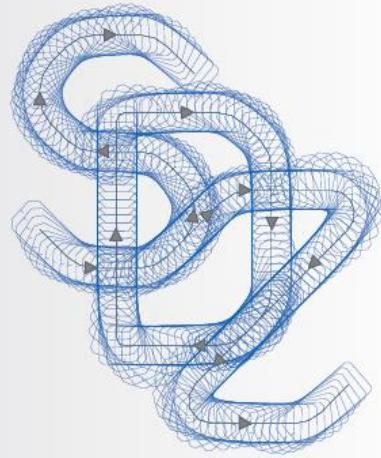


DOSIMIS-3



SDZ GmbH

User Manual

Version 6.2
April 2013

SimulationsDienstleistungsZentrum GmbH
Hauert 20
44227 Dortmund
Tel.: (02 31) 97 50 50 - 0
Fax: (02 31) 97 50 50 - 50



This manual is part of the program
DOSIMIS-3 for MS-Windows
The names DOSIMIS-3 and MS-Windows are reserved.

© 1994-2013 SDZ GmbH
SimulationsDienstleistungsZentrum GmbH
Hauert 20
44227 Dortmund
Tel.: (02 31) 97 50 50 - 0
Fax: (02 31) 97 50 50 - 50
URL: www.sdz.de,
e-mail: Dosimis-3.support@sdz.de

All rights reserved. No part of this script shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical or otherwise, without written permission from SDZ GmbH.



0 Table of Contents

0	Table of Contents	III
1	Introduction.....	1-1
2	Modeling of a Material Flow System.....	2-1
2.1	<i>Introduction to Modeling</i>	2-1
2.1.1	Base Windows.....	2-1
2.1.2	Main Menu File.....	2-2
2.1.3	Main Menu View	2-3
2.1.4	Main Menu Help	2-3
2.1.5	Work Area	2-7
2.1.6	Structure View	2-8
2.1.7	Toolbars.....	2-10
2.1.8	Tooltip	2-11
2.1.9	Context Menu.....	2-12
2.1.10	Positioning of Modules	2-13
2.1.11	Linking of Modules.....	2-16
2.1.12	Linking of Controls with Elements	2-17
2.1.13	Properties of Junctions	2-18
2.1.14	Deleting of Junctions.....	2-18
2.1.15	Object Transfer.....	2-20
2.2	<i>Menu File</i>	2-21
2.2.1	Properties User Interface	2-25
2.2.2	Properties Masking.....	2-27
2.2.3	Properties Simulator.....	2-28
2.2.4	Properties Results.....	2-30
2.2.5	Properties Clipboard.....	2-31
2.2.6	Properties Decision Table	2-32
2.2.7	Properties Statistic.....	2-34
2.3	<i>Menu Edit.....</i>	2-35
2.3.1	Undo.....	2-36
2.3.2	Redo	2-37
2.3.3	Cut (into Clipboard)	2-37
2.3.4	Copy (into Clipboard)	2-37
2.3.5	Paste (from Clipboard)	2-38
2.3.6	Copy (Single)	2-38
2.3.7	Copy (Group)	2-39
2.3.8	Delete	2-39
2.3.9	Parameter.....	2-40
2.3.10	Execute	2-40
2.3.11	Select all	2-40
2.3.12	Previous Selection	2-40
2.3.13	Next Selection	2-41
2.3.14	Move	2-41
2.3.15	Mirror	2-41
2.3.16	Rotate	2-41
2.3.17	Rotate Clockwise.....	2-42
2.3.18	Snap Parameters	2-42
2.3.19	Snap (Individual).....	2-42
2.3.20	Snap (Relative).....	2-42
2.3.21	Hide	2-43
2.3.22	Unhide	2-43
2.3.23	Freeze	2-43



2.3.24	Thaw.....	2-43
2.3.25	Assign Layer	2-43
2.3.26	Characteristics of selected or unselected Elements	2-44
2.4	Menu View.....	2-45
2.4.1	Menu Zoom.....	2-46
2.5	Menu Model	2-50
2.5.1	Search.....	2-51
2.5.2	Submenu Layout	2-53
2.5.3	Submenu Info	2-53
2.5.4	Submenu View	2-56
2.5.5	Submenu Select.....	2-57
2.5.6	Submenu OLE Menu.....	2-58
2.6	Menu Simulation	2-59
2.7	Menu Results	2-60
2.8	Menu Animation.....	2-61
2.9	Menu Programming	2-61
2.10	Menu Graphics.....	2-61
2.11	Menu Window.....	2-62
2.12	Menu Help.....	2-63
3	General Parameters of a Material Flow System.....	3-1
3.1	Strategy Dialog	3-2
3.2	Right-of-way Strategies.....	3-4
3.2.1	Right-of-way Strategy FIFO	3-5
3.2.2	Right-of-way Strategy Priority of Entrances	3-6
3.2.3	Right-of-way Strategy Priority of Object Types	3-6
3.2.4	Right-of-way Strategy Maximum relative Occupancy	3-7
3.2.5	Right-of-way Strategy Maximum absolute Occupancy	3-7
3.2.6	Right-of-way Strategy Minimum free Capacity	3-8
3.2.7	Right-of-way Strategy Percentage.....	3-8
3.2.8	Right-of-way Strategy Bauschuld	3-9
3.2.9	Right-of-way Strategy Self-defined	3-9
3.2.10	Right-of-way Strategy Shortest Path	3-10
3.3	Distribution Strategies	3-11
3.3.1	Distribution Strategy Minimum relative Occupancy.....	3-12
3.3.2	Distribution Strategy Minimum absolute Occupancy	3-12
3.3.3	Distribution Strategy Maximum free Capacity	3-13
3.3.4	Distribution Strategy Alternating Distribution	3-13
3.3.5	Distribution Strategy Priority of Exits.....	3-14
3.3.6	Distribution Strategy Percentage Distribution.....	3-14
3.3.7	Distribution Strategy Bauschuld	3-15
3.3.8	Distribution Strategy Destination-oriented.....	3-15
3.3.9	Distribution Strategy Self-defined.....	3-17
3.3.10	Distribution Strategy shortest Route	3-17
3.3.11	Distribution Strategy Automatic Path Finding	3-18
3.4	Choosing a Distribution Function.....	3-19
3.4.1	Fixed Distribution	3-19
3.4.2	Uniform Distribution	3-20
3.4.3	Triangular Distribution.....	3-21
3.4.4	Normal Distribution	3-22
3.4.5	Exponential Distribution	3-24
3.4.6	Erlang Distribution.....	3-25
3.4.7	Histogram Distribution.....	3-26
3.5	Additional Parameters of Distribution Functions	3-27



3.5.1	<i>Limitation</i>	3-27
3.5.2	<i>Objects per Hour</i>	3-28
3.6	<i>Varying Distribution Function</i>	3-28
3.6.1	<i>Functionality</i>	3-28
3.7	<i>Forward Control</i>	3-29
3.8	<i>Modules with a Length of 0</i>	3-29
3.9	<i>Module Bunch</i>	3-30
3.10	<i>Defaults and Model Constants</i>	3-32
3.11	<i>Input Fields</i>	3-34
3.11.1	<i>Attributes</i>	3-35
3.11.2	<i>Formulas</i>	3-35
3.11.3	<i>Percentage Values</i>	3-35
3.11.4	<i>Erroneous Input</i>	3-35
3.12	<i>Comments and Notes</i>	3-36
3.13	<i>Test Values</i>	3-37
3.14	<i>Lists</i>	3-38
3.15	<i>Passivation of Elements</i>	3-39
3.16	<i>Efficiency Factor</i>	3-40
3.17	<i>Attributes</i>	3-40
3.18	<i>Costs</i>	3-41
3.19	<i>Initializing</i>	3-42
3.20	<i>Layer Selection</i>	3-43
3.21	<i>Breakpoint</i>	3-44
3.22	<i>Collective Parameterization</i>	3-45
3.23	<i>Restrictions in Modeling</i>	3-45
4	Elements of Material Flow	4-1
4.1	<i>Source</i>	4-1
4.1.1	<i>Functionality</i>	4-2
4.1.2	<i>Time of Generation</i>	4-2
4.1.3	<i>Type of Object</i>	4-3
4.1.4	<i>Objects with multiple Tasks</i>	4-3
4.1.5	<i>Generating from File</i>	4-5
4.1.6	<i>Attributes from File</i>	4-8
4.2	<i>Sink</i>	4-9
4.3	<i>Accumulation Conveyor</i>	4-10
4.4	<i>Accumulation Conveyor2</i>	4-12
4.4.1	<i>Sensors</i>	4-13
4.4.2	<i>Object Length</i>	4-14
4.5	<i>Conveying Section</i>	4-15
4.6	<i>Bulk Section</i>	4-16
4.7	<i>LIFO Buffer</i>	4-18
4.8	<i>Storage</i>	4-19
4.8.1	<i>Standard Area Data</i>	4-20
4.8.2	<i>Detailing</i>	4-21
4.8.3	<i>Initializing</i>	4-21
4.8.4	<i>Storage Time</i>	4-22



4.8.5	Delivery Strategy.....	4-23
4.8.6	Bin Location	4-24
4.8.7	Stacker Crane	4-25
4.8.8	Storage Cost	4-26
4.8.9	Storage Status Report	4-27
4.9	<i>Workstation</i>	4-28
4.9.1	Work Procedure.....	4-29
4.9.2	Work Times.....	4-30
4.9.3	Change of Object Type	4-30
4.9.4	Set-up Times	4-30
4.9.5	Configuration	4-31
4.10	<i>Multiple Workstation</i>	4-32
4.10.1	Functionality	4-33
4.11	<i>Assembly Station</i>	4-33
4.11.1	Waiting for Assembly	4-34
4.11.2	Alternative Assembly Lists	4-35
4.11.3	Kind of Assembly.....	4-35
4.11.4	Functionality	4-36
4.12	<i>Disassembly Station</i>	4-37
4.12.1	Functionality	4-39
4.12.2	Disassembly Cost	4-39
4.13	<i>Multifunctional Workstation</i>	4-40
4.13.1	Work Procedures.....	4-40
4.13.2	Cleaning	4-41
4.14	<i>Distributor</i>	4-42
4.15	<i>Discharger</i>	4-43
4.16	<i>Swiveling Belt</i>	4-45
4.17	<i>Combining Station</i>	4-46
4.18	<i>Infeed Element</i>	4-47
4.19	<i>Crossing</i>	4-49
4.19.1	Possible Paths.....	4-50
4.19.2	Functionality	4-50
4.19.3	Lift Time	4-51
4.19.4	Functionality of Lift Time	4-51
4.19.5	Circuit Time	4-52
4.19.6	Functionality of Circuit time	4-52
4.20	<i>Turning Table</i>	4-53
4.20.1	Turning Time	4-54
4.20.2	Functionality	4-55
4.21	<i>Pallet Changer</i>	4-56
4.21.1	Functionality	4-57
4.22	<i>Conveying Circuit</i>	4-58
4.22.1	Functionality	4-59
4.23	<i>Shuttle</i>	4-60
4.23.1	Positions	4-63
4.23.2	Functionality	4-63
4.23.3	Multiple Load.....	4-64
4.23.4	Table.....	4-65
4.23.5	Functionality of the Table	4-65
4.24	<i>Paired Shuttle</i>	4-66
4.24.1	Global Strategy.....	4-67
4.24.2	Double Cycles	4-68
4.24.3	Functionality	4-70



