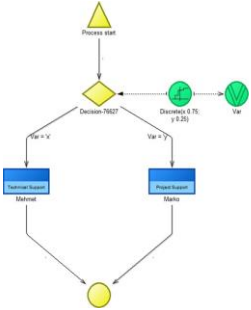
Simulation with ADOxx

Simulation Introduction

- **Real-World Process → Simulation**

Simulation is the reproduction of a real-process (e.g. business- process) over time. For simulating you have to create a model which represents your process and its characteristics. The model describes the system itself, where the simulation describes the operation of the system over time.



Simulation with ADOxx

The aim is to

- model
- study, and
- analysis

the behavior of a complex and dynamic system.

Algorithms

ADOxx provides the following predefined simulation algorithms:

1) Path Analysis:

2) Capacity Analysis:

3) Workload Analysis:
   - ➢ stationary
   - ➢ non-stationary

Algorithms

- Path Analysis (straight forward)
  - Simulation without working environment conditions
    - Expected values of cost and time
    - Critical Paths

- Capacity Analysis
  - Simulation with the assignment of activities to ’processors’
    - Evaluation of human requirement
    - Activity and process costs under personal cost condition

- Workload Analysis
  - Simulation on a time axis by daily calendar and time
    - Activity and process costs under personnel cost condition
    - Capacity plan with process and personnel calendar

Inputs and Outputs

- Path Analysis
  - Input: Process time and waiting time Weighted
  - Output: path results, mean values
- Capacity Analysis
  - Input: Quantity (global/time cycle), processor assignment Global
  - Output: capacity calculation, process costs
- Workload Analysis
  - Input: Amounts per day, attendance time
  - Output: dynamically evaluated capacity curve

General Modeling Conditions

- ∀ models: ∃! Startpoint S
- ∀ models: ∃ Endpoints E_i
- ∀ paths P from S to E_i: P is connected
- Matching Condition:

  Let D be a decision node. ∀ edges ei where D is ancestor
  $\sum P(e_i)$ = 1

Matching Condition & Variable Assignment

The above defined matching condition can be executed by the so called variable assignment. For this purpose, you can choose one of four different random variable distributions.

- Discrete
  - Variable name
  - Probability
- Normal
  - Expectation
  - Standard deviation
- Exponential
  - Expectation
- Uniform
  - Lower bound
  - Upper bound

Discrete(x 0.5; y 0.5)

Normal(50; 5)

Exponential(0,01)

Uniform(1; 5)

Excursion: Probability

Definition

A **Probability space** is a triple (Ω, F , P), where

1) Ω is the set of all possible outcomes or sample space.

2) F is a subset of Ω which satisfies the following three properties.

- $\emptyset \in F$
- $A \in F \rightarrow A^c \in F$
- $A_1, A_2, A_3, \ldots \in F \rightarrow \bigcup_{i=1}^{\infty} A_i \; U^{\infty} \in F$

3) P is the probability for each event A, where P fulfills the following three axioms

- $\forall A : P(A) \geq 0$
- $P(\Omega) = 1$
- If $A_1, A_2, \ldots$ is a sequence of pairwise disjoint events, then

$$P\left(\bigcup_{i=1}^{\infty} Ai\right) = \sum_{i=1}^{\infty} P(A_i)$$

Excursion: Probability

Definition

Let (Ω, F, P) be a probability space and $X : \Omega \rightarrow \Omega'$ feasible. We call X as a Ω' valued **random variable**.

Definition

Let $\Omega' = R$. The map $F : R \rightarrow [0, 1]$ which is defined by $F(t) = P(X \leq t)$ is called **distribution function** of the random variable X .

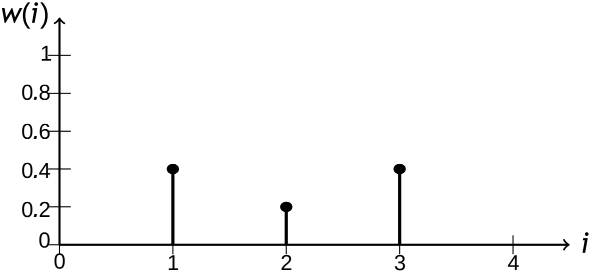
Definition

A random variable X is called **continuous**, if there exists an integrable function $f : R \rightarrow R^{+}$, so that $P(X \leq t) = \int_{\infty}^{t} f(x)d(x) \quad \forall t \in R$. We say f is the **probability density function** of X.

Discrete Distribution

Definition

A random variable *X* is called discrete, if the number of its values are finite or countably many. For i ∈ R we define w(i) = P(X = i), where R is the domain of *X* .

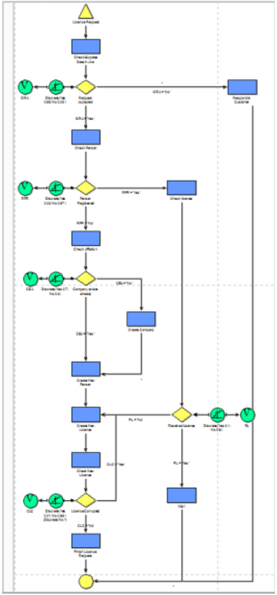


Application: Installation Code Generation Process

The Installation Code Process of ADOxx is a well-defined procedure which can be modeled like on the right hand-side.

The first decision is, if the request is accepted or not. This decision is assigned to a random variable, which is discretely distributed with

$$P(X = Yes) = 0.9 \quad P(X = No) = 0.1.$$



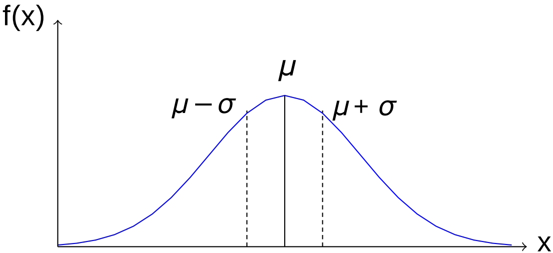
Normal Distribution

Definition

Let μ ∈ R and σ > 0. A random variable X with the domain R and the Probability density

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

is called normal distributed or N(μ, σ)−distributed.



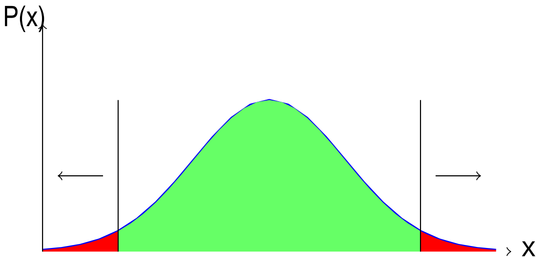
Application: Outlier Tests

The so called maximum normed residual test is a statistical test used to detect outliers in a univariate data set assumed to come from a normally distributed population. There are three kinds of outlier tests:

(i) One sided outlier

- High outlier
- Low outlier

(ii) Two sided outlier: high and low



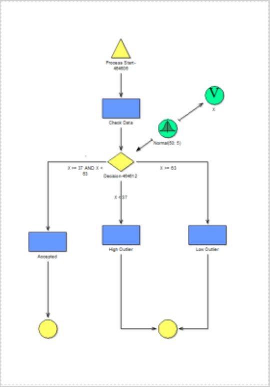
Application: Two Sided Outlier Model

In a physical experiment the outcomes are normally distributed with mean value μ and the standard deviation σ.

The values $X < \mu - \delta$ and $X > \mu + \epsilon$ are outliers.

Condition:

If the probability $P(\mu - \delta \leq X \leq \mu + \epsilon) < 45\%$
then the experiment and so the thesis has failed.



Exponential Distribution

Definition

Let λ > 0 and R^+ be the support of a continuous random variable X. We call X exponential distributed or if its probability density function is

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x \in R^+ \\ 0 & \text{else.} \end{cases}$$

Therefore the distribution function is

$$F(x) = \begin{cases} 1-e^{-\lambda x} & x \in R^+ \\ 0 & \text{else.} \end{cases}$$

Exponential Distribution

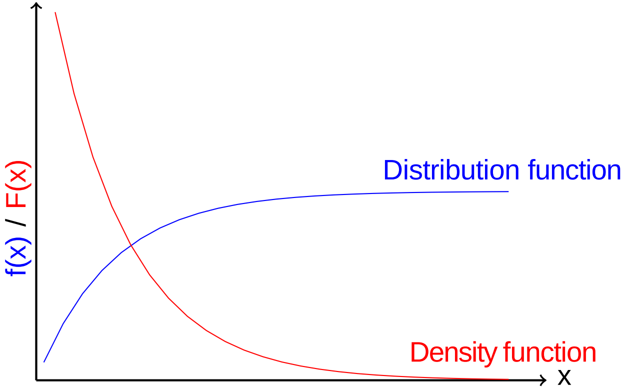


Figure : Exponential Distribution

Application: Call Center Simulation

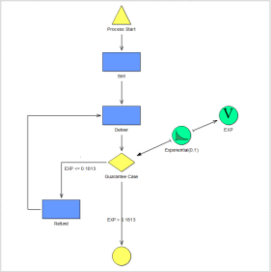
A very common example for the exponential distribution is the call center simulation. In call center models we can define the income process with the exponential-function. The time between two calls in the call center is exponential distributed.

