordnung zu einem **random_variable**-Objekt wird per **observe** bestimmt:

Anweisung (Syntax)

observe *randomvariable_ident, …* **over** *interval_ident, …;*

Beispiel

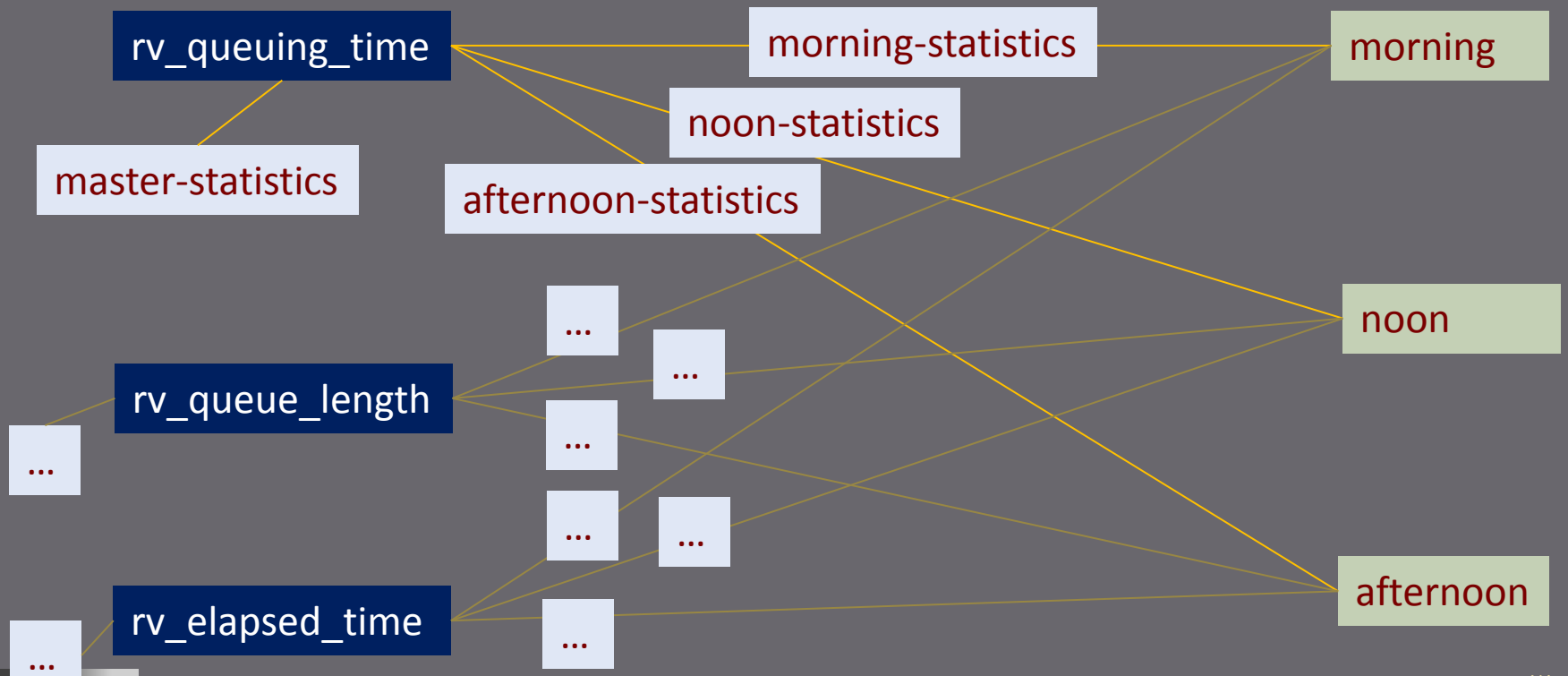
```
observe rv_elapsed_time, rv_queuing_time, rv_queue_length
        over moorning, noon, afternoon;
```