5 Transport Systems	5-1
5.1 <i>Introduction.....</i>	5-1
5.2 <i>Modules of a Transport System.....</i>	5-1
5.2.1 Track	5-2
5.2.2 Loading Station	5-4
5.2.3 Unloading Station.....	5-7
5.2.4 Workstation with Transport Systems	5-9
5.2.5 Service Station	5-11
5.2.6 Waiting Position.....	5-11
5.3 <i>The Transport-Strategy Level.....</i>	5-12
5.3.1 Fully Automatic Mode	5-12
5.3.2 Transport Scheduling	5-12
5.3.3 Local Transport Control	5-15
5.3.4 Scheduling.....	5-16
5.3.5 Automatic Routing	5-17
5.3.6 Subsequent Dispatch	5-18
5.3.7 Energy Monitoring	5-18
5.3.8 Vehicle Bypass.....	5-20
5.3.9 Forcing of Double Cycles.....	5-20
5.4 <i>Deadlock Control.....</i>	5-20
5.5 <i>Result Graphics.....</i>	5-21
5.5.1 Vehicle Statistic	5-21
5.5.2 Order Statistic.....	5-22
5.5.3 Order Throughput Time	5-23
5.5.4 Order Waiting Queue	5-24
5.6 <i>Result Table.....</i>	5-24
5.6.1 Order Statistic.....	5-24
5.6.2 Vehicle Statistic	5-25
5.7 <i>Consistency Check at Transport System.....</i>	5-27
5.8 <i>Restrictions.....</i>	5-28
5.9 <i>Files in the Transport System.....</i>	5-28
6 Petri Nets.....	6-1
6.1 <i>Introduction to Petri Nets Theory</i>	6-1
6.2 <i>Condition/Event Nets.....</i>	6-1
6.3 <i>Positions/Transitions Networks.....</i>	6-2
6.4 <i>Petri Nets in DOSIMIS-3.....</i>	6-2
6.5 <i>Petri Net State</i>	6-4
6.6 <i>Petri Net Event</i>	6-5
6.7 <i>Examples</i>	6-6
6.7.1 Example 1.....	6-6
6.7.2 Example 2.....	6-7
7 Work Area	7-1
7.1 <i>Introduction.....</i>	7-1
7.1.1 Worker Request.....	7-1
7.1.2 Assignment of Workers.....	7-1
7.1.3 Duration of Tasks.....	7-2
7.1.4 Creating a Work Area	7-2
7.1.5 Creating a Working Place.....	7-2
7.1.6 Parameters of a Working Place	7-3



7.2 Area Parameters	7-4
7.2.1 List of Workers	7-4
7.2.2 Scheduling Rules.....	7-5
7.2.3 Task List.....	7-6
7.3 Work Breaks.....	7-12
7.3.1 Show Work Breaks.....	7-13
7.4 Groups of Workers	7-14
7.5 Route List	7-14
7.6 State Cost	7-16
7.7 Result Graphic	7-17
7.7.1 Work Area Statistic	7-17
7.7.2 Worker Employment Diagram	7-18
7.8 Result Table.....	7-18
8 Controls.....	8-1
8.1 Failure / Maintenance / Break	8-1
8.1.1 Defined Failure.....	8-4
8.1.2 Random Failure	8-5
8.1.3 Periodical Failure	8-6
8.1.4 Dependent on Throughput.....	8-7
8.1.5 Reference Failure	8-8
8.1.6 Overlay of Failures and Breaks.....	8-9
8.2 Capacity Monitoring	8-11
8.2.1 Travel Control	8-13
8.2.2 Result Graphic	8-14
8.2.3 Result Table	8-15
8.2.4 Statistic.....	8-15
8.3 Connector.....	8-17
8.4 Shuttle Control	8-18
8.5 Storage Control.....	8-19
8.6 Throughput Time Measurement	8-20
8.6.1 Statistics	8-21
8.6.2 Result Graphic	8-22
8.6.3 Result Table	8-23
8.7 Measuring Element	8-24
8.7.1 Measuring Classes.....	8-26
8.7.2 Result Graphic	8-27
8.7.3 Result Table	8-30
8.7.4 Statistic.....	8-30
8.8 Monitoring	8-31
8.9 Evaluation	8-31
9 Text Editing / Drawing Polygons	9-1
10 Simulation Run.....	10-1
10.1 Simulation Parameters	10-3
10.2 Random Numbers	10-6
10.3 Start of Simulation.....	10-7
10.4 Online Simulation.....	10-9



<i>10.5 Optimization</i>	10-12
10.5.1 Parameters	10-14
10.5.2 Objective Function	10-14
10.5.3 Target Value	10-14
10.5.4 Optimization Run	10-15
10.5.5 Algorithm	10-16
10.5.6 Result Graphic	10-17
10.5.7 Result Table	10-17
11 Simulation Results / Output	11-1
11.1.1 Comparing Results Graphics.....	11-2
11.2 Result Parameters	11-2
11.2.1 Selection.....	11-2
11.2.2 Series.....	11-3
11.2.3 X-Axis	11-5
11.2.4 Y-Axis	11-6
11.2.5 Layout	11-7
11.2.6 Throughput.....	11-8
11.2.7 Percentage	11-9
11.2.8 Cycle Time	11-9
11.2.9 Shop Floor Control.....	11-10
<i>11.3 Result Color</i>	<i>11-10</i>
11.4 Statistics File	11-11
11.4.1 Total/Standard Statistic	11-11
11.4.2 Utilization Statistics	11-13
11.4.3 Additional Statistic Dependent on Modules.....	11-14
11.4.4 Throughput Statistics.....	11-17
11.4.5 Throughput Time Statistics	11-17
11.4.6 Cost Statistics	11-18
11.4.7 Distribution of Random Time	11-19
11.4.8 Further Statistics.....	11-20
<i>11.5 Results Output in the Layout</i>	<i>11-20</i>
11.5.1 Representation of Percentages.....	11-21
11.5.2 Representation of absolute Value	11-22
<i>11.6 Standard Results Diagram</i>	<i>11-24</i>
11.7 Cost	11-26
11.7.1 Capital Commitment	11-26
11.7.2 Object Cost	11-27
11.7.3 Module Cost	11-28
11.7.4 Worker Costs.....	11-29
11.7.5 Cost Measurement	11-30
11.7.6 Cost Measurement (single).....	11-31
<i>11.8 Continuous Diagrams</i>	<i>11-31</i>
11.8.1 Buffer Analysis	11-32
11.8.2 Cycle Analysis	11-38
11.8.3 State Diagram.....	11-42
11.8.4 Production Diagram	11-43
<i>11.9 Discrete-Time Histograms</i>	<i>11-43</i>
11.9.1 Histograms of special Module Types	11-44
11.9.2 Throughput Histogram	11-47
11.9.3 Throughput Time Histograms	11-48
<i>11.10 Further Results Graphics</i>	<i>11-49</i>
11.11 Control of Diagrams	11-49
11.11.1 Scrolling and Zooming.....	11-49
11.11.2 Filter Data.....	11-50
11.11.3 Caption	11-50



11.11.4 Call of Dialog Result Parameter.....	11-50
12 Animation	12-1
<i>12.1 Animation Parameters.....</i>	<i>12-2</i>
12.1.1 Standard Parameters	12-2
12.1.2 Representation	12-3
12.1.3 Kind of Animation	12-4
<i>12.2 Animation Menu.....</i>	<i>12-5</i>
<i>12.3 Animation States.....</i>	<i>12-6</i>
12.3.1 Modules and Objects.....	12-6
12.3.2 Disturbed Modules	12-7
12.3.3 Movement	12-7
12.3.4 Conveying Circuit	12-7
12.3.5 Storage	12-7
12.3.6 Junctions.....	12-9
12.3.7 Failures.....	12-10
12.3.8 Work Areas	12-10
<i>12.4 Animations Toolbar.....</i>	<i>12-10</i>
12.4.1 Control	12-10
12.4.2 Forward	12-11
12.4.3 Kind of Animation	12-11
12.4.4 Hide	12-11
12.4.5 Parameters	12-11
12.4.6 Log Window.....	12-11
12.4.7 Backward.....	12-12
12.4.8 Save State	12-12
<i>12.5 Breakpoint.....</i>	<i>12-13</i>
<i>12.6 View Objects.....</i>	<i>12-14</i>
<i>12.7 Bitmap Animation.....</i>	<i>12-15</i>
12.7.1 Introduction	12-15
12.7.2 Create Bitmap.....	12-15
12.7.3 Adjusting Bitmaps for DOSIMIS-3	12-16
12.7.4 Bitmaps for different Objects	12-17
12.7.5 Animation Run	12-18
12.7.6 Animation Figures Export	12-19
<i>12.8 Continuous Animation.....</i>	<i>12-19</i>
12.8.1 Introduction	12-19
12.8.2 Parameters	12-20
13 Files.....	13-1
<i>13.1 File Types.....</i>	<i>13-1</i>
<i>13.2 Task Protocol</i>	<i>13-2</i>
<i>13.3 The MFS file.....</i>	<i>13-4</i>
14 Introduction to the Decision Tables.....	14-1
15 Connection of the Decision Tables	15-1
<i>15.1 User-defined Strategies.....</i>	<i>15-2</i>
15.1.1 User-defined Right-of-way / Distribution Strategy	15-2
15.1.2 Self defined Working Time	15-4
15.1.3 Shuttle Decision table.....	15-5
15.1.4 Conveying Circuit Decision Table	15-6
15.1.5 Transport Strategy	15-6



15.1.6	Work Area Strategies	15-7
15.2	<i>Definition of a Decision Table</i>	15-8
16	The Parameter Mask of Decision Tables	16-1
16.1	<i>Initializations / Conditions / Actions</i>	16-2
16.1.1	Step 1: Initialization	16-2
16.1.2	Step 2: Condition.....	16-3
16.1.3	Step 3: Action.....	16-3
16.1.4	Step 4: Rules	16-3
16.2	<i>Input</i>	16-4
16.2.1	Selection of a Line.....	16-4
16.2.2	Adding Lines	16-4
16.2.3	Editing Lines	16-4
16.2.4	Finish Input of a Line	16-5
16.2.5	Syntax Editor.....	16-5
16.2.6	Context Sensitive Selection.....	16-7
16.2.7	Deleting the Input Line	16-8
16.2.8	Deactivating single action	16-8
16.3	<i>Rule Window</i>	16-8
16.3.1	Creating Rules	16-9
16.3.2	Change Rules	16-9
16.3.3	Delete Rules	16-9
16.3.4	Copy Rules	16-9
16.3.5	Set Breakpoint	16-10
16.4	<i>Highlighting by Colors</i>	16-11
16.5	<i>The Watch Window</i>	16-11
16.6	<i>Structure View</i>	16-12
16.6.1	Type of Tables.....	16-12
16.6.2	Creating Decision Tables	16-13
16.6.3	Delete Decision Tables.....	16-14
16.6.4	Rename Decision Table	16-14
16.6.5	Copy Decision Table.....	16-14
16.6.6	Move Decision Table	16-14
16.6.7	Copy/Move per Drag & Drop	16-15
16.7	<i>Variable view</i>	16-15
16.7.1	Types of Variables	16-15
16.8	<i>References</i>	16-17
16.8.1	Definition of References	16-18
16.8.2	Deletion of References	16-19
16.8.3	Exchange of References	16-19
16.8.4	Definition of Activating Modules	16-19
16.8.5	Definition of Activating Timer.....	16-19
16.8.6	Converting of Reference Modules to Activating Modules	16-20
16.8.7	Usage of References.....	16-20
16.9	<i>Call Stack</i>	16-20
16.10	<i>Macros</i>	16-21
16.11	<i>Terminating Input</i>	16-21
17	Further Components.....	17-1
17.1	<i>Animation text</i>	17-1
17.2	<i>Quicktable</i>	17-2
17.2.1	Display	17-4
17.2.2	Data Format.....	17-4