For example let $\lambda = 0,4$ ($\rightarrow \mu = 2,5$). The probability that between two Calls elapse 2 min is therefore

$$P(X \leq 2) = 1 - e^{-0,4*2} = 0,5507$$

Application: Lifecycle

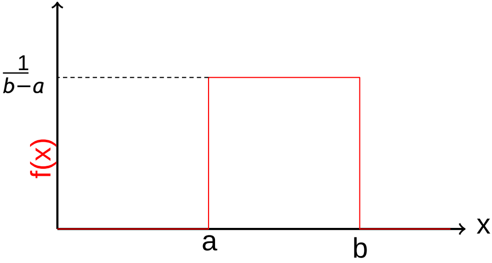
The probability of failure of an electronic component is exponential distributed where its expected durability is about 10 years. If the component does not work within 2 years any more the producer has to refund it because of the guarantee conditions.



Uniform Distribution

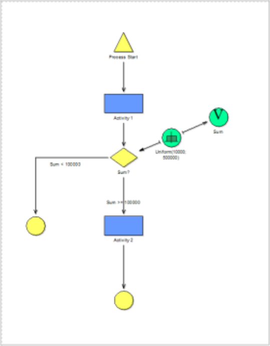
The **uniform distribution** is a distribution, that has constant probability. The density function is defined by

$$P(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & \text{else.} \end{cases}$$



Uniform Distribution: Application

Suppose that an insurance company sells a product that from an amount insured of EUR 100.000.− they have to support an additional activity. The product has an insurance volume between EUR 10.000.− and 500.000 amount insured. The sum of all the insurance contracts are distributed uniformly in this interval.



Path Analysis

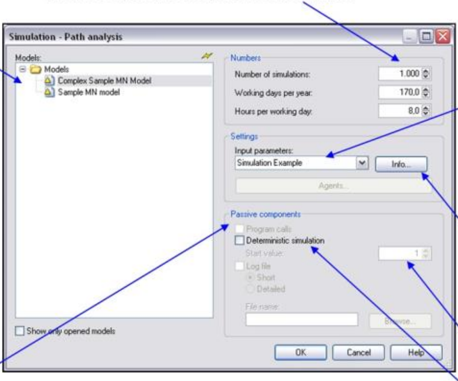
Path Analysis

Select/enter the values you need for the path analysis simulation.

Select the model you want to simulate

Indicate how many processes are to be "run through". The number of simulations selected affects the accuracy - the higher the number, the more exact the simulation results will be.

Select the input parameter combination you want to work with. The input parameters are defined in the „Simmapping“ library attribute of the dynamic library in your metamodel, using the „SIMOPTION“ keyword. (see above slides of Simmapping)



Simulation - Path analysis

Models:
- ⊟ Models
  - Complex Sample MN Model
  - Sample MN model

□ Show only opened models

Numbers
- Number of simulations: 1.000
- Working days per year: 170.0
- Hours per working day: 8.0

Settings
- Input parameters: Simulation Example
- Info...
- Agents...

Passive components
- □ Program calls
- □ Deterministic simulation
- Start value: 1
- □ Log file
  - ● Short
  - ○ Detailed
- File name:
- Browse...

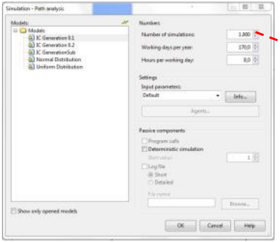
OK | Cancel | Help

Display information about the selected input parameter combination (SIMOPTION).

"Program calls" (default setting: deactivated)
If this option is activated, program calls specific to the activity will be carried out during the simulation of each activity. The selected input parameter combination will determine which program calls will be concerned by this .

"Deterministic simulation„ (default setting: disabled)
When enabling this option the simulation is initialised with the same start value. This ensures that with the same start value independent simulation runs will determine the same simulation results.

Path Analysis



Simulation - Path analysis

Models:
- Models
  - IC Generation 0.1
  - IC Generation 0.2
  - IC GenerationSub
  - Normal Distribution
  - Uniform Distribution

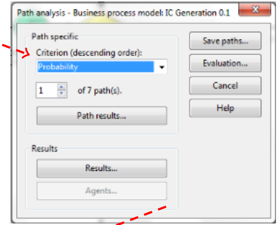
□ Show only opened models

Numbers
- Number of simulations: 1.000
- Working days per year: 170,0
- Hours per working day: 8,0

Settings
- Input parameters: Default

Passive components
- □ Program calls
- □ Deterministic simulation
- Derivation: 1
- □ Log file
  - ● Short
  - ○ Detailed
- File name:

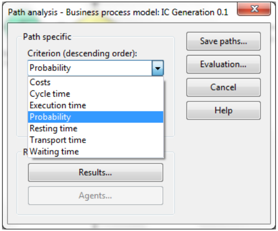
1. Select number of Simulations



Path analysis - Business process model: IC Generation 0.1

Path specific
- Criterion (descending order): Probability
- 1 of 7 path(s).

Results

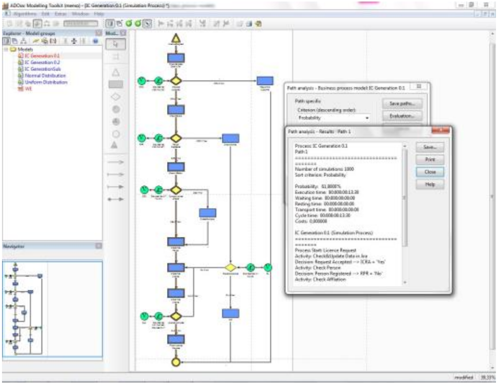


Path analysis - Business process model: IC Generation 0.1

Path specific
- Criterion (descending order): Probability

2. Order outputs by criteria you want

Results



Path analysis - Business process model: IC Generation 0.1

Path specific

Criterion (descending order): Probability

Path analysis - Results : Path 1

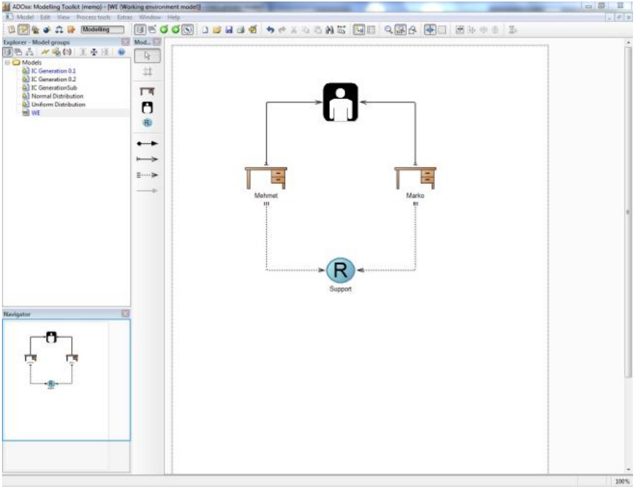
```
Process: IC Generation 0.1
Path 1
=====================================
=======
Number of simulations: 1000
Sort criterion: Probability

Probability:  51,8000%
Execution time: 00:000:00:13:20
Waiting time: 00:000:00:00:00
Resting time: 00:000:00:00:00
Transport time: 00:000:00:00:00
Cycle time: 00:000:00:13:20
Costs: 0,000000

IC Generation 0.1 (Simulation Process)
=====================================
=======
Process Start: Licence Request
Activity: Check&Update Data in Jira
Decision: Request Accepted --> ICRA = 'Yes'
Activity: Check Person
Decision: Person Registered --> RPR = 'No'
Activity: Check Affiliation
```

- Select any path you want and click ”OK” to display information of it. The selected path will be marked on your model.
- The simulation results can be
  - saved and
  - printed.