Objekt-Konfiguration

observe rv_elapsed_time, rv_queuing_time, rv_queue_length
over moorning, noon, afternoon;

random_variable- Instanzen

interval- Instanzen



Attribute von interval

(Zeit-)Intervall-Länge

```
class interval (string(*) report_title)  {
                string (16)                 title;
                double                      base_time;
                boolean                     active;
                set (statistics)            interval_statistics;

                static string (*)           interval_format = …;
…
}
```

je zugeordneter Random-Variable ein Statistik-Objekt

Kennzeichnung aktiver und passiver Phasen bei der Datenaufzeichnung

für dynamische Übergänge stehen spezielle Anweisungen zur Verfügung:

- **start_interval** *interval_ident*;
- **stop_interval** *interval_ident*;

Intervall-bezogene Beobachtungen

- Aktivierung/Deaktivierung von **interval**-Objekten kann in Abhängigkeit von
  - der Simulationszeit oder
  - Zustandsbedingungen

  erfolgen

Beispiel

```
/* Morning Time */
start_interval morning;
advance 4*60;
stop_interval morning;
```

```
reset interval_ident, … ;
                    // Zurücksetzung eines Intervalls
```

Beispiel-Erweiterung

Bank mit

- 5 Schaltern
- 1 gemeinsame Warteschlange der Schalter
- wechselnde Kundenstrom-Intensität
- Über die Mittagszeit schließen zwei Schalter

Untersuchungsziel: statistische Kennwerte für

- Wartezeit (zusätzlich mit Histogramm)
- Warteschlangenlänge
- Verweilzeit

Randbedingung:

- Bank schließt nach 8 h
- Simulation ist beendet, wenn letzter Kunde die Bank verlassen hat

storage clerk

random_variable

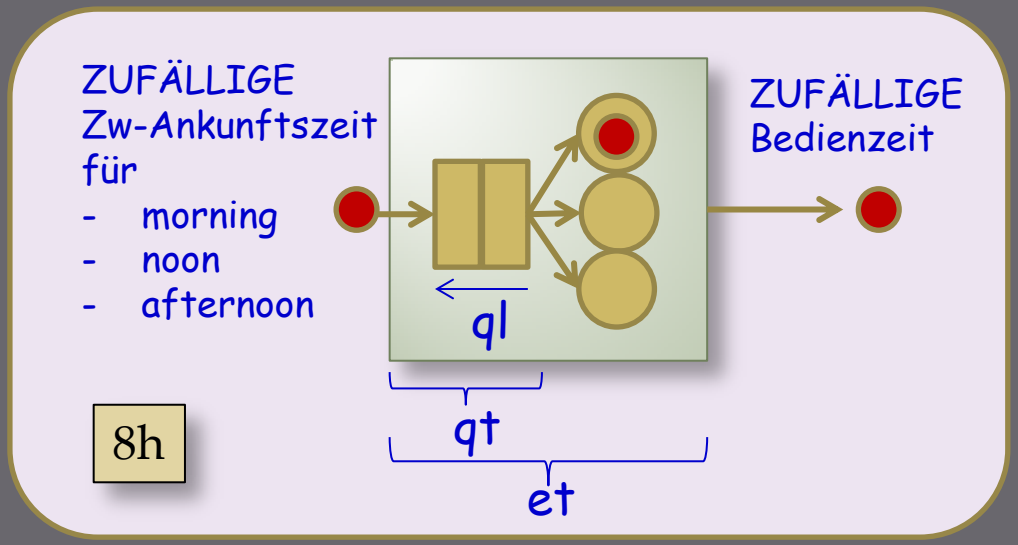
- rv_queuing_time
- rv_queue_length
- rv_elapsed_time

interval

- morning
- noon
- afternoon

Zufallsgrößen und Intervall-Statistik

(Zusammenfassung-2)