<i>17.3 Monitoring</i>	<i>17-4</i>
<i>17.4 Timer</i>	<i>17-5</i>
<i>17.5 Signal Display</i>	<i>17-6</i>
<i>17.6 Progress indicator</i>	<i>17-7</i>
18 Language Reference.....	18-1
<i>18.1 Types</i>	<i>18-2</i>
<i>18.2 Operator.....</i>	<i>18-3</i>
<i>18.3 String</i>	<i>18-4</i>
<i>18.4 Basic Elements</i>	<i>18-5</i>
18.4.1 Selectors	18-5
18.4.2 Variables	18-6
18.4.3 Random Function	18-8
18.4.4 Mathematical Function	18-9
18.4.5 Decision Tables	18-10
18.4.6 Text Output	18-11
18.4.7 Transport System (TS)	18-12
18.4.8 Miscellaneous	18-13
<i>18.5 Module attribute</i>	<i>18-16</i>
18.5.1 Values (Read Only)	18-16
18.5.2 Values (Write Only)	18-17
18.5.3 Values (Read/Write)	18-17
18.5.4 Event and State	18-18
18.5.5 Objects	18-19
18.5.6 Strategies	18-20
18.5.7 Transport Systems (TS)	18-21
18.5.8 Storage	18-21
18.5.9 Model	18-22
18.5.10 Operations	18-22
<i>18.6 Object attribute</i>	<i>18-23</i>
18.6.1 Values (Read Only)	18-23
18.6.2 Values (Read/Write)	18-24
18.6.3 Transport Vehicles(TV)	18-25
<i>18.7 Junction attribute</i>	<i>18-26</i>
18.7.1 Value (Read Only)	18-26
18.7.2 Operation	18-27
<i>18.8 Quicktable</i>	<i>18-28</i>
18.8.1 Values	18-28
18.8.2 Operation	18-29
<i>18.9 Timer</i>	<i>18-30</i>
18.9.1 Values	18-30
18.9.2 Operations	18-30
<i>18.10 Worker</i>	<i>18-31</i>
18.10.1 Values	18-31
<i>18.11 Transport order</i>	<i>18-31</i>
18.11.1 Values	18-31
<i>18.12 Work</i>	<i>18-32</i>
18.12.1 Values	18-32
<i>18.13 Working Step</i>	<i>18-32</i>
18.13.1 Values	18-32
<i>18.14 Working plan</i>	<i>18-32</i>
18.14.1 Values	18-32



<i>18.15 Order</i>	18-33
18.15.1 Values.....	18-33
<i>18.16 Sensor</i>	18-33
18.16.1 Values.....	18-33
<i>18.17 Transport system</i>	18-33
18.17.1 Operations	18-33
<i>18.18 File</i>	18-34
18.18.1 Operations	18-34
<i>18.19 Animation text</i>	18-34
18.19.1 Operations	18-34
19 Error Situations	19-1
20 Log Window	20-1
20.1 <i>Searching in the Log window</i>	20-1
21 Animation of Decision Tables	21-1
22 Online - Simulation	22-1
22.1 <i>Profiling</i>	22-3
23 Examples	23-1
23.1 <i>Example of a Target and GC Decision Tables</i>	23-1
23.1.1 Description of the Example	23-1
23.2 <i>Example of a Distribution-DT</i>	23-4
23.2.1 Description on the Examples.....	23-4
23.3 <i>ATS-Model with global DT-Data</i>	23-5
23.3.1 Solution by using a global DT	23-7
24 Shop Floor Control	24-1
24.1 <i>Introduction (Shop Floor Control)</i>	24-1
24.2 <i>Work plan</i>	24-1
24.3 <i>The Shop Floor Control Editor</i>	24-2
24.3.1 Overview	24-2
24.3.2 Work Plans and Operations	24-3
24.3.3 Production Order	24-7
24.4 <i>Result Graphic</i>	24-8
24.4.1 Diagrams	24-8
24.5 <i>Result Table</i>	24-10
24.5.1 Statistic File.....	24-10
24.6 <i>Old File Formats of Work Plans</i>	24-12
24.7 <i>Example</i>	24-13
24.7.1 Model	24-13
24.7.2 Problem	24-13
24.7.3 Modeling	24-13
24.7.4 Order	24-14
24.7.5 Work Plan.....	24-15
24.7.6 Statistic	24-16
25 Graphic Comments	25-1



25.1 The Menu „Graphic“	25-1
25.1.1 Graphic Palette	25-1
25.1.2 Group	25-2
25.1.3 Ungroup	25-2
25.1.4 View Group	25-3
25.1.5 View Symbols	25-3
25.1.6 Delete unused Groups	25-3
25.1.7 Delete unused Symbols	25-3
25.1.8 Foreground	25-3
25.1.9 Background	25-3
25.1.10 One Layer front	25-3
25.1.11 One Layer back	25-3
25.1.12 Reorder	25-3
25.1.13 Export/DXF- Export/DXG-Export	25-3
25.1.14 Import/DXF- Import/DXG-Import	25-4
25.1.15 Select (all Graphic)	25-4
25.1.16 Hide all Graphic	25-5
25.1.17 Freeze	25-5
25.2 Graphic Palette	25-5
25.2.1 Drawing with the Graphic Palette	25-5
25.2.2 Manipulating Graphic Objects with the Graphic Pallet	25-6
25.3 Graphic Elements	25-7
25.3.1 Line	25-7
25.3.2 Rectangle	25-8
25.3.3 Circle	25-9
25.3.4 Polyline	25-10
25.3.5 Polygon	25-10
25.3.6 Reference	25-11
25.3.7 Text	25-12
25.3.8 Bitmap	25-13
25.3.9 The Group Selection Dialog	25-14
25.3.10 The Dialog „Name of Group“	25-14
25.4 The Symbols	25-15
26 3D Visualization	26-1
26.1 Introduction	26-1
26.1.1 Installation	26-1
26.1.2 System Requirement	26-1
26.2 Application	26-1
26.2.1 Multiple View	26-2
26.3 Navigation	26-3
26.3.1 Rotation	26-3
26.3.2 Forward/Backwards	26-3
26.3.3 Sidewise Movement	26-3
26.4 Operations in the 3D-View	26-4
26.5 Operation on the 3D-Objects	26-4
26.5.1 Move	26-5
26.5.2 Scaling	26-5
26.5.3 Rotate	26-5
26.5.4 Parameterization	26-5
26.6 3D Toolbar	26-5
26.6.1 Functions of the 3D Toolbar	26-5
26.7 The Window “object property”	26-6
26.8 Graphics Exchange	26-7
26.8.1 Convert the Representation	26-7



26.9	<i>Duality of the 2D and 3D Display</i>	26-8
26.10	<i>Snap of the 3D-Objects</i>	26-9
26.11	<i>Animation</i>	26-10
26.11.1	Navigation during the Animation.....	26-11
26.12	<i>Animation Parameter</i>	26-11
26.12.1	Object type Assignment	26-11
26.12.2	Animation Speed.....	26-12
26.13	<i>Camera</i>	26-12
26.13.1	Active Camera.....	26-12
26.13.2	Properties of the Camera	26-12
26.13.3	Adding Cameras.....	26-12
26.13.4	Choosing a Camera	26-12
26.13.5	Deleting a Camera	26-12
26.13.6	Initialize a Camera	26-13
26.14	<i>Camera Animation</i>	26-13
26.14.1	Creating a Camera Animation	26-13
26.15	<i>Overview of the Key- and Mouse Action in 3D View</i>	26-14
26.16	<i>Restrictions</i>	26-14
27	Excel-Interface	27-1
27.1	<i>Introduction</i>	27-1
27.2	<i>Working Method</i>	27-1
27.3	<i>Structure of Table</i>	27-1
27.4	<i>Connecting with Excel</i>	27-1
27.5	<i>Transfer of Data</i>	27-2
27.6	<i>Automatic Creating of a Table</i>	27-3
27.7	<i>Closing of the Data Transfer</i>	27-3
27.8	<i>Keywords and Parameters</i>	27-3
27.8.1	Command Set	27-3
27.8.2	Command Get	27-6
27.8.3	Command Start.....	27-7
27.8.4	Command Save	27-8
27.8.5	Command Save_as	27-8
27.8.6	Command Goto	27-8
27.8.7	Command Keyword	27-8
27.8.8	Further Commands	27-9
27.9	<i>Example</i>	27-9
27.10	<i>Note</i>	27-11
28	COM - Server	28-1
28.1	<i>Introduction</i>	28-1
28.2	<i>Data Types</i>	28-1
28.3	<i>Methods</i>	28-2
28.3.1	Overview	28-2
28.3.2	Methods of IDs3Application	28-3
28.3.3	Methods of IDs3Control.....	28-4
28.3.4	Methods of IDs3Document	28-4
28.3.5	Methods of IDs3Element	28-8
28.3.6	Methods of IDs3Evaluation.....	28-9
28.3.7	Methods of IDs3Failure	28-9



28.3.8	Methods of IDs3Junction	28-10
28.3.9	Methods of IDs3Workarea	28-11
28.4	<i>Connecting to Visual Basic for Excel</i>	28-12
28.4.1	Usage.....	28-13
28.4.2	Example: Experiment	28-15
28.4.3	Example: Model Creation	28-16
28.5	<i>Connecting to Visual C++</i>	28-17
28.5.1	Usage.....	28-18
28.5.2	Example: Experiment	28-20
29	Program Interface for Controls	29-1
29.1	<i>Introduction</i>	29-1
29.2	<i>Little C/C++/MFC Glossary</i>	29-1
29.2.1	Keywords	29-1
29.2.2	Common Words and Terms of C/C++	29-2
29.2.3	Common Words and Terms of Visual-C++	29-2
29.3	<i>Menu</i>	29-2
29.4	<i>Configuration</i>	29-4
30	Integrated Editor	30-1
31	Program Interface Common	31-1
31.1	<i>Information of the Interface</i>	31-1
31.2	<i>Debugging User Libraries</i>	31-2
31.2.1	Debug by Using Control Output	31-2
31.2.2	Debugging using the Visual-C++ - Debugger (Visual Studio 2008 / 2010)	31-4
31.2.3	Debugging using the Visual-C++ - Debugger (Visual Studio 2005).....	31-7
31.2.4	Debugging using the Visual Studio Debugger (Visual Studio 6.0)	31-9
31.2.5	Using the Dosimis-3 Data Structures	31-11
31.3	<i>Template Files</i>	31-12
31.3.1	Customizing an own Environment	31-12
31.4	<i>Directory Structure</i>	31-12
31.4.1	Project Directory	31-12
31.4.2	Workspace Directory.....	31-13
31.4.3	Files of the Program Interface	31-13
31.5	<i>Control Structure</i>	31-14
31.5.1	Offline Simulation.....	31-14
31.5.2	Online Simulation	31-14
31.5.3	Multi Threading	31-14
32	Creating a new User Library	32-1
32.1	<i>Description (Visual Studio 2008 / 2010)</i>	32-1
32.1.1	Step 1.....	32-1
32.1.2	Step 2.....	32-2
32.1.3	Step 3	32-3
32.2	<i>Description (Visual Studio 2005)</i>	32-4
32.2.1	Step 1.....	32-4
32.2.2	Step 2.....	32-5
32.2.3	Step 3	32-6
32.3	<i>Description (Visual Studio 6.0)</i>	32-7
32.3.1	Step 1.....	32-7
32.3.2	Step 2.....	32-8