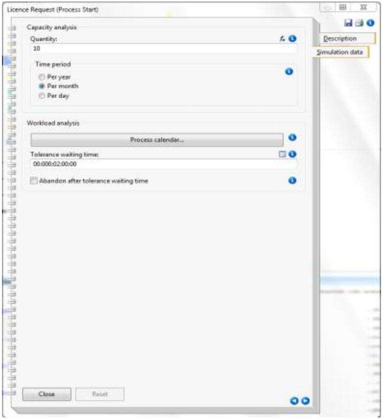
Capacity Analysis

Working Environment

- ○ Create a working environment model



Capacity Analysis



Licence Request (Process Start)

Capacity analysis

Quantity:
10

Time period

○ Per year
● Per month
○ Per day

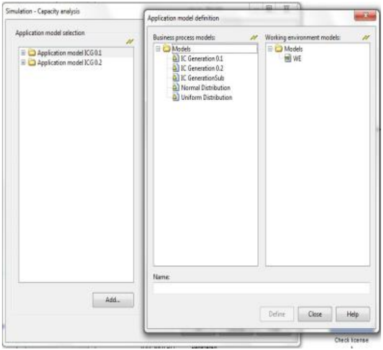
Workload analysis

Tolerance waiting time:
00:000:02:00:00

□ Abandon after tolerance waiting time

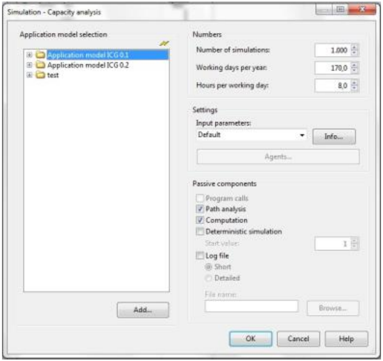
- Open Notebook of the bp start-object
- Go to chapter ’Simulation Data’
- Insert the simulation amount per
  - year,
  - month, or
  - day.

Capacity Analysis



- Go to Capacity Analysis
- Create a new application library consisting of:
  - at least one business process model and
  - exactly one working environment model.

Capacity Analysis



Simulation - Capacity analysis

Application model selection

- Application model ICG 0.1
- Application model ICG 0.2
- test

Numbers

Number of simulations: 1.000

Working days per year: 170,0

Hours per working day: 8,0

Settings

Input parameters: Default

Passive components

□ Program calls

☑ Path analysis

☑ Computation

□ Deterministic simulation

Start value: 1

□ Log file

● Short

○ Detailed

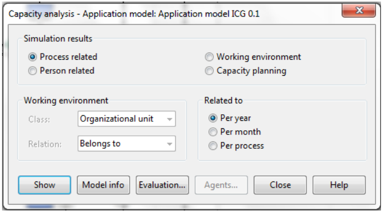
File name:

Select:

- Application Model
- Numbers
  - Number of Simulations
  - Working days per year
  - Hours per working day
- Settings
- Passive components
  - Path analysis
  - Computation
  - Deterministic simulation
  - Log file

Results

- Select criteria after that the results should be ordered.



Capacity analysis - Application model: Application model ICG 0.1

Simulation results

● Process related
○ Working environment
○ Person related
○ Capacity planning

Working environment

Class: Organizational unit

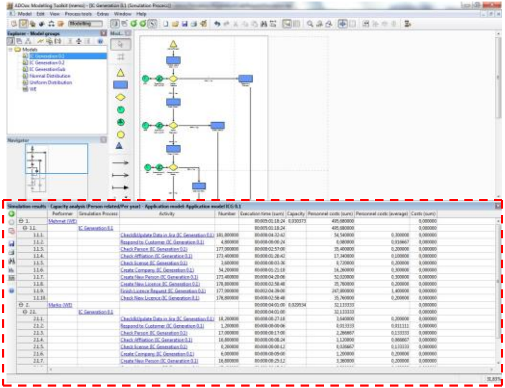
Relation: Belongs to

Related to

● Per year
○ Per month
○ Per process

- Simulation Results
  - Process related
  - Person related
  - Working environment*
  - Capacity planning
- *Working Environment
  - Class
  - Relation
- Related to
  - Per year
  - Per month
  - Per process

Results



Simulation results - Capacity analysis (Person related/Per year) - Application model: Application model ICG 0.1

	Performer	Simulation Process	Activity	Number	Execution time (sum)	Capacity	Personnel costs (sum)	Personnel costs (average)	Costs (sum)
1.	Mehmet (WE)				00:005:01:18:24	0,030373	405,680000		0,000000
1.1.		IC Generation 0.1			00:005:01:18:24		405,680000		0,000000
1.1.1.			Check&Update Data in Jira (IC Generation 0.1)	181,800000	00:008:04:32:42		54,540000	0,300000	0,000000
1.1.2.			Respond to Customer (IC Generation 0.1)	4,800000	00:008:08:06:24		0,080000	0,016667	0,000000
1.1.3.			Check Person (IC Generation 0.1)	177,000000	00:008:02:57:00		35,400000	0,200000	0,000000
1.1.4.			Check Affiliation (IC Generation 0.1)	173,400000	00:008:01:26:42		17,340000	0,100000	0,000000
1.1.5.			Check license (IC Generation 0.1)	3,600000	00:008:08:03:36		0,720000	0,200000	0,000000
1.1.6.			Create Company (IC Generation 0.1)	54,200000	00:008:01:21:18		16,260000	0,300000	0,000000
1.1.7.			Create New Person (IC Generation 0.1)	173,400000	00:008:04:28:08		52,020000	0,300000	0,000000
1.1.8.			Create New Licence (IC Generation 0.1)	178,800000	00:008:02:58:48		35,760000	0,200000	0,000000
1.1.9.			Finish Licence Request (IC Generation 0.1)	177,000000	00:002:04:36:00		247,800000	1,400000	0,000000
1.1.10.			Check New Licence (IC Generation 0.1)	178,800000	00:008:02:58:48		35,760000	0,200000	0,000000
2.	Marko (WE)				00:008:04:01:00	0,029534	32,133333		0,000000
2.1.		IC Generation 0.1			00:008:04:01:00		32,133333		0,000000
2.1.1.			Check&Update Data in Jira (IC Generation 0.1)	18,200000	00:008:08:27:18		3,640000	0,200000	0,000000
2.1.2.			Respond to Customer (IC Generation 0.1)	1,200000	00:008:08:00:06		0,013333	0,011111	0,000000
2.1.3.			Check Person (IC Generation 0.1)	17,000000	00:008:08:17:00		2,266667	0,133333	0,000000
2.1.4.			Check Affiliation (IC Generation 0.1)	16,800000	00:008:08:08:24		1,120000	0,066667	0,000000
2.1.5.			Check license (IC Generation 0.1)	0,200000	00:008:08:08:12		0,026667	0,133333	0,000000
2.1.6.			Create Company (IC Generation 0.1)	6,000000	00:008:08:09:00		1,200000	0,200000	0,000000
2.1.7.			Create New Person (IC Generation 0.1)	16,800000	00:008:08:25:12		3,360000	0,200000	0,000000

31,83%

The simulation results can

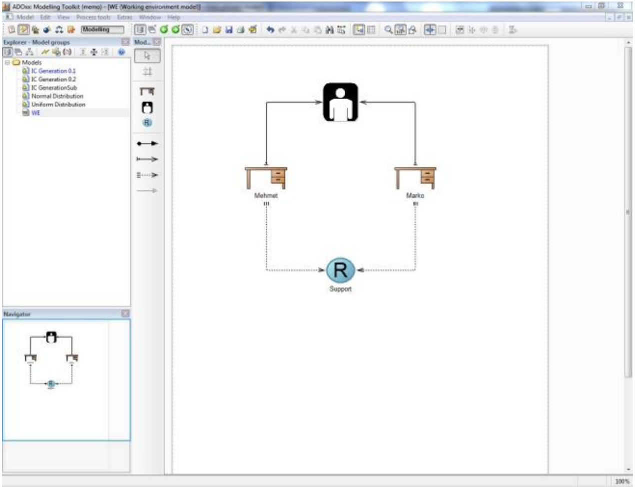
- saved,
- printed,
- displayed as diagrams, and
- compared.

Workload Analysis

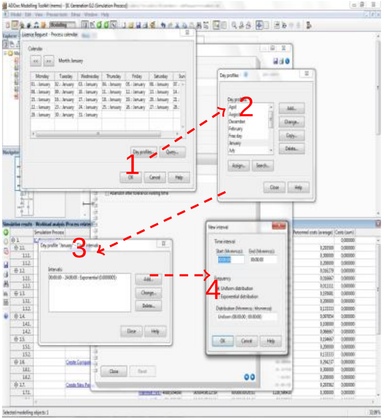
(Steady State)

Working Environment

- Create a working environment model



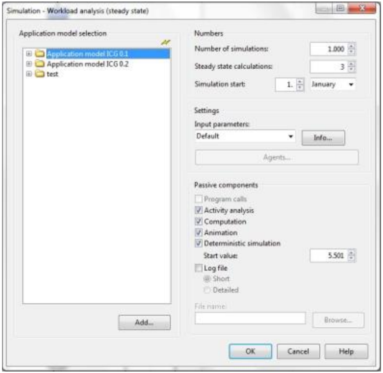
Workload Analysis: Process Calendar



Define process calendar of the bp-start instance:

1) Go to 'Day profiles'
2) Add Day profiles
3) Add assign time interval to the day profile
4) Define time interval
   - Uniform distributed
     (Process is triggered e.g. every 5 minutes)
   - Exponential distributed
     (The probability between two process starts is exponential distributed)

Workload Analysis



Simulation - Workload analysis (steady state)

Application model selection

- Application model ICG 0.1
- Application model ICG 0.2
- test

Numbers

Number of simulations: 1.000

Steady state calculations: 3

Simulation start: 1. January

Settings

Input parameters: Default

Passive components

□ Program calls

☑ Activity analysis

☑ Computation

☑ Animation

☑ Deterministic simulation

Start value: 5.501

□ Log file

● Short

○ Detailed

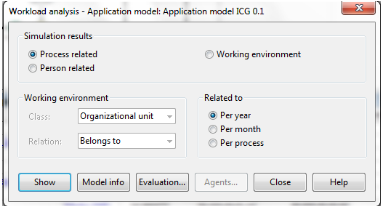
File name:

Select:

- Application Model
- Numbers
  - Number of Simulations
  - Steady state calculation
  - Simulation start
- Settings
- Passive components
  - Activity analysis
  - Computation
  - Animation
  - Deterministic simulation
  - Log file

Results

- Select criteria after that the results should be ordered.