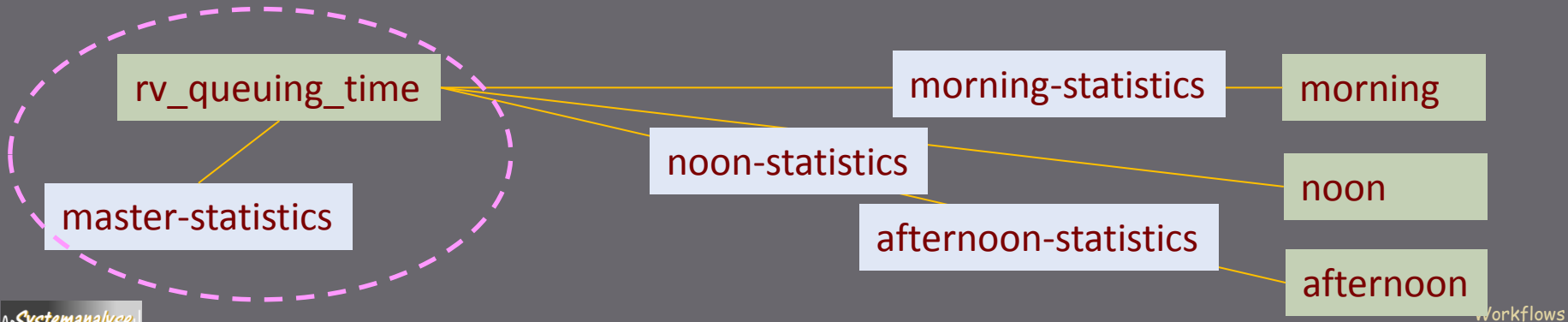



```
random_variable x ... ;
interval y;
observe x over y;
```

random_variable- Instanzen

statistics- Instanzen

interval- Instanzen



```
//*************************************************************************
// Example EX0023
//*************************************************************************
import <stats>
import <h7>
module basic {
      rn_stream arrive, service ;
      random_variable rv_elapsed_time,
                  rv_queueing_time
                  histogram start=0.0 width=0.5 count = 20;
      random_variable (time) rv_quelength;
      storage clerk capacity=5;
      interval morning, noon , afternoon ;
      control int in_customer, out_customer;
      int que_length;
      constant float close_time=8*60, service_time=1.3 ;
      boolean door_closed;
class cl_customer {
       actions {
                  in_customer ++; // increment customer counter
                  que_length++; // increment queue length
                  tabulate rv_quelength=que_length; // tabulate
                  enter clerk; // try to catch a clerk
                  que_length --; // decrement queue length
                  tabulate rv_quelength=que_length count=0 ;
                  tabulate rv_queueing_time= time - ACTIVE->mark_time;
                  advance rv_expo ( service , service_time ); // service time
                  leave clerk;
                  out_customer ++;
                  tabulate rv_elapsed_time = time - ACTIVE->mark_time;
       }
}
procedure main() {
                  observe rv_elapsed_time, rv_queueing_time , rv_quelength over
                  morning, noon , afternoon ;
                  run_model();
}// main
```

```
procedure run_model () {
   float intensity=2.0;
   fork  {     // Arriving Customer
         forever  {
                  advance rv_expo ( arrive , 1/intensity );
                  activate new cl_customer;
                  if ( door_closed ) terminate;
         }
   }
   fork  {   // Controlling the bank
    /* Morning Time */
      start_interval morning;
      intensity = 3 ;
      advance 3*60 ;
      stop_interval morning;
    /* Noon Time */
      enter clerk units=2;   //Entzug von Schaltern, evtl. Verzögerung
                             // des Beginns der Mittagszeit
      start_interval noon;
      intensity = 2 ;
      advance 2*60 ;
      stop_interval noon;
      leave clerk units=2 ;
    /* Afternoon Time */
      start_interval afternoon;
      intensity = 3;
      advance 3*60;
      stop_interval afternoon;
      door_closed = TRUE;
      terminate;
   }
   wait until ( (time > close_time) && ( in_customer == out_customer ));
   report ( system );
}
}
```