32.3.3 Step 3.....	32-9
32.4 Placing new Controls in the Model Layout.....	32-9
33 Interface for Controls	33-1
33.1 API Functions.....	33-1
33.1.1 User Interface	33-2
33.1.2 Parameter.....	33-2
33.1.3 Simulator	33-3
33.1.4 Functions of the Simulator Library	33-3
33.2 Example.....	33-4
33.2.1 Symbol	33-5
33.2.2 Simulator	33-7
33.2.3 Consistency Check	33-9
33.2.4 Simulator (part 2)	33-10
33.2.5 Using Parameters.....	33-12
33.2.6 Saving and Restoring Control Parameters	33-13
33.2.7 Changing Parameters from the User Interface	33-14
33.3 Away from Theory	33-16
33.3.1 Data Structure.....	33-16
33.3.2 User Interface Functions	33-17
33.3.3 Dialog.....	33-17
33.3.4 Simulator Functions	33-17
33.3.5 Abstract	33-20
34 Interface for Modules	34-1
34.1 Basics	34-1
34.1.1 Events.....	34-1
34.1.2 State Machines	34-1
34.2 Interface Functions	34-3
34.3 Helper for Modules	34-5
34.4 Data Input	34-7
34.5 Templates	34-10
34.5.1 Template Work	34-10
34.5.2 Template Crossing	34-13
35 Program Interface for Decision Tables	35-1
35.1 Introduction.....	35-1
35.2 Declaration	35-1
35.3 Consistency Check	35-2
35.4 List of Parameters	35-2
36 English - German Dictionary	36-1
37 Installation Instructions.....	37-1
37.1 Software Protection.....	37-2
37.1.1 Single License	37-2
37.1.2 Network License	37-2
37.2 Trouble Shooting	37-2
37.2.1 Windows Runtime.....	37-2
37.2.2 Excel 97.....	37-2
37.2.3 DirectX Installation.....	37-2



37.2.4	Windows Help System and Windows 7 / Vista.....	37-3
37.3	<i>Installation Files</i>	37-3
37.3.1	Sample Files	37-3
37.3.2	Module Symbols	37-3
37.3.3	Bitmap Examples	37-4
37.3.4	HTML	37-4
37.3.5	Support	37-4
37.3.6	User Library	37-4
37.3.7	X-Files.....	37-5
38	Update Instructions.....	38-1
38.1	<i>Additional Note for Update Installations</i>	38-1
38.2	<i>Updating the CodeMeter USB Stick</i>	38-1
39	Release Notes 6.2	39-1
39.1	<i>Standard Functions</i>	39-1
39.1.1	User Interface	39-1
39.1.2	Modules.....	39-1
39.1.3	Controls.....	39-1
39.2	<i>Decision Tables</i>	39-1
40	Release Notes 6.1	40-1
40.1	<i>Standard Functions</i>	40-1
40.1.1	User Interface	40-1
40.1.2	Modules.....	40-1
40.1.3	Controls.....	40-1
40.2	<i>Decision Table</i>	40-2
41	Release Notes 6.0	41-1
41.1	<i>Standard functions</i>	41-1
41.1.1	User Interface	41-1
41.1.2	Modules.....	41-1
41.1.3	Controls.....	41-1
41.2	<i>Decision tables</i>	41-1
42	Release Notes 5.1	42-1
42.1	<i>Standard Functions</i>	42-1
42.1.1	User Interface	42-1
42.1.2	Modules.....	42-1
42.1.3	Controls.....	42-1
42.2	<i>Decision tables</i>	42-2
42.3	<i>3D-Animation</i>	42-2
43	Release Notes 5.0	43-1
43.1	<i>Standard functions</i>	43-1
43.1.1	User Interface	43-1
43.1.2	Modules.....	43-1
43.1.3	Controls.....	43-1
43.1.4	Results	43-1
43.1.5	Animation.....	43-1
43.2	<i>Decision Table</i>	43-2



43.3 Programming Interface	43-2
43.3.1 COM-Interface	43-2
43.4 3D-Animation	43-2
43.5 Changes and Corrections	43-2
43.5.1 Excel-Interface	43-2
44 Release Notes 4.3a	44-1
44.1 Standard Functions	44-1
44.1.1 User Interface	44-1
44.1.2 Modules	44-1
44.1.3 Controls	44-1
44.1.4 Results	44-1
44.1.5 Animation	44-1
44.2 Programming Interface	44-2
44.2.1 Modules Programming	44-2
44.3 3D-Visualization	44-2
45 Release Notes 4.3	45-1
45.1 Standard Functions	45-1
45.1.1 User Interface	45-1
45.1.2 Modules	45-1
45.1.3 Controls	45-1
45.1.4 Results	45-1
45.1.5 Animation	45-2
45.2 Decision Tables	45-2
45.2.1 Usage	45-2
45.2.2 New Functions	45-2
45.3 Programming Interface	45-2
45.3.1 Excel-Interface	45-2
45.3.2 COM-Interface	45-3
45.4 3D-Visualization	45-3
45.5 Changes and Corrections	45-3
46 Release Notes 4.2	46-1
46.1 Standard Functions	46-1
46.1.1 User Interface	46-1
46.1.2 Modules	46-1
46.1.3 Controls	46-1
46.1.4 Work Plans	46-1
46.1.5 Animation	46-1
46.2 Decision Tables	46-2
46.2.1 Usage	46-2
46.2.2 New Functions	46-2
46.3 Programming Interface	46-2
46.3.1 Excel - Interface	46-2
46.3.2 COM - Interface	46-2
46.4 Changes and Corrections	46-2
47 Release Notes 4.1	47-1
47.1 Standard Function	47-1
47.1.1 User Interface	47-1



47.1.2	Modules.....	47-1
47.1.3	Controls.....	47-1
47.1.4	Working Plans	47-1
47.1.5	Animation.....	47-1
47.1.6	Miscellaneous.....	47-2
47.2	<i>Decision Tables</i>	47-2
47.2.1	Usage.....	47-2
47.2.2	New Function.....	47-2
47.3	<i>Programming Interface</i>	47-2
47.3.1	Excel-Interface	47-2
47.3.2	COM-Interface	47-2
47.3.3	Programming Interface.....	47-2
48	Release Notes 4.0b/c	48-1
48.1	<i>Changes and Corrections</i>	48-1
49	Release Notes 4.0a	49-1
49.1	<i>Changes and Corrections</i>	49-1
50	Release Notes 4.0	50-1
50.1	<i>Standard Function</i>	50-1
50.1.1	User Interface	50-1
50.1.2	Modules.....	50-1
50.1.3	Controls.....	50-1
50.1.4	Statistics	50-2
50.1.5	Miscellaneous.....	50-3
50.1.6	Cost Simulation	50-3
50.2	<i>Decision Tables</i>	50-4
50.2.1	Usage.....	50-4
50.2.2	Configuration / Parameter	50-4
50.2.3	New Functions	50-4
50.2.4	Searching Errors.....	50-5
50.3	<i>Programming Interface</i>	50-5
50.3.1	COM-Interface	50-5
50.3.2	Programming Interface.....	50-5
51	Release Notes 3.2b	51-1
51.1	<i>Changes and Corrections</i>	51-1
52	Release Notes 3.2a	52-1
52.1	<i>Changes and Corrections</i>	52-1
53	Release Notes 3.2	53-1
53.1	<i>New Functionality</i>	53-1
53.2	<i>Changes and Corrections</i>	53-2
54	Release Notes 3.1a/b/c	54-1
54.1	<i>New Functionality</i>	54-1
54.2	<i>Changes and Corrections</i>	54-1
55	Release Notes 3.1	55-1



55.1	<i>New Functionality</i>	55-1
55.2	<i>Changes and Corrections</i>	55-2
56	Release Notes 3.0a	56-1
56.1	<i>Added Functionality</i>	56-1
56.1.1	International Version.....	56-1
56.1.2	Corrections	56-1
57	Release Notes 3.0	57-1
57.1	<i>Added Functionality</i>	57-1
57.1.1	Interaction	57-1
57.1.2	Representation.....	57-1
57.1.3	Default.....	57-1
57.1.4	Dynamic Element Linking	57-1
57.1.5	Automatic Guided Vehicles	57-1
57.1.6	Export.....	57-1
57.1.7	Failures Dependent on Throughput	57-1
57.1.8	New Parameter with Right of Way and Distribution Strategy	57-1
57.1.9	Validation.....	57-1
57.1.10	Documentation	57-2
57.2	<i>New Functionality</i>	57-2
57.2.1	Decision Tables	57-2
57.2.2	Decision Tables Dialog	57-2
57.2.3	New Disposition of Worker	57-2
57.2.4	Undo.....	57-2
57.2.5	Online - Simulation	57-2
57.2.6	Animation.....	57-2
57.2.7	Programming interface	57-2
57.2.8	Excel interface.....	57-2
57.3	<i>Corrections</i>	57-2
57.3.1	Start Value of Random Numbers	57-2
57.3.2	Statistic Shuttle and Turntable	57-3
57.3.3	Continuation of Generating Events	57-3
58	Release Notes 2.3a	58-1
58.1	<i>Added Functionality</i>	58-1
58.1.1	Interaction	58-1
58.1.2	Representation.....	58-1
58.1.3	Dialogs	58-1
58.1.4	Calculation	58-1
58.2	<i>New Functionality</i>	58-1
58.2.1	Strategy	58-1
58.2.2	Dialogs	58-1
58.2.3	Animation.....	58-1
58.2.4	Results	58-2
58.2.5	Excel interface.....	58-2
58.3	<i>Corrections</i>	58-2
58.3.1	Random Processes of DTs.....	58-2
58.3.2	Several Order in Agv Strategy	58-2
59	Release Notes 2.3	59-1
59.1	<i>New Functionality</i>	59-1
59.1.1	Extension to <u>Failures</u>	59-1
59.2	<i>Easy Handling</i>	59-1



59.2.1	New Selection of Elements	59-1
59.3	<i>Decision Tables</i>	59-1
59.4	<i>Support of Validation</i>	59-1
59.5	<i>Support of the Clipboard</i>	59-1
59.6	<i>Support During Experimentation</i>	59-1
59.7	<i>Graphic</i>	59-2
59.7.1	New Graphic Types.....	59-2
59.7.2	<u>Graphic-Toolbar</u>	59-2
59.7.3	<u>Preview for Groups</u>	59-2
59.8	<u>Bitmap-Animation</u>	59-2
59.9	<u>Element Symbols</u>	59-2
60	Release Notes 2.2	60-1
60.1	<i>New Functionality</i>	60-1
60.1.1	Decision Tables	60-1
60.1.2	Interaction	60-1
60.1.3	New Strategy in <u>Source</u>	60-1
60.1.4	Results	60-1
60.1.5	Work Area	60-1
60.2	<i>Corrections</i>	60-2
60.2.1	Random Numbers.....	60-2
60.2.2	Animation.....	60-2
60.2.3	Work Areas	60-2
60.2.4	Elements	60-2
60.2.5	AGV	60-2
61	Introduction to the Tutorial (Part 1)	61-1
61.1	<i>Structure of the Tutorial</i>	61-1
61.2	<i>Installation of DOSIMIS-3</i>	61-2
62	Task	62-1
62.1	<i>Philosophy of Modeling</i>	62-1
62.2	<i>Exercises</i>	62-1
62.2.1	Model	62-1
62.2.2	Question	62-2
62.3	<i>Database</i>	62-3
62.3.1	Source.....	62-3
62.3.2	Accumulation Conveyor	62-3
62.3.3	Shuttle	62-3
62.3.4	Work Station	62-3
62.3.5	Combining Station.....	62-4
62.3.6	Distributor	62-4
62.3.7	Sink	62-4
63	Modeling of a Production System	63-1
63.1	<i>Entry</i>	63-1
63.1.1	Linking of Modules	63-5
63.2	<i>Input of Data with the Aid of Parameter Masks</i>	63-7
63.2.1	Parameter Source.....	63-7
63.2.2	Parameter Accumulation Conveyor	63-9
63.2.3	Parameter Shuttle	63-10