Workload analysis - Application model: Application model ICG 0.1

Simulation results

● Process related ○ Working environment

○ Person related

Working environment

Class: Organizational unit

Relation: Belongs to

Related to

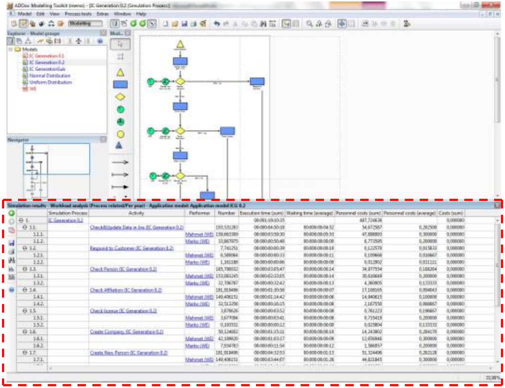
● Per year

○ Per month

○ Per process

- Simulation Results
  - Process related
  - Person related
  - Working environment*
- *Working Environment
  - Class
  - Relation
- Related to
  - Per year
  - Per month
  - Per process

Results



Simulation results - Workload analysis (Process related/Per year) - Application model: Application model ICG 0.2

	Simulation Process	Activity	Performer	Number	Execution time (sum)	Waiting time (average)	Personnel costs (sum)	Personnel costs (average)	Costs (sum)
1.	IC Generation 0.2				00:001:19:10:35		487,724638		0,000000
1.1.		Check&Update Data in Ins (IC Generation 0.2)		193,531283	00:000:04:50:18	00:000:00:04:52	54,672587	0,282500	0,000000
1.1.1.			Mehmet (WE)	159,663389	00:000:03:59:30	00:000:00:05:53	47,898893	0,300000	0,000000
1.1.2.			Marko (WE)	33,867975	00:000:00:50:48	00:000:00:00:00	6,773595	0,200000	0,000000
1.2.		Respond to Customer (IC Generation 0.2)		7,741251	00:000:00:00:39	00:000:00:00:18	0,122579	0,015833	0,000000
1.2.1.			Mehmet (WE)	6,580064	00:000:00:00:33	00:000:00:00:11	0,109668	0,016667	0,000000
1.2.2.			Marko (WE)	1,161188	00:000:00:00:06	00:000:00:00:00	0,012902	0,011111	0,000000
1.3.		Check Person (IC Generation 0.2)		185,790032	00:000:03:05:47	00:000:00:00:14	34,977554	0,188264	0,000000
1.3.1.			Mehmet (WE)	153,083245	00:000:02:33:05	00:000:00:00:14	30,616649	0,200000	0,000000
1.3.2.			Marko (WE)	32,706787	00:000:00:32:42	00:000:00:00:13	4,360905	0,133333	0,000000
1.4.		Check Affiliation (IC Generation 0.2)		181,919406	00:000:01:30:58	00:000:00:00:07	17,108165	0,094043	0,000000
1.4.1.			Mehmet (WE)	149,406151	00:000:01:14:42	00:000:00:00:06	14,940615	0,100000	0,000000
1.4.2.			Marko (WE)	32,513256	00:000:00:16:15	00:000:00:00:08	2,167550	0,066667	0,000000
1.5.		Check license (IC Generation 0.2)		3,870626	00:000:00:03:52	00:000:00:00:08	0,761223	0,196667	0,000000
1.5.1.			Mehmet (WE)	3,677094	00:000:00:03:41	00:000:00:00:08	0,735419	0,200000	0,000000
1.5.2.			Marko (WE)	0,193531	00:000:00:00:12	00:000:00:00:00	0,025804	0,133333	0,000000
1.6.		Create Company (IC Generation 0.2)		50,124602	00:000:01:15:11	00:000:00:00:18	14,243802	0,284170	0,000000
1.6.1.			Mehmet (WE)	42,189820	00:000:01:03:17	00:000:00:00:09	12,656946	0,300000	0,000000
1.6.2.			Marko (WE)	7,934783	00:000:00:11:54	00:000:00:00:12	1,586857	0,200000	0,000000
1.7.		Create New Person (IC Generation 0.2)		181,919406	00:000:04:32:53	00:000:00:02:13	51,324496	0,282128	0,000000
1.7.1.			Mehmet (WE)	149,406151	00:000:03:44:07	00:000:00:01:26	44,821845	0,300000	0,000000

33,99%

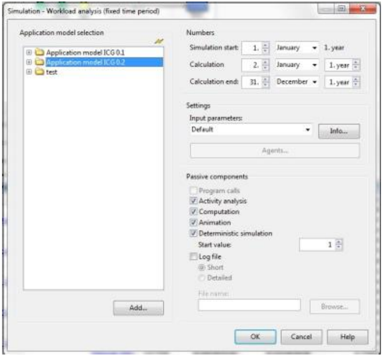
The simulation results can be

- saved,
- printed,
- displayed as diagrams, and
- compared.

Workload Analysis

(Fixed Time Period)

Workload Analysis



Simulation - Workload analysis (fixed time period)

Application model selection

- Application model ICG 0.1
- Application model ICG 0.2
- test

Numbers

Simulation start: 1. January 1. year

Calculation 2. January 1. year

Calculation end: 31. December 1. year

Settings

Input parameters: Default

Passive components

- □ Program calls
- ☑ Activity analysis
- ☑ Computation
- ☑ Animation
- ☑ Deterministic simulation

Start value: 1

- □ Log file
  - ● Short
  - ○ Detailed

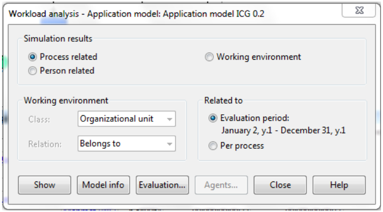
File name:

Select:

- Application Model
- Numbers
  - Simulation start
  - Calculation
  - Calculation end
- Settings
- Passive components
  - Activity analysis
  - Computation
  - Animation
  - Deterministic simulation
  - Log file

Results

- Select criteria after that the results should be ordered.



Workload analysis - Application model: Application model ICG 0.2

Simulation results

● Process related
○ Person related
○ Working environment

Working environment

Class: Organizational unit

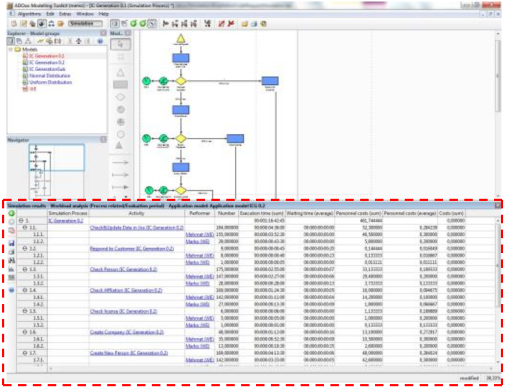
Relation: Belongs to

Related to

● Evaluation period:
January 2, y.1 - December 31, y.1
○ Per process

- Simulation Results
  - Process related
  - Person related
  - Working environment*
- *Working Environment
  - Class
  - Relation
- Related to
  - Evaluation period
  - Per process

Results



Simulation results - Workload analysis (Process related/Evaluation period) - Application model: Application model ICG 0.2