EX-0023 (Bank, Interval).slx: SLX-64 OV031 Lines: 7,507 Errors: 0 Warnings: 0 Lines/Second: 292,008 Memory: 4 MB

Execution begins

System Status at Time 481.8975

Random Stream	Sample Count	Initial Position	Current Position	Antithetic Variates	Chi-Square Uniformity
arrive	1263	200000	201263	OFF	0.38
service	1263	400000	401263	OFF	0.33

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_elapsed_time	1263	1.870	1.656		0.01	12.93
rv_quelength	1263	1.476	2.565		0.00	13.00
rv_queueing_time	1263	0.563	0.978		0.00	5.36

Master-Statistik

Lower	Upper	Frequency	Percent	
0.0	0.5	872	69.042	| **********
0.5	1.0	123	9.739	| *******
1.0	1.5	110	8.709	| ******
1.5	2.0	49	3.880	| **
2.0	2.5	43	3.405	| **
2.5	3.0	14	1.108	|
3.0	3.5	10	0.792	|
3.5	4.0	8	0.633	|
4.0	4.5	25	1.979	| *
4.5	5.0	8	0.633	|
5.0	5.5	1	0.079	|

3. Intervall

Statistics Collection Intervals

Interval	Elapsed Time
afternoon	180.0000

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_elapsed_time	507	1.687	1.484		0.01	12.93
rv_queueing_time	511	0.385	0.590		0.00	2.57

Lower	Upper	Frequency	Percent	
0.0	0.5	364	71.233	| **********
0.5	1.0	53	10.372	| *******
1.0	1.5	61	11.937	| ********
1.5	2.0	22	4.305	| ***
2.0	2.5	10	1.957	| *
2.5	3.0	1	0.196	|

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_quelength	511	1.094	1.951		0.00	11.00

Interval	Elapsed Time
morning	180.0000

3. Intervall

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_elapsed_time	513	1.692	1.544		0.02	8.67
rv_queueing_time	514	0.352	0.609		0.00	3.36

Lower	Upper	Frequency	Percent	
0.0	0.5	395	76.848	| **********
0.5	1.0	51	9.922	| ******
1.0	1.5	28	5.447	| ***
1.5	2.0	22	4.280	| **
2.0	2.5	14	2.724	| *
2.5	3.0	2	0.389	|
3.0	3.5	2	0.389	|

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_quelength	514	1.006	1.831	0.00	9.00	

Interval	Elapsed Time
noon	120.0000

2. Intervall

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_elapsed_time	237	2.636	1.992		0.01	7.61
rv_queueing_time	237	1.404	1.637		0.00	5.36

Lower	Upper	Frequency	Percent	
0.0	0.5	112	47.257	| **********
0.5	1.0	19	8.017	| ********
1.0	1.5	21	8.861	| *********
1.5	2.0	5	2.110	| **
2.0	2.5	19	8.017	| ********
2.5	3.0	11	4.641	| ****
3.0	3.5	8	3.376	| ***
3.5	4.0	8	3.376	| ***
4.0	4.5	25	10.549	| **********
4.5	5.0	8	3.376	| ***
5.0	5.5	1	0.422	|

Random Variable	#Observed	Mean or ~Value	Std Dev or ~Error	Sig. Digits	Minimum	Maximum
rv_quelength	237	2.772	3.679		0.00	13.00

Storage	Total Capacity	Avail %Util	Unavl %Util	Average %Util	Current Entries	Average Time/Puck	Current Status	Maximum %Avail	Contents	Contents	Contents
clerk											

Inhalt C.6

- **Teil A** Aspekte dynamis
- **Teil B** Die Moc
- **Teil C** Die ausf SLX
- **Teil D** Modellie

- **C.1** Einführung und Ba
- **C.2** Stochastische Pro
- **C.3** GPSS-Elemente
- **C.4** ODEMx-Elemente
- **C.5** Obektorientierung
- **C.6** DISCO-Elemente