63.2.4	Parameter Work Station	63-13
63.2.5	Parameter Combining Station	63-16
63.2.6	Parameter Distributor	63-17
63.2.7	Parameter Sink	63-18
63.3	<i>Starting a Simulation Run</i>	63-19
63.4	<i>Results</i>	63-20
63.5	<i>Problems</i>	63-22
64	Experiments using the Practice Example	64-1
64.1	<i>Initial Situation</i>	64-1
64.2	<i>Steps of Optimization</i>	64-2
64.2.1	Step 1 - Deadlock	64-2
64.2.2	Step 2 - Presorting	64-3
64.2.3	Step 3 - Decoupling (Increase of Buffer Size)	64-6
64.2.4	Step 4 - Increase Shuttle Speed	64-8
64.2.5	Step 5 – Preferred Disposal	64-9
64.2.6	Step 6 - Buffer before Sink.....	64-10
64.2.7	Step 7 - Optimization of Set-up Time	64-13
64.2.8	Step 8 - Mitigation of the Sink	64-15
64.2.9	Step 9 - Decrease of Buffer Size before the Sink	64-17
64.2.10	Step 10 - Decrease of Buffer Size before Work Stations	64-17
64.2.11	Step 11 - Factory Tuning.....	64-18
64.3	<i>Results of this Simulation Study</i>	64-19
65	Graphical Comments	65-1
65.1	<i>Insertion of Graphical Elements</i>	65-1
65.2	<i>Change the Size of a Graphical Element</i>	65-1
65.3	<i>Add a Rectangle</i>	65-2
66	Summary: Data of the Study	66-1
66.1	<i>Model Parameter</i>	66-1
66.2	<i>Summary of all Simulation Runs</i>	66-3
67	Introduction to the Tutorial (Part 2)	67-1
67.1	<i>Structure of the Tutorial</i>	67-1
67.2	<i>Icons</i>	67-1
68	Failures and Pauses	68-1
68.1	<i>Task</i>	68-1
68.2	<i>Theory</i>	68-1
68.3	<i>Including of Failures in the Simulation Model</i>	68-2
68.4	<i>Editing of Failure Parameters</i>	68-4
68.5	<i>Analysis of Failures</i>	68-7
68.6	<i>Statistics File</i>	68-10
68.7	<i>Task</i>	68-12
68.8	<i>Shift Model</i>	68-13
68.9	<i>Filter Failures and Pauses out of Statistic Data</i>	68-16



68.10	<i>Disable Failures</i>	68-19
68.10.1	Disable a single Break Module.....	68-19
68.10.2	Disable all Break Modules global	68-20
69	Work Area	69-1
69.1	<i>Theory</i>	69-1
69.2	<i>Task</i>	69-1
69.3	<i>Parameter Setting of Work Area</i>	69-3
69.3.1	Work Stations.....	69-3
69.3.2	Work Area	69-5
69.4	<i>Analysis of Work Area</i>	69-8
69.5	<i>Statistic</i>	69-10
69.6	<i>Several Worker per Task</i>	69-12
69.7	<i>Interrupt of Tasks</i>	69-15
69.8	<i>Passivate Work Areas</i>	69-17
70	Index.....	70-1



1 Introduction

DOSIMIS-3 is a module-oriented graphical interactive standard simulator. The simulator works based on discrete events and allows simulation of discrete-time material flow systems. In marketing DOSIMIS-3, the SimulationsDienstleistungsZentrum GmbH realizes the idea of making a simulation technique available among members of planning departments in companies. A simulated production process can be graphically and interactively developed on the monitor without any particular knowledge in the area of computer science. Standard elements such as sources, sinks, workstations, buffers, vehicles etc., which in their structure represent essential modules from the material flow field, allow a rational layout by means of a menu-controlled user interface. Modules with several entrances and exits are capable of an intelligence with which local strategies such as FIFO, minimal occupancy of the succeeding module etc. can be realized when controlling the object flow. Theoretically the modular concept does not restrict scope and size of the simulation. Super-ordinated levels enable the planner to define failures and breaks or to simulate the deployment of workers in any number of freely definable work sections.

In order to evaluate the simulation results, DOSIMIS-3 offers a variety of tables and graphics. A dynamic presentation of the model behavior is presented within the animation. With its help, the model can be checked, or the occurrence of certain situations, such as a deadlock, can be analyzed. On the other hand, exact statistics can be kept for every module of the model, whose output can be made either in the form of tables or, if desired, in graphical form, such as bar diagrams, occupancy diagrams, or synoptical presentations of the whole system using different colors and state diagrams. All the above-mentioned functions are part of the standard functions of DOSIMIS-3 and are described in the first part of the user manual.

All functions mentioned above belong to the standard functions of DOSIMIS-3 and will be described in this manual.

In the following chapters a guide to DOSIMIS-3 for MS-WINDOWS is given. According the usage of MS-WINDOWS the user should consult those software-manuals.





2 Modeling of a Material Flow System

2.1 Introduction to Modeling

2.1.1 Base Windows

The system is started by double-clicking the DOSIMIS-3 icon whereby the following window appears:

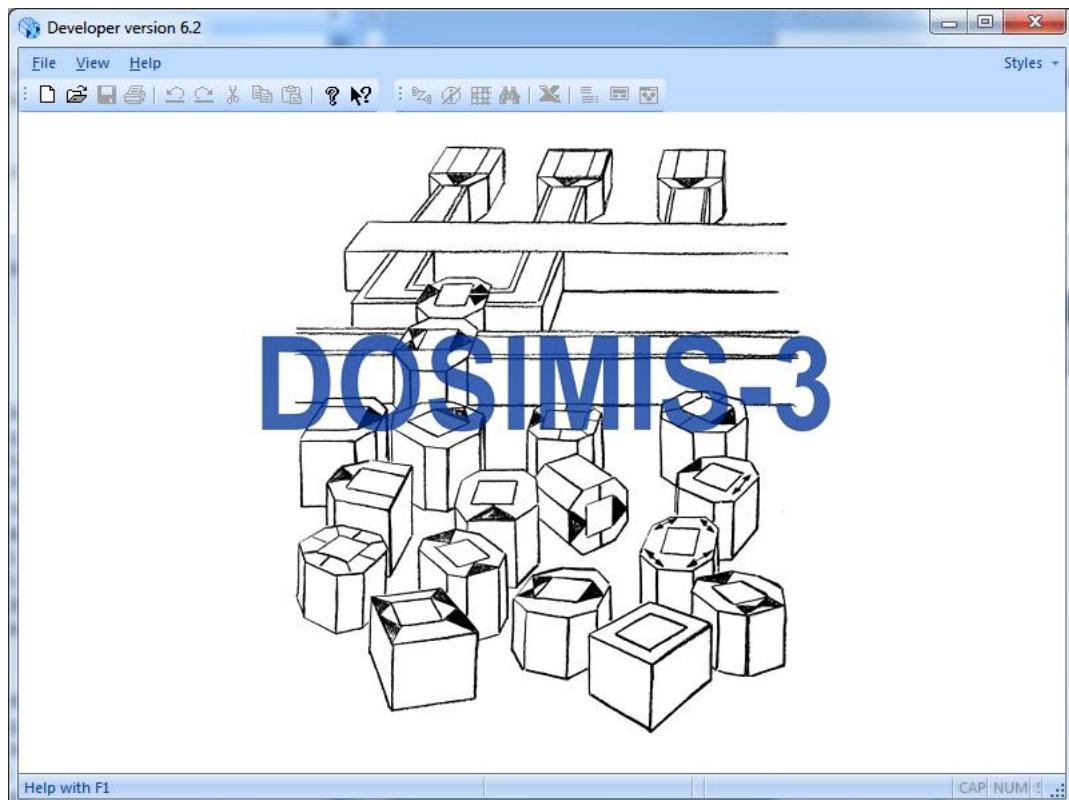


Figure 2.1: Screen layout

The screen is built up consistent to MS-WINDOWS standards. The possibility exists to overlay different windows (menu **Window/Cascade**), to place windows side by side (menu **Window/Tile**) or to display only an individual window on the entire screen interface. For further information please see the MS-WINDOWS manual. The implementation of DOSIMIS-3 is strictly in accordance with MS-WINDOWS.

At the top of a window there is the **title bar**. In the left part of this bar the name of the MFS is shown. Next is the **menu bar**. The **menu bar** presents the possible commands to the user. At the beginning of a session the menu bar is shown as follows.



Figure 2.2: Menu bar at the start of a session

Right below the menu bar there is the **toolbar**.



Figure 2.3: Toolbar

This contains some **buttons**. A menu is processed by clicking the appropriate element with the mouse. Some buttons behave like a switch, having the position **ON** or **OFF**. Other buttons start a process. In this way each part of the menu bar can also be activated by a button. Buttons are not restricted to menus; they are also used especially to set module parameters. Further toolbars, which facilitate an operation, are described in the next section.

At the bottom of the frame, the **status bar** shows control outputs, failure messages of the system or other hints for the user. Messages are displayed in the baseline of the work area. As a result, the user can obtain supplementary information about menu options, simulation run, co-ordinates, and so on.



Figure 2.4: Status bar

If one of the menus are selected by clicking with the left mouse button, then further submenus become visible. If some menus or submenus have light gray text, then these are not selectable at this point in time. For example, a simulation cannot be started, if all necessary parameters or linkings of modules have not yet been set.

2.1.2 Main Menu File

The next figure shows the submenus to the menu **File**:

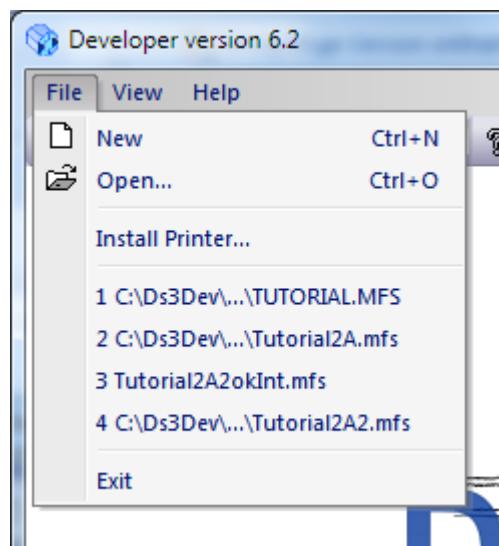


Figure 2.5: Submenus „File“



2.1.2.1 Creating a Model

As a first step the user should enter a material flow system to be dealt with (shortened in the following MFS) with which there are following two alternatives:

If the user wishes to create a new MFS, he must click the **New** button and the modeling of a new MFS can begin.

Shortcuts

Toolbar:



Keyboard:

CTRL+N

If the user wants to load an existing material flow system, he has to enter the name of the material flow system to be processed whereby there are **two** alternatives. If an existing MFS is to be edited, or further processed, the user can call it up by clicking the corresponding name in the menu **File/Open**. (Normally the left button of the mouse is used.) Then the file open dialog opens, where the user can select a saved MFS. The input must be confirmed with <return> or with a click on the **OK** button.