	Simulation Process	Activity	Performer	Number	Execution time (sum)	Waiting time (average)	Personnel costs (sum)	Personnel costs (average)	Costs (sum)
⊖ 1.	IC Generation 0.2				00:001:16:42:45		461,744444		0,000000
⊖ 1.1.		Check&Update Data in ... (IC Generation 0.2)		184,000000	00:008:04:38:00	00:000:00:00:00	52,300000	0,284239	0,000000
1.1.1.			Mehmet (WE)	155,000000	00:008:03:52:30	00:000:00:00:00	46,500000	0,300000	0,000000
1.1.2.			Marko (WE)	29,000000	00:008:00:43:30	00:000:00:00:00	5,800000	0,200000	0,000000
⊖ 1.2.		Respond to Customer (IC Generation 0.2)		9,000000	00:008:00:08:45	00:000:00:00:20	0,144444	0,016049	0,000000
1.2.1.			Mehmet (WE)	8,000000	00:008:00:08:40	00:000:00:00:23	0,133333	0,016667	0,000000
1.2.2.			Marko (WE)	1,000000	00:008:00:00:05	00:000:00:00:00	0,011111	0,011111	0,000000
⊖ 1.3.		Check Person (IC Generation 0.2)		175,000000	00:008:02:55:00	00:000:00:00:07	33,133333	0,189333	0,000000
1.3.1.			Mehmet (WE)	147,000000	00:008:02:27:00	00:000:00:00:06	29,400000	0,200000	0,000000
1.3.2.			Marko (WE)	28,000000	00:008:00:28:00	00:000:00:00:13	3,733333	0,133333	0,000000
⊖ 1.4.		Check Affiliation (IC Generation 0.2)		168,000000	00:008:01:24:30	00:000:00:00:05	16,000000	0,094675	0,000000
1.4.1.			Mehmet (WE)	142,000000	00:008:01:11:00	00:000:00:00:04	14,200000	0,100000	0,000000
1.4.2.			Marko (WE)	27,000000	00:008:00:13:30	00:000:00:00:09	1,800000	0,066667	0,000000
⊖ 1.5.		Check license (IC Generation 0.2)		6,000000	00:008:00:06:00	00:000:00:00:00	1,133333	0,188889	0,000000
1.5.1.			Mehmet (WE)	5,000000	00:008:00:05:00	00:000:00:00:00	1,000000	0,200000	0,000000
1.5.2.			Marko (WE)	1,000000	00:008:00:01:00	00:000:00:00:00	0,133333	0,133333	0,000000
⊖ 1.6.		Create Company (IC Generation 0.2)		48,000000	00:008:01:12:00	00:000:00:00:16	13,100000	0,272917	0,000000
1.6.1.			Mehmet (WE)	35,000000	00:008:00:52:30	00:000:00:00:09	10,500000	0,300000	0,000000
1.6.2.			Marko (WE)	13,000000	00:008:00:18:30	00:000:00:00:15	2,600000	0,200000	0,000000
⊖ 1.7.		Create New Person (IC Generation 0.2)		168,000000	00:008:04:13:30	00:000:00:00:06	48,000000	0,284024	0,000000
1.7.1.			Mehmet (WE)	142,000000	00:008:03:33:00	00:000:00:00:05	42,600000	0,300000	0,000000

The simulation results can be

- saved,
- printed,
- displayed as diagrams, and
- compared.

How to Realize Simulation in ADOxx

The standard parameters for the simulation algorithms need to be configured according to the classes defined in the dynamic and the static library to be simulated.

1) Path Analysis:
simulates a dynamic model alone.

2) Capacity Analysis:
Simulates a dynamic model & the corresponding static model(s)

3) Workload Analysis:
Simulates a dynamic model & the corresponding static model(s)

How to Realize Simulation in ADOxx

Modeling Language Definition

In order to perform simulation at first we need a **dynamic** model with the following classes:

- A class that will represent the initial point of the model, derived from the abstract class __Start__
- A class that will represent the final point of the model, derived from __D_end__
- One or more classes that will represend the active objects of the model, derived from __Activity__
- A class that will represent the points of decision in the model, derived from __Decision__
- A class that will define the variables in the model derived from __D_variable__
- A class that will define the random generator derived from __D_random_generator__
- The “Subsequent“, “Sets variable“, and “Sets“ relation, already defined in the ADOxx metamodel, for connecting the objects or defining random generators.

How to Realize Simulation in ADOxx

For the workload- and the capacity analysis we have also to define a working environment in the **static** library with the following classes:

- “Performer“derived from __S_person__
- “Organizational unit“ derived from __S_group__
- “Role“derived from __S_group__

and relation classes:

- “Belongs to“: <Performer> → <Organizational unit>
- “Is manager“: <Performer> → <Organizational unit>
- “Has role“: <Performer> → <Role>
- “has Cross-reference“: <__S-construct__ > → <__S-construct__>

How to Realize Simulation in ADOxx

Dynamic Library

In the dynamic library we have to change the following library attributes (available in the Simulation tab):

- ”Simtext” contains some user-specific expressions used by ADOxx to label simulation results
- ”Simmapping” contains the definition of the input sets for the Simulation and a group of classes which are then used in simulation-related Actions.
- ”Sim result mapping” defines which simulation results are written back into which attributes of a model when you click on the ”Evaluation” button.

Static Library

In the static library we have to adapt the ”Sim result mapping”-attribute.

How to Realize Simulation in ADOxx

SIMTEXT:

Simtext is used in all four algorithms for labeling of simulation results.

SYNTAX:

Simtext:	**SIMTEXT undefined** [1] | Settings
Settings:	**bp:** "term for <business process>" **cycletime:** "term for <cycle time>" **activity:** "term for <activity>" **number:** "term for <number (count)>" **actor:** "term for <person>" **perscost:** "term for <personnel costs>" **resource:** "term for <resource>" **rescost:** "term for <resource costs>"

[1] "undefined" causes the Simtext to be ignored.

How to Realize Simulation in ADOxx

Simmapping:

Simtext allows the definition of input sets for the simulation and for the analytic evaluation. Additionally it specifies a set of classes which is used for simulation related actions.

SYNTAX

SimOption :

SIMOPTION [invalid] name: ”option name”

  activity: ”name of activity-class”

  [**executiontime:** ”attribute name of execution time”] [

  waitingtime: ”attribute name of waiting time”]

  [**restingtime:** ”attribute name of resting time”]

  [**transporttime:** ”attribute name of transport time”] [

  userattribute-1: ”additional attribute name 1”]

  ...

  [PerformerAssignment (for Subprocesses)]

  *{*SimActions*}*

How to Realize Simulation in ADOxx

PerformerAssignment (for Subprocesses):

processcall: ”class name of subprocess call”
subperformerattr: ”attribute name of default performer assignment for subprocesses”

SimActions :

ACTION
class: ”class name”
attribute: ”attribute name”
[**event:** **start** | **interrupt** | **continue** | **finish**]

SimClasses:

SIMCLASSES
bp-all | **bp-none**
[**bp-1:** ”bp class name”
...
bp-n: ”bp class name”]
we-all | **we-none**
[**we-1:** ”we class name”
...
we-n: ”we class name”]

How to Realize Simulation in ADOxx

Sim result mapping (dynamic):

The attribute ”Sim result mapping” defines which simulation results are written back into which attributes of a model within the evaluation.

```
PROCESSSTART ”Process Start” fixedinfo:”Info on
      results” fixedcycletime:”Aggregated cycle time”
      fixedpersonalcosts:”Aggregated personnel
costs”
FROMCLASS ”Activity” fromattribute:”Costs”
      resultatatribute:”Aggregated costs”
...
```

PROCESSSTART is a keyword used for assigning the name of the class that represents the starting point of the model that you want to simulate.

FROMCLASS is a keyword used for selecting additional classes (FromClassname) and specify values from the fromattribute attribute values (FromAttributename) specified. The selected attributes of this class can be transferred back through toattribute into the respective attribute (ToAttributename).

```
ACTIVITY ”Activity” fixedinfo:”Info
      on results”
      fixednumber:”Number”
      fixedpersonalcosts:”Aggregated personnel
costs”
...
```

ACTIVITY is a keyword used for assigning the name of the main class used in the model.

How to Realize Simulation in ADOxx

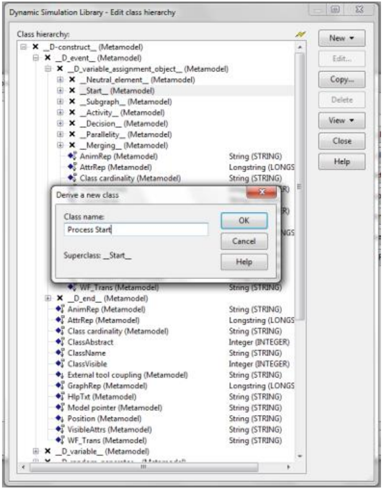
Sim result mapping (static):

The parameters of the static library attributes has also to be defined by editing the following into the ”Sim result mapping” attribute:

```
[PERSON ”Name_of_person_class”
       [fixedinfo:”Name_of_info_attribute”]
       [fixedworkload:”Name_of_workload_attribute”]
       [fixedcapacity:”Name_of_capacity_attribute”]
       [fixedpersonalcosts:”Name_of_personalcosts_attribute”]
{ FROMCLASS ”Name of fromclass”
       fromattribute:”Name_of_fromattribute”
       toattribute:”Name_of_toattribute” } ]
```