1. Simula-Bibliothek DISCO
2. SLX-Bibliothek Statistics
   - Modellgrößen, Bewertungsgrößen, Kennwertermittlung
   - Random_Variable, Histogram, Statistics
   - Einfache Simulationsläufe, Histogramm-Auswertung
   - Zeitintervall-basierte Beobachtung und Auswertung
   - Konfidenz-Intervall-Schätzung
   - Sequential Sampling
   - Vergleich zwischen simulierten Systemvarianten

Stichprobe, Konfidenzintervall

Stichprobe eine Beobachtung von bestimmten Parametern mit Zufallsschwankungen

Beispiel: ø-Wartezeit, … von Bank für einen Arbeitstag

Grundgesamteit Beobachtung **unendlich** vieler Stichproben (∞Arbeitstage)

Konfidenzintervall

Wertebereich, in dem man den interessierenden Parameter der Grundgesamtheit mit einer bestimmten Wahrscheinlichkeit erwartet bzw. schätzt

Präzision der Schätzung

Breite des Konfidenzintervalls

~ gewünschte *Sicherheit* der Schätzung (im Beispiel 99%),

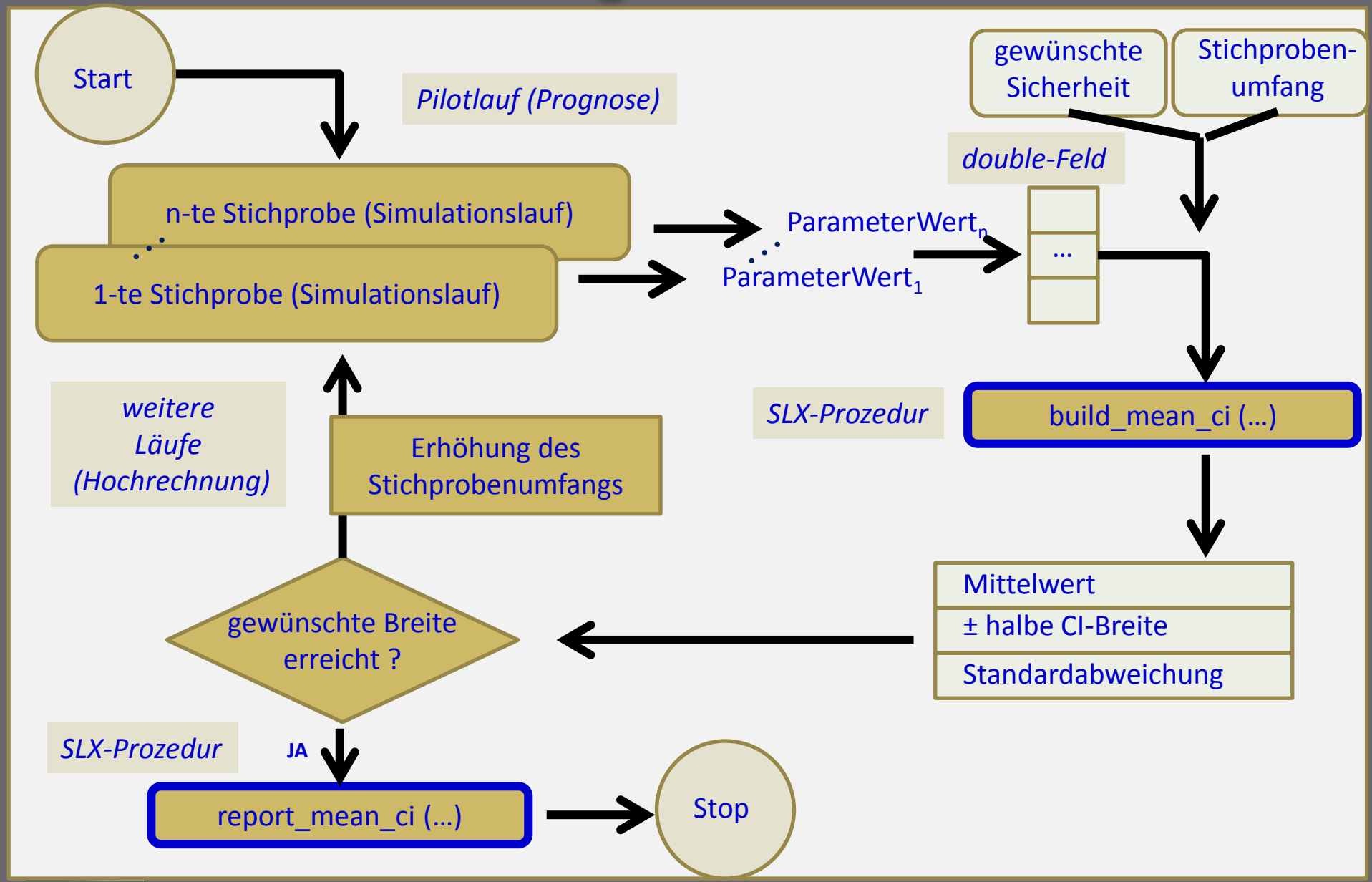
~ dem »Stichprobenumfang« sowie dem Standardfehler der Stichprobenstatistik