Shortcut

Toolbar:



Keyboard:

CTRL+O

One of the MFS models specified from **1** to **4** can also be selected in the **File** menu. The model of the MFS is then shown on the display.

ATTENTION: If several partitions have been created on the computer, it can come to some inconsistency if one of the models specified under 1 to 4 is on a partition other than the active one (Example: If, while opening, the file is stored under **D:\ds3\ DOSIMIS-3.mfs**). In this case the model should be opened by means of the menu option **Open**. Same applies, if in the file opening menu the file name is indicated directly with partition letters.

2.1.3 Main Menu View

The menu **View** will be described later.

2.1.4 Main Menu Help

The submenus of the menu **Help** are:

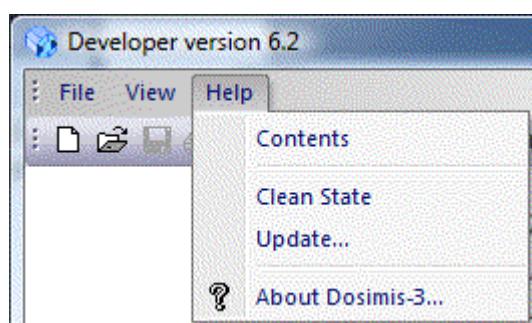


Figure 2.6: Submenus „Help“

Contents leads to an **online help** of DOSIMIS-3-Windows.



Using Help gives information about the help menu.

About Dosimis-3... contains information about the version of DOSIMIS-3.

The usage of the menu item **Update...** leads to a further selection. Here you can choose whether you want to update the dongle or the program.

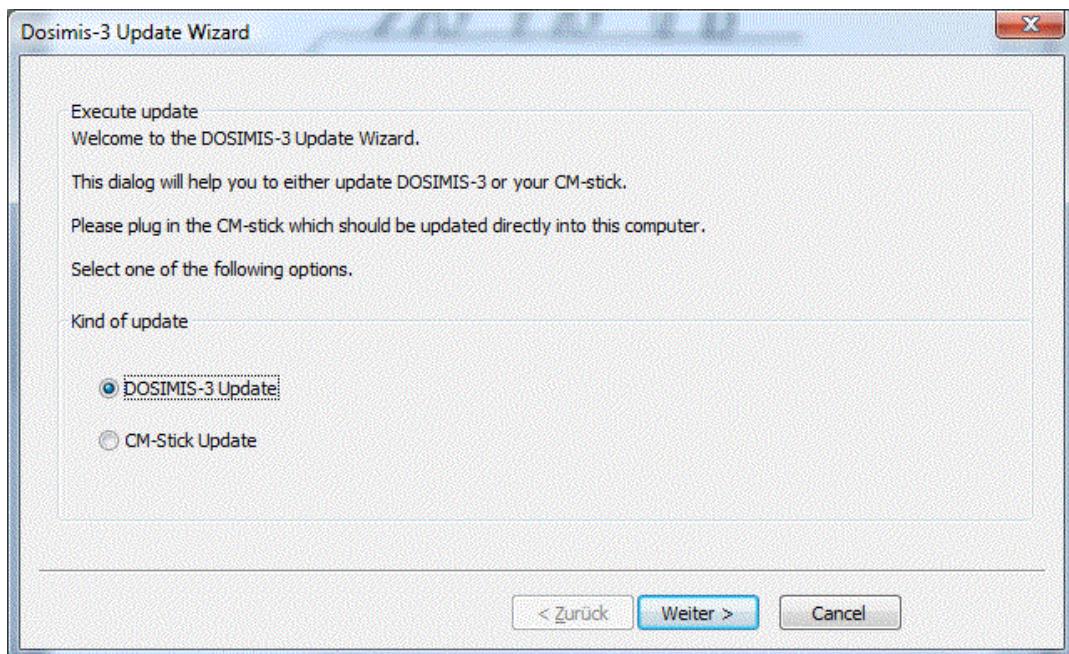


Figure 2.7: Selection Update

2.1.4.1 Update Dongle

On your CodeMeter stick the maximum valid license for the use of DOSIMIS-3 is deposited. If the license is insufficient for your current version, so DOSIMIS-3 switches to demo mode.

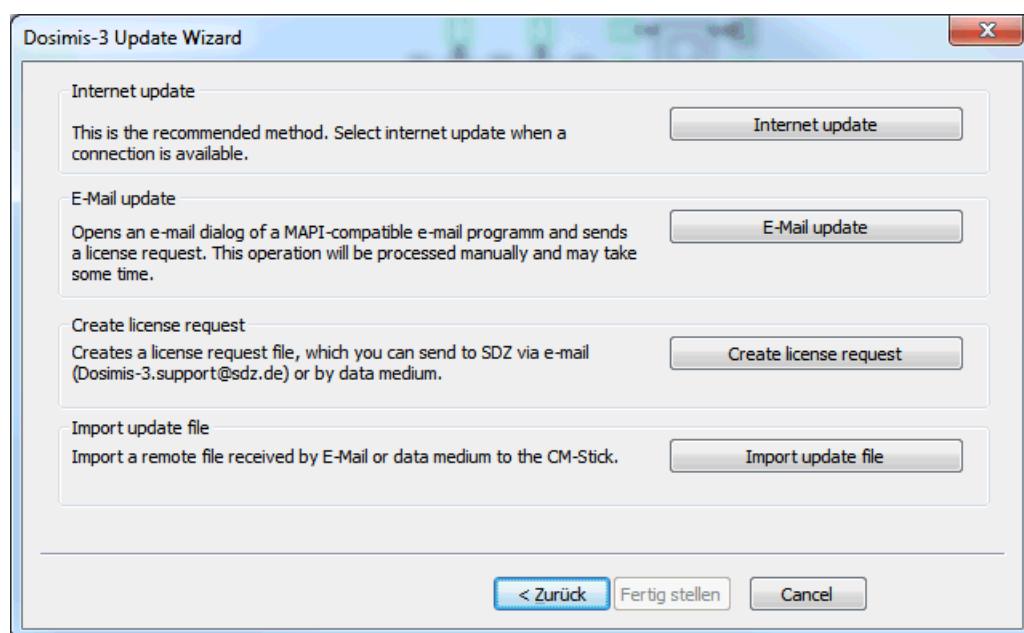


Figure 2.8: Selection Update Dongle



To update the license information on your CodeMeter stick, insert the CodeMeter stick to your PC. Updating a CodeMeter stick can only take place if it is plugged into the computer, where Dosimis-3 is launched. When an internet connection is available, select the entry **Internet Update**.

The system will send information about the inserted CodeMeter stick to the SDZ GmbH. Besides the already existing information on the stick no data is transmitted. The server of the SDZ GmbH examines whether you are entitled to update and transmits the necessary information for the license update of the dongle. This information is transferred to your CodeMeter stick, and after a few seconds the new license is ready for use.

Alternative to direct Internet connectivity the following update paths can be used:

By selecting the **E-Mail update** DOSIMIS-3 attempts on your favorite e-mail program, send an email to the SDZ GmbH, which contains the license information from your CodeMeter stick. This mail will be processed by us manually - please understand for resulting delays. You will receive an email with the necessary update information as an attachment. This information can be transmitted via **import update file** to your CodeMeter stick.

If your email program does not support the Microsoft MAPI mechanism used, you have the option to send the license file by another alternative (eg, an electronic medium, or by manually attach it to an e-mail) to us. To do this, **create license request** and specify the directory for the output file.

The message sent to us for licensing must be processed manually. You will then receive an update file which can be transmitted to your CodeMeter stick by **import update file**.

2.1.4.2 Update Programm

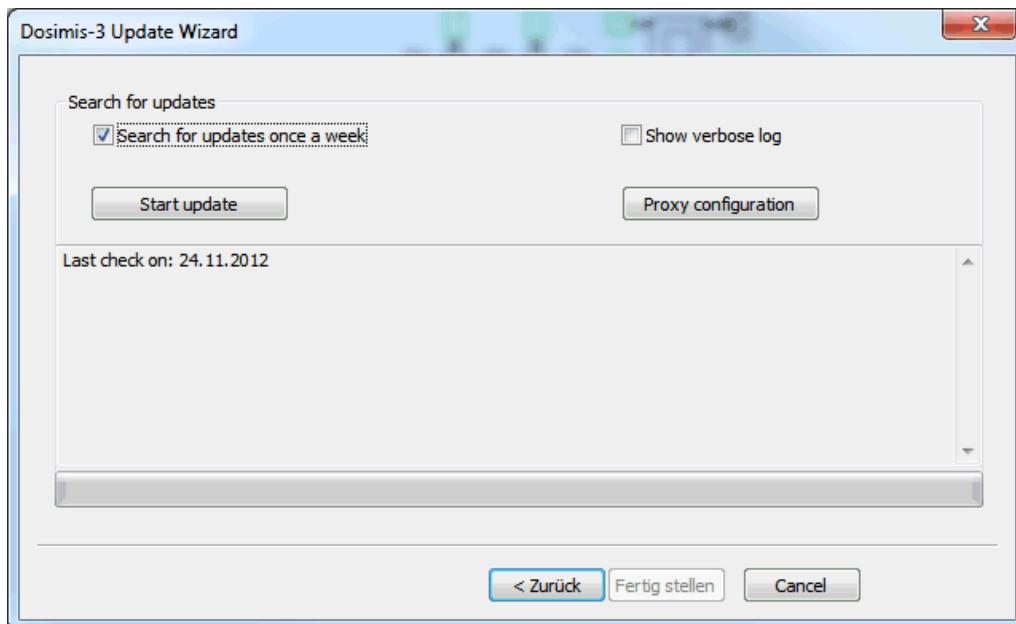


Figure 2.9: Selection Update Program

You get the dialog above. There you have the following options:

- To enable or disable the auto-update mode.



- You can make Proxy settings, unless you have no direct Internet connection. If not directly connected to the Internet, a proxy connection must be established. Your network administrator can help you with the setting.
- Start an update.

If the update process is started, the system searches for an update and downloads it. Before running the downloaded update, you may be asked to finish DOSIMIS-3. Then you have, of course, the opportunity to finish the update process and restart it later.

If an error occurs during download, it may be helpful to **Show the Verbose Log**. You will receive detailed status messages for each operation.

When installing DOSIMIS-3 the cycle of searching for updates is set to one week. So each week the system will look for a program to update automatically. When an update is available, you get a small reminder window that asks you to carry out the update.

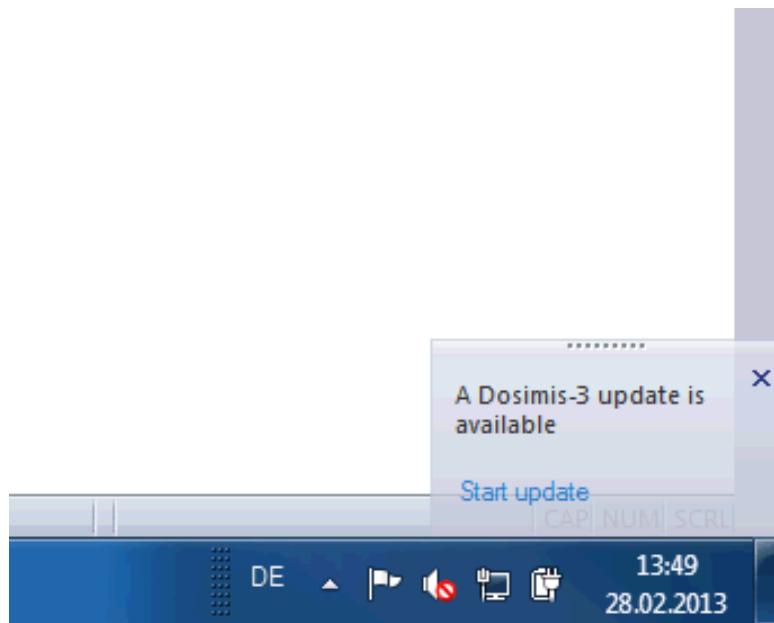


Figure 2.10: An Update of the program was found

Unless you have disabled the auto-update mode, or you want search directly for a program update, you can call *Update...* from the help menu. Select from the following dialog *DOSIMIS-3 Update*.