Simulation with ADOxx HANDS-ON

HANDS-On: Create Dynamic Classes



Class hierarchy:

```
⊟ ✕ __D-construct__ (Metamodel)
  ⊟ ✕ __D_event__ (Metamodel)
    ⊟ ✕ __D_variable_assignment_object__ (Metamodel)
      ⊞ ✕ __Neutral_element__ (Metamodel)
      ⊞ ✕ __Start__ (Metamodel)
      ⊞ ✕ __Subgraph__ (Metamodel)
      ⊞ ✕ __Activity__ (Metamodel)
      ⊞ ✕ __Decision__ (Metamodel)
      ⊞ ✕ __Parallelity__ (Metamodel)
      ⊞ ✕ __Merging__ (Metamodel)
        AnimRep (Metamodel)                 String (STRING)
        AttrRep (Metamodel)                 Longstring (LONGS
        Class cardinality (Metamodel)       String (STRING)
        WF_Trans (Metamodel)                String (STRING)
    ⊞ ✕ __D_end__ (Metamodel)
      AnimRep (Metamodel)                   String (STRING)
      AttrRep (Metamodel)                   Longstring (LONGS
      Class cardinality (Metamodel)         String (STRING)
      ClassAbstract                         Integer (INTEGER)
      ClassName                             String (STRING)
      ClassVisible                          Integer (INTEGER)
      External tool coupling (Metamodel)    String (STRING)
      GraphRep (Metamodel)                  Longstring (LONGS
      HlpTxt (Metamodel)                    String (STRING)
      Model pointer (Metamodel)             String (STRING)
      Position (Metamodel)                  String (STRING)
      VisibleAttrs (Metamodel)              String (STRING)
      WF_Trans (Metamodel)                  String (STRING)
  ⊞ ✕ __D_variable__ (Metamodel)
```

Derive a new class

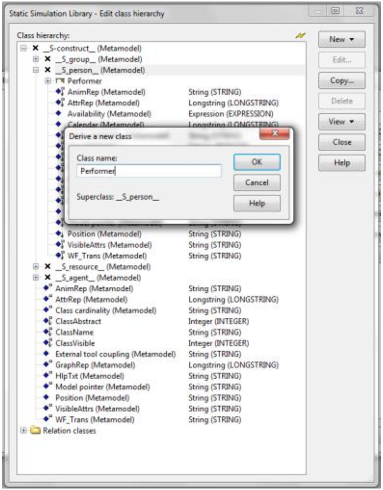
Class name: Process Start

Superclass: __Start__

- Open the ”Class hierarchy” for the Dynamic library.
- Activate the ”Metamodel” view then ”class hierarchy”
- Create the following classes

  1) ”Process Start” derived from __Start__
  2) ”Activity” derived from __Activity__
  3) ”Decision” derived from __Decision__
  4) ”Variable” derived from __Variable__
  5) ”Random Generator” derived from __Random_generator__
  6) ”End” derived from __D_end__

HANDS-On: Create Static Classes



Static Simulation Library - Edit class hierarchy

Class hierarchy:

```
__S-construct__ (Metamodel)
  __S_group__ (Metamodel)
  __S_person__ (Metamodel)
    Performer
    AnimRep (Metamodel)                 String (STRING)
    AttrRep (Metamodel)                 Longstring (LONGSTRING)
    Availability (Metamodel)            Expression (EXPRESSION)
    Calendar (Metamodel)                Longstring (LONGSTRING)
    Position (Metamodel)                String (STRING)
    VisibleAttrs (Metamodel)            String (STRING)
    WF_Trans (Metamodel)                String (STRING)
  __S_resource__ (Metamodel)
  __S_agent__ (Metamodel)
  AnimRep (Metamodel)                   String (STRING)
  AttrRep (Metamodel)                   Longstring (LONGSTRING)
  Class cardinality (Metamodel)         String (STRING)
  ClassAbstract                         Integer (INTEGER)
  ClassName                             String (STRING)
  ClassVisible                          Integer (INTEGER)
  External tool coupling (Metamodel)    String (STRING)
  GraphRep (Metamodel)                  Longstring (LONGSTRING)
  HlpTxt (Metamodel)                    String (STRING)
  Model pointer (Metamodel)             String (STRING)
  Position (Metamodel)                  String (STRING)
  VisibleAttrs (Metamodel)              String (STRING)
  WF_Trans (Metamodel)                  String (STRING)
Relation classes
```

Derive a new class

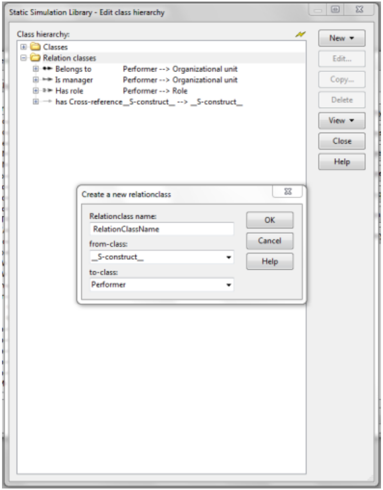
Class name: Performer

Superclass: __S_person__

- ○ Open the "Class hierarchy" for the Static library.
- ○ Activate the "Metamodel" view then "class hierarchy"
- ○ Create the following classes

  1) "Performer" derived from „__S_Person__"
  2) "Organizational unit" derived from „__S_Group__"
  3) "Role" derived from „__S_Group__"

HANDS-On: Create Relation Classes



Static Simulation Library - Edit class hierarchy

Class hierarchy:

- Classes
- Relation classes
  - Belongs to   Performer --> Organizational unit
  - Is manager   Performer --> Organizational unit
  - Has role   Performer --> Role
  - has Cross-reference__S-construct__ --> __S-construct__

Create a new relationclass

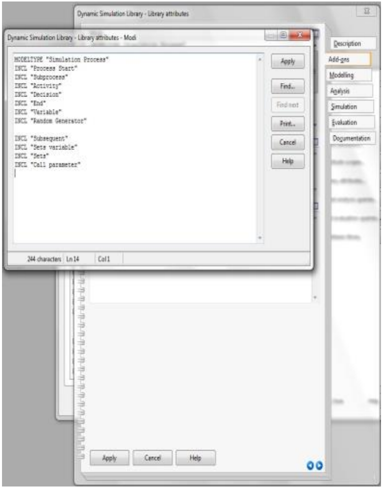
Relationclass name: RelationClassName

from-class: __S-construct__

to-class: Performer

- Open the ”Class hierarchy” for the Static library.
- Create the following relation classes
  1) “Belongs to“: <Performer> → <Organizational unit>
  2) “Is manager“:<Performer> → <Organizational unit>
  3) “Has role“:<Performer> → <Role>
  4) “has Cross-reference“: <__S-construct__> → <__S-construct__>

Define Dynamic Model Type

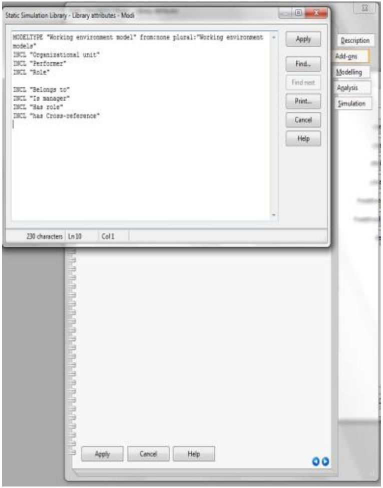


```
MODELTYPE "Simulation Process"
INCL "Process Start"
INCL "Subprocess"
INCL "Activity"
INCL "Decision"
INCL "End"
INCL "Variable"
INCL "Random Generator"

INCL "Subsequent"
INCL "Sets variable"
INCL "Sets"
INCL "Call parameter"
```

- ○ Open dynamic-library attributes
- ○ Select Add-ons
- ○ Go to Modi and define:

MODELTYPE ”Simulation Process”
INCL ”Process Start”
INCL ”Subprocess”
INCL ”Activity”
INCL ”Decision”
INCL ”End”
INCL ”Variable”
INCL ”Random Generator”
INCL ”Subsequent”
INCL ”Sets variable”
INCL ”Sets”
INCL ”Call parameter”

Define Static Model Type



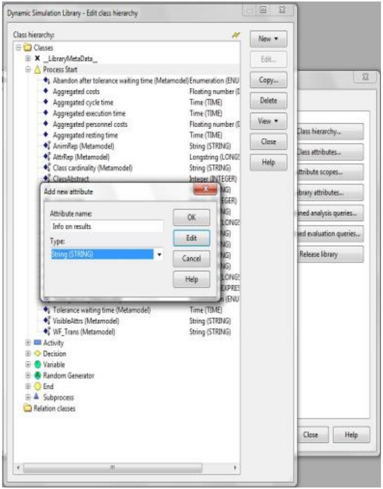
```
MODELTYPE "Working environment model" from:none plural:"Working environment
models"
INCL "Organizational unit"
INCL "Performer"
INCL "Role"

INCL "Belongs to"
INCL "Is manager"
INCL "Has role"
INCL "has Cross-reference"
```