Beispiele

- Mittelwert
- Regressionskoeffizient
- Differenz zweier Wahrscheinlichkeiten
- Differenz zweier Mittelwerte

Zufallsstichproben
(realisiert über endliche Simulationsläufe)
erlauben Aussagen über eine unbekannte Grundgesamtheit
als gesicherte Schätzung

CI-Bestimmung in SLX: Datenfluss



Berechnung in SLX

- **build_mean_ci** (samples[*], scount, level, smean, stdev, half_width)

Eingabe

- **double**-Feld
  mit Parameterwerten
  (berechnet per Simulation)
- Stichproben-Nummer
  (**int**)~ Index
- Konfidenzkoeffizient
  (**double**), z.B.: 0,95

Ausgabe

- Stichprobenmittelwert
  (**double**)
- Standardabweichung der
  Stichprobe
  (**double**)
- halbe Breite des Konfidenzintervalls
  (**double**), z.B.: 0,95)

CI-Berechnung und Ausgabe in SLX

- **report_mean_ci** (title, level, samples[*], scount)

Eingabe
- Überschrift (**string**(*))
- Konfidenzkoeffizient (**double**), z.B.: 0,95
- Feld mit Parameterwerten (**double**)
- Stichprobennummer (**int**)

Anwendung im Bank-Beispiel

Konfidenz-Intervall-Schätzung für
- queuing_time (Wartezeit)
- elapsed_time (Verweilzeit)
- queue_length (WS-Länge)

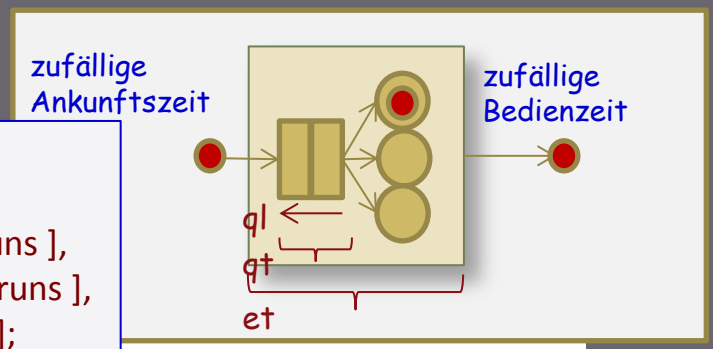
Stichproben-Erzeugung

100 Läufe

Ausgabe-Erweiterung

Replications: 100 95% C.I. 2.2204 +- 0.2107 Std Dev: 1.0619

Beispiel-Änderung




```
constant   integer     n_runs = 100 ;

           double      sample_elapsed_time [ 1 .. n_runs ],
                       sample_queueing_time [ 1 .. n_runs ],
                       sample_quelength [ 1 .. n_runs ];
```