2.1.5 Work Area

After a MFS is loaded, the work area looks as follows.

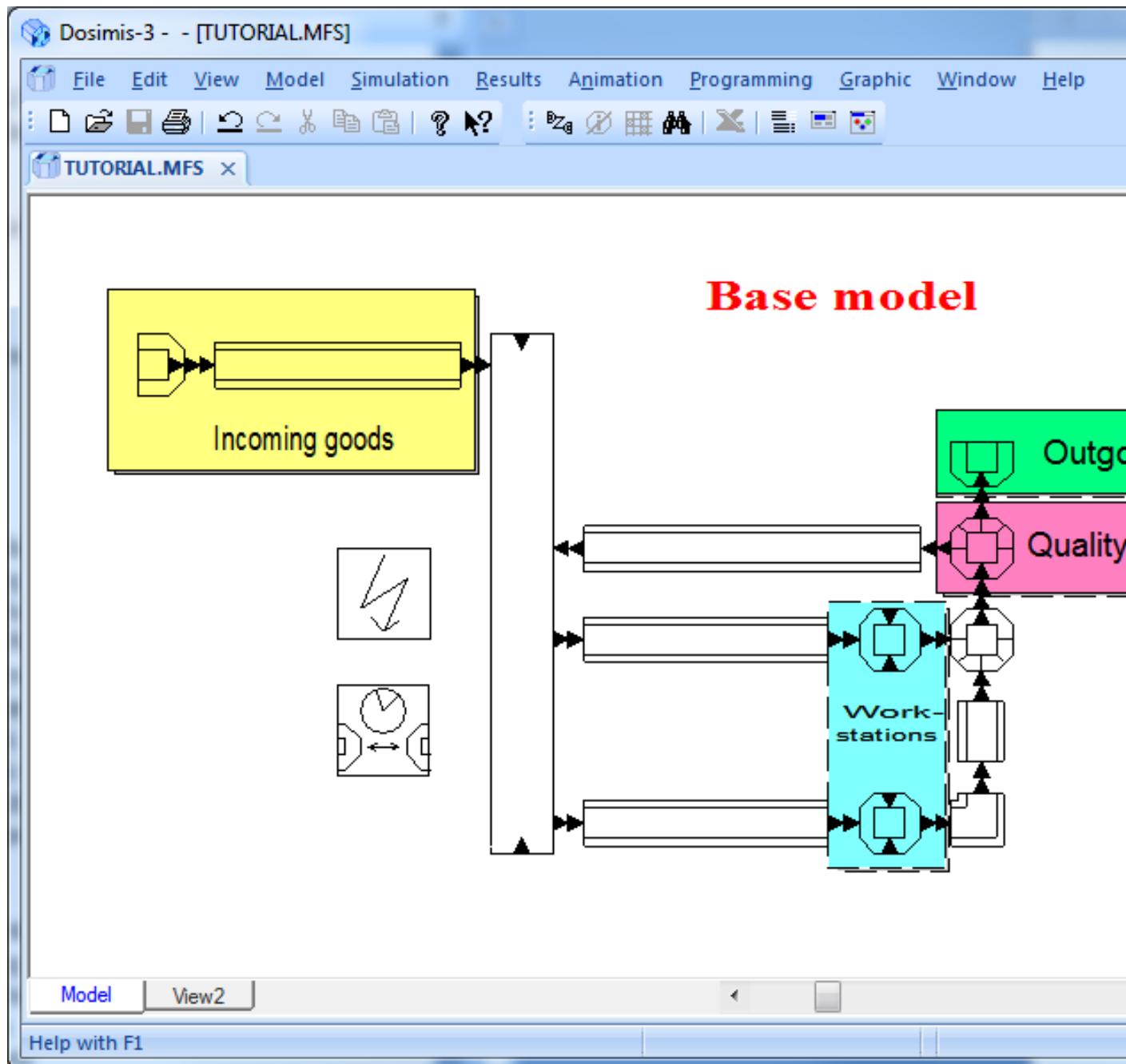


Figure 2.11: Work area

Further menu items which are explained at length in subsequent chapters are available for processing the MFS.

The model is displayed in the work area. The work area can be represented in a maximized size. It normally appears in the normal window size. It is also possible to enlarge and reduce it. The work area is moved horizontally and/or vertically by the scrolling bars.



2.1.6 Structure View

Large models can be structured by different views. Each model automatically possesses the view *model* and connected layer. These are always available and cannot be deleted.

Within the lower area of the work area new views can be created with the help of the context menu.

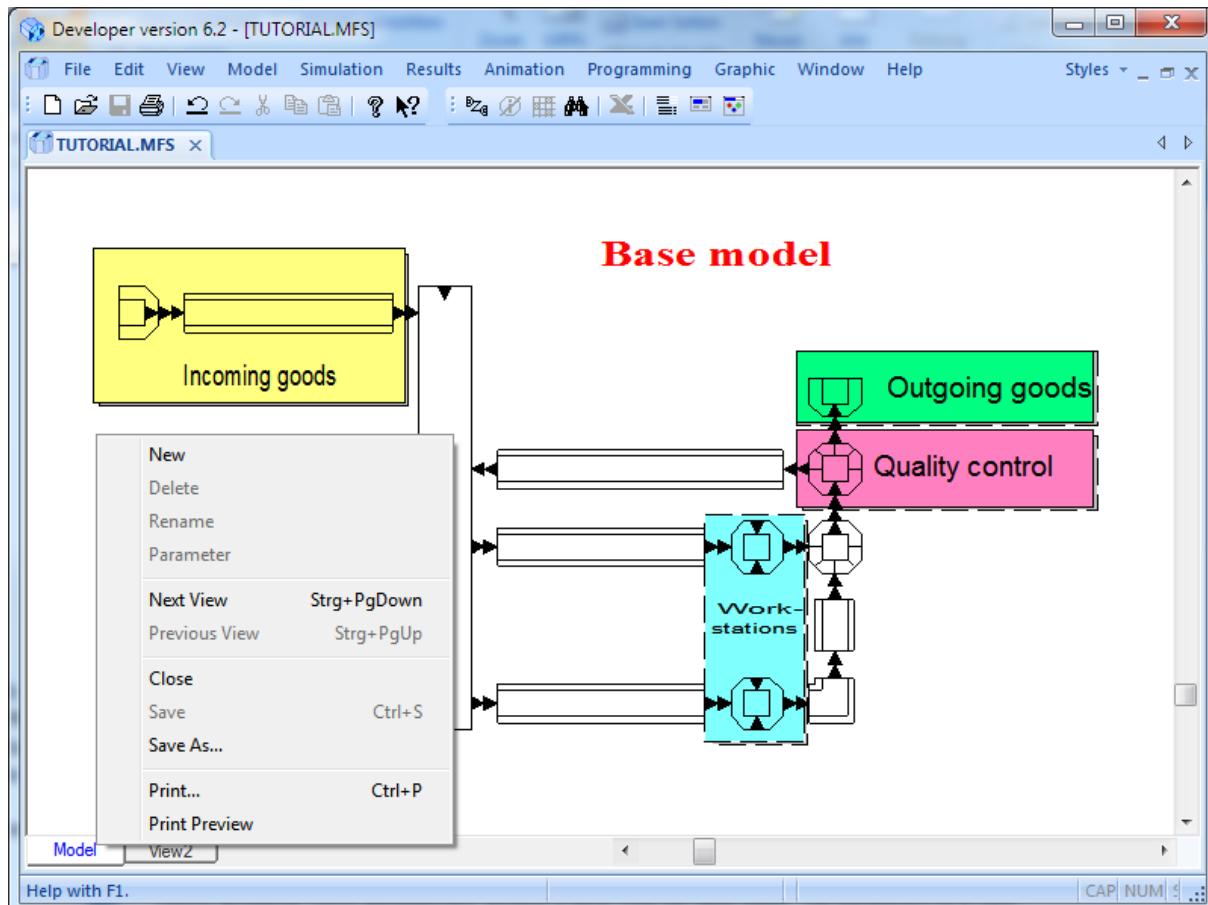


Figure 2.12: New views

Newly produced views automatically possess a layer with the same name. The parameters of a view consist of a list of layers, which are indicated.

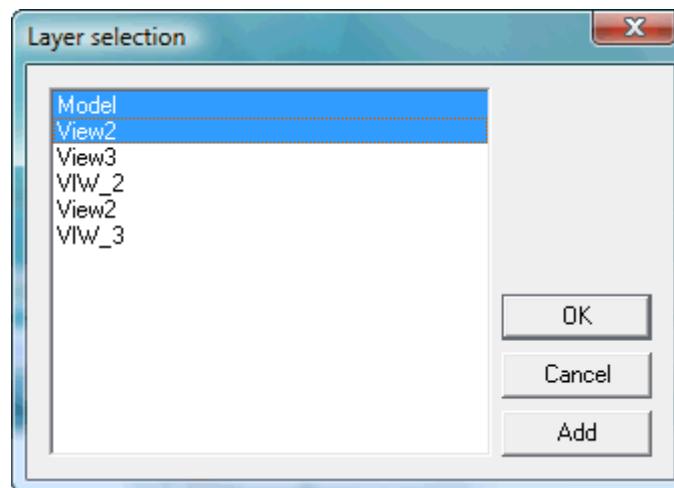


Figure 2.13: Parameters of layer selection

The definition and administration of the layers are described under [model/layer](#). In the view all layers are represented, which are activated in the parameters. Here several layers can be selected.



2.1.7 Toolbars

Beside the standard functions, the toolbar offers further so-called symbol toolbars, which enable fast interaction with DOSIMIS-3. These can be activated at will and deactivated again, by clicking with the right mouse button in the menu area. Symbol toolbars can be shown as either attached at the edge of the work area or unattached on the work surface. Attaching can be disabled if the Ctrl key is held while shifting.

In order to quickly hide an unattached toolbar, click the **Close** button on the symbol's border.