- Open static-library attributes
- Select Add-ons
- Go to Modi and define:

MODELTYPE ”Working environment model”
from:none plural:”Working environment models”
INCL ”Organizational unit”
INCL ”Performer”
INCL ”Role”

INCL ”Belongs to”
INCL ”Is manager”
INCL ”Has role”
INCL ”has Cross-reference”

”Process Start”-Class



Dynamic Simulation Library - Edit class hierarchy

Class hierarchy:

Classes	
LibraryMetaData	
Process Start	
Abandon after tolerance waiting time (Metamodel)	Enumeration (ENU
Aggregated costs	Floating number (I
Aggregated cycle time	Time (TIME)
Aggregated execution time	Time (TIME)
Aggregated personnel costs	Floating number (I
Aggregated resting time	Time (TIME)
AnimRep (Metamodel)	String (STRING)
AttrRep (Metamodel)	Longstring (LONGS
Class cardinality (Metamodel)	String (STRING)
ClassAbstract	Integer (INTEGER)
Tolerance waiting time (Metamodel)	Time (TIME)
VisibleAttrs (Metamodel)	String (STRING)
WF_Trans (Metamodel)	String (STRING)
Activity	
Decision	
Variable	
Random Generator	
End	
Subprocess	
Relation classes	

Add new attribute

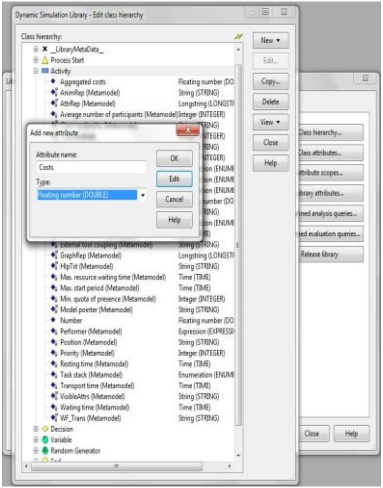
Attribute name: Info on results

Type: String (STRING)

- ○ Open the ”Class hierarchy” for the dynamic library.
- ○ Select ”Process Start”
- ○ Create Attribute:
  - ➢ "Info on results" of type String
- ○ Define Notebook:

```
NOTEBOOK
CHAPTER ”Description”
ATTR ”Name”
CHAPTER ”Simulation data”
GROUP ”Capacity analysis”
ATTR ”Quantity”
ATTR ”Time period” ctrltype:radio
ENDGROUP
GROUP ”Workload analysis”
ATTR ”Process calendar” dialog:processstart-calendar
ATTR ”Tolerance waiting time”
ATTR ”Abandon after tolerance waiting time”  ctrltype:check
checked-value:”yes” unchecked-value:”no”
ENDGROUP
```

"Activity"-Class


Dynamic Simulation Library - Edit class hierarchy

Class hierarchy:

LibraryMetaData	
Process Start	
Activity	
Aggregated costs	Floating number (DO
AnimRep (Metamodel)	String (STRING)
AttrRep (Metamodel)	Longstring (LONGSTI
Average number of participants (Metamodel)	Integer (INTEGER)
External tool coupling (Metamodel)	String (STRING)
GraphRep (Metamodel)	Longstring (LONGSTI
HlpTxt (Metamodel)	String (STRING)
Max. resource waiting time (Metamodel)	Time (TIME)
Max. start period (Metamodel)	Time (TIME)
Min. quota of presence (Metamodel)	Integer (INTEGER)
Model pointer (Metamodel)	String (STRING)
Number	Floating number (DO
Performer (Metamodel)	Expression (EXPRESSI
Position (Metamodel)	String (STRING)
Priority (Metamodel)	Integer (INTEGER)
Resting time (Metamodel)	Time (TIME)
Task stack (Metamodel)	Enumeration (ENUME
Transport time (Metamodel)	Time (TIME)
VisibleAttrs (Metamodel)	String (STRING)
Waiting time (Metamodel)	Time (TIME)
WF_Trans (Metamodel)	String (STRING)
Decision	
Variable	
Random Generator	

Add new attribute

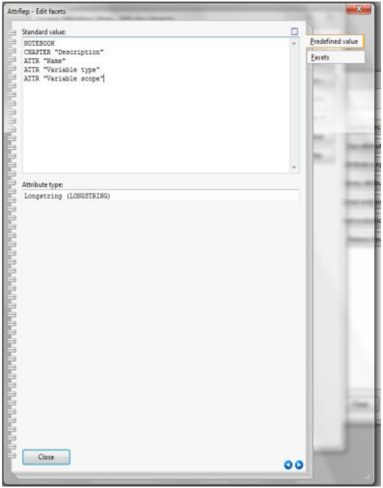
Attribute name: Costs

Type: Floating number (DOUBLE)

- Open the "Class hierarchy" for the dynamic library.
- Select "Activity"
- Create Attribute:
  - "Costs" of type Floating Number
  - "Number" of type Floating Number
- Define Notebook:

NOTEBOOK
CHAPTER "Description"
ATTR "Name"
CHAPTER "Times/Costs"
GROUP "Activity times"
ATTR "Execution time"
ATTR "Waiting time"
ATTR "Resting time"
ATTR "Transport time"
GROUP "Acitvity costs"
ATTR "Costs"
CHAPTER "Working environment"
ATTR "Performer" dialog:actor lines:3
ATTR "Task stack"
ATTR "Done by"

”Variable”-Class



Standard value:

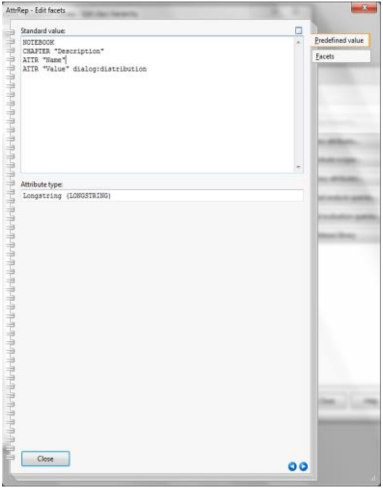
```
NOTEBOOK
CHAPTER "Description"
ATTR "Name"
ATTR "Variable type"
ATTR "Variable scope"
```

Attribute type: Longstring (LONGSTRING)

- ○ Open the ”Class hierarchy” for the dynamic library.
- ○ Select ”Variable”
- ○ Define Notebook:

NOTEBOOK
CHAPTER ”Description”
ATTR ”Name”
ATTR ”Variable type”
ATTR ”Variable scope”

”Random Generator”-Class



Standard value:

```
NOTEBOOK
CHAPTER "Description"
ATTR "Name"
ATTR "Value" dialog:distribution
```

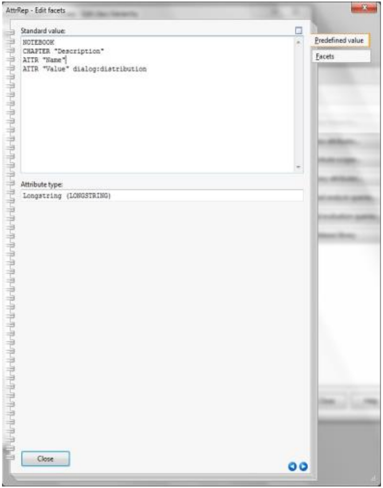
Attribute type:

```
Longstring (LONGSTRING)
```

- Open the ”Class hierarchy” for the dynamic library.
- Select ”Random Generator”
- Define Notebook:

NOTEBOOK
CHAPTER ”Description”
ATTR ”Name”
ATTR ”Value” dialog:distribution

”Performer”-Class



Standard value:

```
NOTEBOOK
CHAPTER "Description"
ATTR "Name"
ATTR "Value" dialog:distribution
```

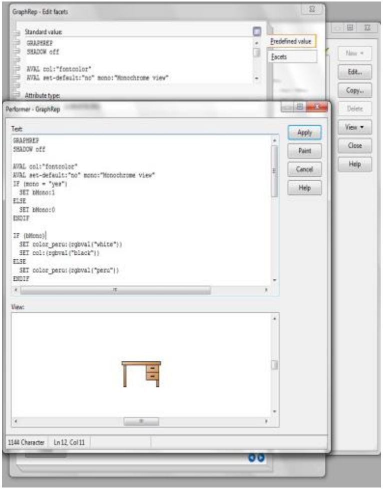
Attribute type:

```
Longstring (LONGSTRING)
```

- Open the ”Class hierarchy” for the static library.
- Select ”Performer”
- Create Attributes of type DOUBLE:
  - ”Capacity”
  - ”Info on results”
  - ”Personnel costs”
  - ”Workload”
- Define Notebook:

NOTEBOOK
CHAPTER ”Description”
ATTR ”Name”
ATTR ”Hourly wages”
ATTR ”Personnel costs”
ATTR ”Availability”
ATTR ”Calendar” dialog:person-calendar
CHAPTER ”Simulation results”
ATTR ”Personnel costs” write-protected
ATTR ”Capacity” write-protected
ATTR ”Workload” write-protected
ATTR ”Info on results” write-protected lines:5

”Performer”-Class GraphRep



```
GRAPHREP
SHADOW off

AVAL col:"fontcolor"
AVAL set-default:"no" mono:"Monochrome view"
```

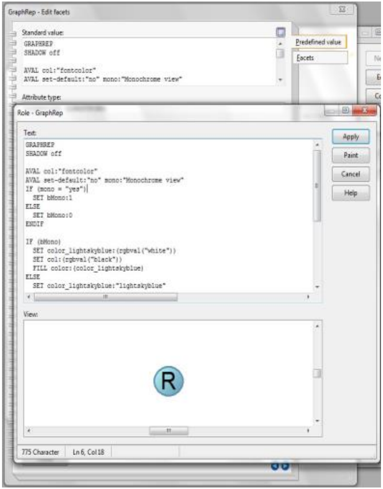
```
GRAPHREP
SHADOW off

AVAL col:"fontcolor"
AVAL set-default:"no" mono:"Monochrome view"
IF (mono = "yes")
  SET bMono:1
ELSE
  SET bMono:0
ENDIF

IF (bMono)
  SET color_peru:(rgbval("white"))
  SET col:(rgbval("black"))
ELSE
  SET color_peru:(rgbval("peru"))
ENDIF
```

- Open the ”Class hierarchy” for the static library.
- Select ”Performer”
- Define GraphRep[2]:

```
GRAPHREP
SHADOW off
AVAL col:”fontcolor”
AVAL set-default:”no” mono:”Monochrome view”
IF (mono = ”yes”)
SET bMono:1
ELSE
SET bMono:0
ENDIF
IF (bMono)
SET color peru:(rgbval(”white”))
SET col:(rgbval(”black”))
ELSE
SET color peru:(rgbval(”peru”))
ENDIF
...
```

[2] You can download the GraphRep-code from the adoxx.org GrapRep repository: http://www.adoxx.org/live/adoxx-graphrep-repository-wiki/-/wiki/GRAPHREP+Repository/FrontPage

”Role”-Class GraphRep



```
GRAPHREP
SHADOW off

AVAL col:"fontcolor"
AVAL set-default:"no" mono:"Monochrome view"
```

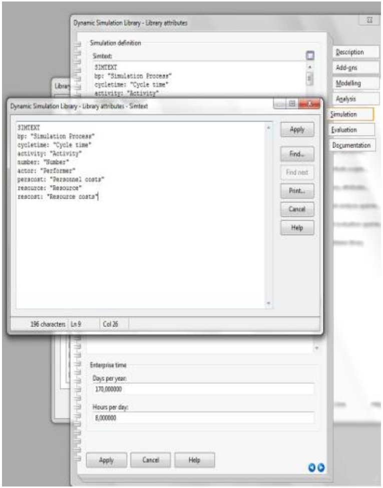
```
GRAPHREP
SHADOW off

AVAL col:"fontcolor"
AVAL set-default:"no" mono:"Monochrome view"
IF (mono = "yes")
  SET bMono:1
ELSE
  SET bMono:0
ENDIF

IF (bMono)
  SET color_lightskyblue:(rgbval("white"))
  SET col:(rgbval("black"))
  FILL color:(color_lightskyblue)
ELSE
  SET color_lightskyblue:"lightskyblue"
```

- Open the ”Class hierarchy” for the static library.
- Select ”Role”
- Define GraphRep[3]:

GRAPHREP
SHADOW off
AVAL col:”fontcolor”
AVAL set-default:”no” mono:”Monochrome view”
IF (mono = ”yes”)
SET bMono:1
ELSE
SET bMono:0
ENDIF
IF (bMono)
SET color lightskyblue:(rgbval(”white”))
SET col:(rgbval(”black”))
FILL color:(color lightskyblue)
ELSE
SET color lightskyblue:”lightskyblue”
SHADOW off
CLIP_ELLIPSE rx:.78cm ry:.68cm
GRADIENT_RECT x:-.75cm y:-0.75cm w:1.5cm h:1.5cm
style:downdiag color1:(rgbval (color lightskyblue, 1.4))
color2:(rgbval (color lightskyblue, 0.7))
...

[3] See: RoleGraphRep.leo

Dynamic Library Attribute ”Simtext”



Dynamic Simulation Library - Library attributes

Simulation definition

Simtext:

```
SIMTEXT
bp: "Simulation Process"
cycletime: "Cycle time"
activity: "Activity"
```

Dynamic Simulation Library - Library attributes - Simtext

```
SIMTEXT
bp: "Simulation Process"
cycletime: "Cycle time"
activity: "Activity"
number: "Number"
actor: "Performer"
perscost: "Personnel costs"
resource: "Resource"
rescost: "Resource costs"
```

196 characters | Ln 9 | Col 26

Enterprise time

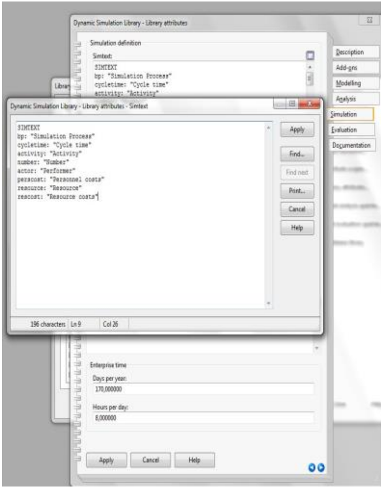
Days per year: 170,000000

Hours per day: 8,000000

- ○ Open dynamic-library attributes
- ○ Select Simulation
- ○ Go to Simtext and define:

SIMTEXT
bp: ”Simulation Process”
cycletime: ”Cycle time”
activity: ”Activity”
number: ”Number”
actor: ”Performer”
perscost: ”Personnel costs”
resource: ”Resource”
rescost: ”Resource costs”

Dynamic Library Attribute ”Simmapping”



Dynamic Simulation Library - Library attributes

Simulation definition

Simtext:

```
SIMTEXT
bp: "Simulation Process"
cycletime: "Cycle time"
activity: "Activity"
```

Dynamic Simulation Library - Library attributes - Simtext

```
SIMTEXT
bp: "Simulation Process"
cycletime: "Cycle time"
activity: "Activity"
number: "Number"
actor: "Performer"
perscost: "Personnel costs"
resource: "Resource"
rescost: "Resource costs"
```

196 characters | Ln 9 | Col 26

Enterprise time

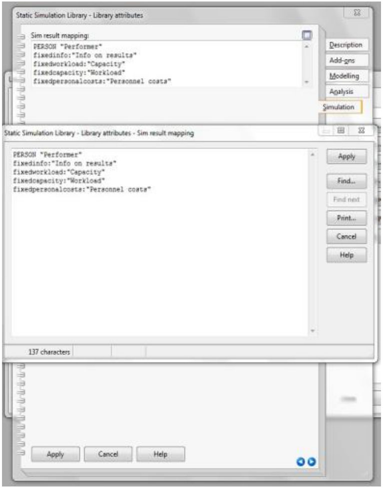
Days per year: 170,000000

Hours per day: 8,000000

- Open dynamic-library attributes
- Select Simulation
- Go to Simmaping and define:

SIMOPTION
name: ”Default”
activity: ”Activity”
executiontime: ”Execution time”
waitingtime: ”Waiting time”
restingtime: ”Resting time”
transporttime: ”Transport time”
userattribute-1: ”Costs”
SIMCLASSES
bp-all
we-1: ”Performer”
we-2: ”Organizational unit”
we-3: ”Role”
processcall: ”Subprocess” subperformerattr:”Performer”

Static Library Attribute ”Sim result mapping”


Static Simulation Library - Library attributes
Description
Add-ons
Modelling
Analysis
Simulation
Static Simulation Library - Library attributes - Sim result mapping
Apply
Find...
Find next
Print...
Cancel
Help
137 characters
Apply
Cancel
Help

Sim result mapping:

```
PERSON "Performer"
fixedinfo:"Info on results"
fixedworkload:"Capacity"
fixedcapacity:"Workload"
fixedpersonalcosts:"Personnel costs"
```

```
PERSON "Performer"
fixedinfo:"Info on results"
fixedworkload:"Capacity"
fixedcapacity:"Workload"
fixedpersonalcosts:"Personnel costs"
```

- Open static-library attributes
- Select Simulation
- Go to ”Sim result mapping” and define:

PERSON ”Performer”
fixedinfo:”Info on results”
fixedworkload:”Capacity”
fixedcapacity:”Workload”
fixedpersonalcosts:”Personnel costs”

We thank you for your attention!

In case of any questions, please contact

tutorial@adoxx.org