… wird eingefügt

```
//***************
// Example EX0022
//***************
import <stats>
import <h7>
module basic  {
        rn_stream               arrive,
                                service ;
        random_variable         rv_elapsed_time,
                        rv_queueing_time histogram start=0.0 width=0.5 count = 20;
        random_variable (time) rv_quelength;
        storage clerk capacity=3;
        control int in_customer,
                    out_customer;
        int que_length;
        constant float close_time=8*60, service_time=1.3 ;
        boolean door_closed;

        class cl_customer {
        actions {
                        in_customer ++; // increment customer counter
                        que_length++;   // increment queue length
                        tabulate rv_quelength=que_length; // tabulate
                        enter clerk;       // try to catch a clerk
                        que_length --;   // decrement queue length
                        tabulate rv_quelength=que_length count = 0;
                        tabulate rv_queueing_time= time - ACTIVE->mark_time;
                        advance rv_expo ( service , service_time ); // service time
                        leave clerk;
                        out_customer ++;
                        tabulate rv_elapsed_time = time - ACTIVE->mark_time;
        } //actions
        } // cl_customer
```

```
procedure run_model () {
    float intensity=2.0;
    fork  { // Arriving Customer
            forever {
                    advance rv_expo ( arrive , 1/intensity );
                    activate new cl_customer;
                    if ( door_closed ) terminate;
            }
    }
    fork { // Controlling the bank
            advance close_time;
            door_closed = TRUE;
            terminate;
    }
    wait until ( (time > close_time) &&
                    ( in_customer == out_customer ));
    report ( system );
}
```

```
procedure main() {
    run_model();
} // main

}
```

… wird erweitert

… werden Prozedure werden eingefügt

```
//*****************************************************************************
// Example EX0023
//*****************************************************************************
...
}
procedure main() {
        for ( run=1; run<= n_runs; run++ ) {
                run_model();
                report_model();
                clear_model();
        }
}// main
```

```
constant integer    n_runs = 100 ;

float               sample_elapsed_time [ 1 .. n_runs ],
                    sample_queueing_time [ 1 .. n_runs ],
                    sample_quelength [ 1 .. n_runs ];
```

Makro für Mittelwertbestimmung einer beobachteten Zufallsgröße

```
procedure report_model () {
        print ( run ) "Run * finished \n";
    // Collecting data for every run
        sample_elapsed_time [ run ]   = sample_mean ( rv_elapsed_time );
        sample_queueing_time [ run ] = sample_mean ( rv_queueing_time );
        sample_quelength [ run ]       = sample_mean ( rv_quelength );

        if ( run ==n_runs ) {
                report ( system );
                print (run) "Confidence Intervals after * runs \n";
                report_mean_ci
                        ( "Elapsed Time ", 0.95, sample_elapsed_time , run );
                report_mean_ci
                        ( "Queueing Time ", 0.95, sample_queueing_time , run );
                report_mean_ci
                        ( "Average Quelength ", 0.95,sample_quelength , run );
        }
}
```

```
procedure clear_model() {
        clear system; // Clearing for SLX - features
        /* Clearing model specific variables */
        in_customer = 0;
        out_customer = 0;
        que_length = 0;
        door_closed = FALSE ;
}
```

Beispiel: SLX-Report

Confidence Intervals after 100 runs

Konfindenzintervall

Elapsed Time				
Replications: 100	95% C.I.	3.5150	+- 0.2167	Std Dev: 1.0920
Queueing Time				
Replications: 100	95% C.I.	2.2204	+- 0.2107	Std Dev: 1.0619
Average Quelength				
Replications: 100	95% C.I.	4.4085	+- 0.4298	Std Dev: 2.1654