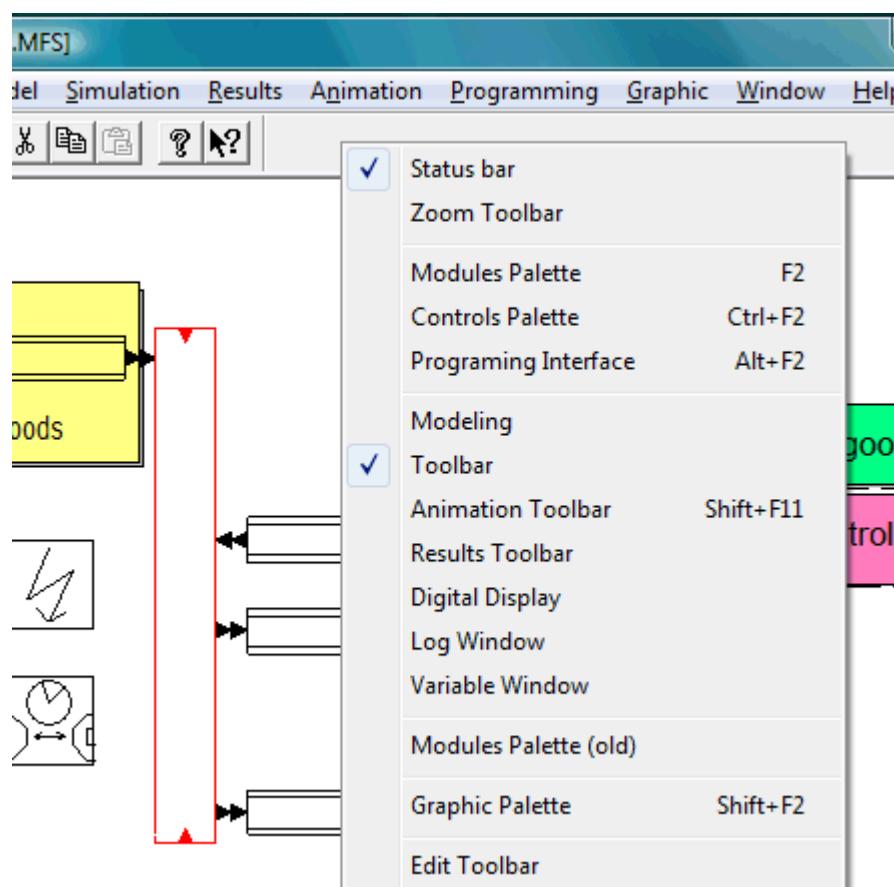


Figure 2.14: Toolbars

Apart from the activation of status bar and function bar, further toolbars are available.

[Zoom](#)
[Modules palette](#)
[Control palette](#)
[Programming interface](#)
[Modeling](#)
[Animation toolbar](#)
[Result toolbar](#)

[Digital display](#)
[Protocol](#)
[Variable window](#)
[Modules palette \(old\)](#)
[Graphic palette](#)
[Edit](#)

A detailed description is to be inferred from the appropriate sections.



2.1.8 Tooltip

DOSIMIS-3 supports tool tips. These are insertions, which the current item under the mouse pointer briefly describes. This is activated, when the mouse pointer hovers for some time over the item. The following picture shows this by the example of the modeling palette. Furthermore a detailed text appears in the status bar as a supplement to the quick information.

Tool tips can also be activated, when the cursor stays for a while either on a DOSIMIS-3 item (name and number of the item is displayed there) or on some input fields in the parameter dialogs.

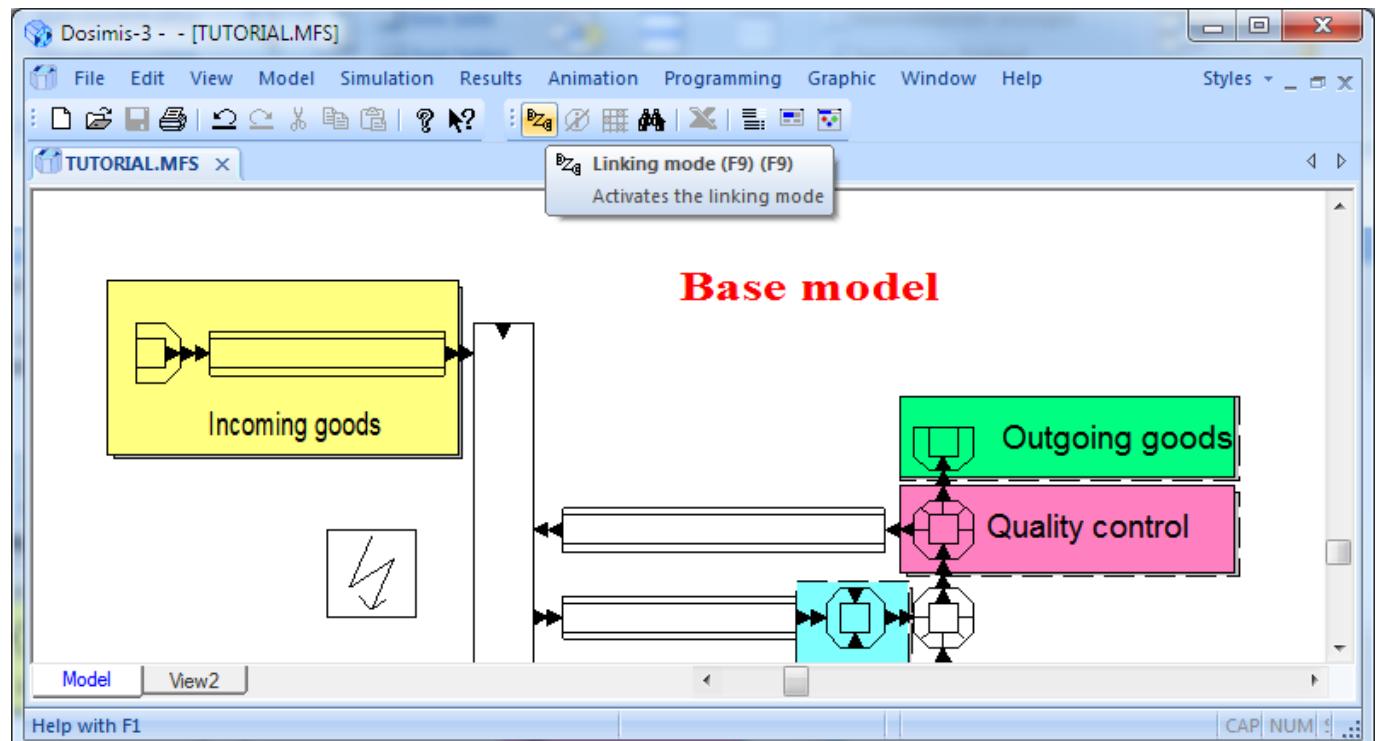


Figure 2.15: Quick information



2.1.9 Context Menu

DOSIMIS-3 offers context menus at all places during the work in the work area. These are activated by clicking with the right mouse button in the work area. The term context menu means that functions of a selected item are displayed, which are meaningful in this context.

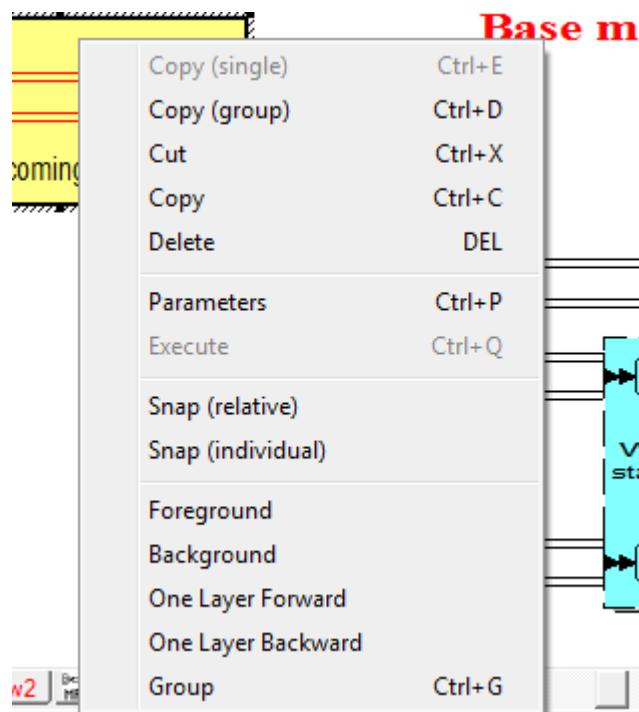


Figure 2.16: Context menu

For example, the context menu of a graphic item offers the operations, which are admissible on all items of DOSIMIS-3 (Copy, Delete, Parameters), as well as the graphic-specific operations (Foreground, Group, ...). Most options also appear in the menu. Therefore the description is to be inferred from the appropriate sections.



2.1.10 Positioning of Modules

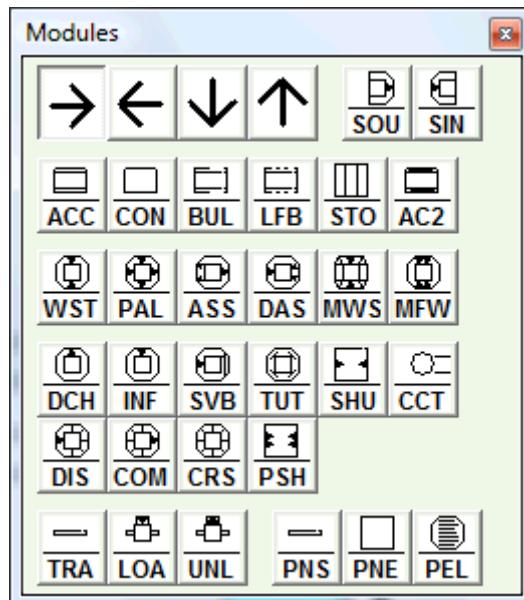


Figure 2.17: Module Palette

In order to place elements, a window with the available elements and four arrows is opened via the menu selection **View/Modules palette**. The arrows indicate the direction of material flow within an element that can be selected. This is necessary, in order to display the element correctly in the display area. The symbols represent the elements. If one of the orientation arrows is shown depressed and in gray color after insertion of an element symbol, it means that this orientation has been chosen. Other orientations can be selected by clicking them.

To start with the placement of elements, it is necessary to set the orientation of the element by



choosing one of the four arrows situated above the module symbols. These arrows simplify the orientation and show the flow of the objects throughout the whole system. The current orientation is represented by a light-gray background, thus distinguishing it from the others. When placing a new module, its orientation can be changed by clicking the right mouse button before placing the module.

The positioning and linking of modules as well as the graphics operations are done with the mouse. In order to start building a model of an MFS the user first has to create and position the modules needed. The use of the mouse in connection with the elements positioning is described in detail later. The elements can be divided with regard to their positioning into two different types, elements with fixed size and elements with variable size.

Connecting of the elements is possible only in the mode **Linking active**, which however prevents a positioning of the elements. Therefore first the elements should be placed and then connected in the second step.

Placed modules are shown in **green**. This means, that these are not parameterized completely, so they are inconsistent. When the modules are connected with their neighbors and the parameters are set, their color changes to **black**. Information on incorrectly parameterized modules can be seen in the error file (*.chk). Further information can be found in the section **Consistency check** of chapter [Simulation run](#).



In addition to the modules palette there is another **palette for controls**. While the modules palette contains elements, which build the material flow, the control palette contains elements, which externally affect the material flow.

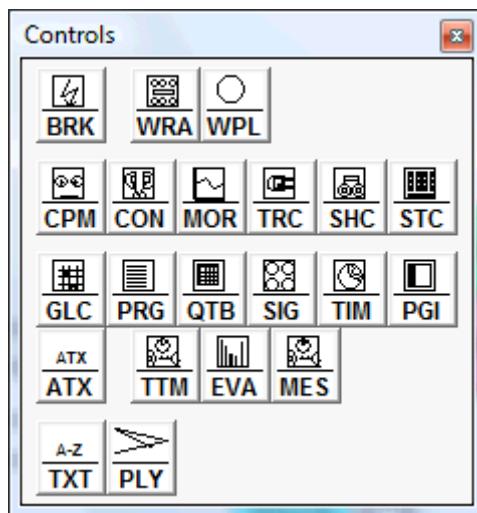


Figure 2.18: Control Palette

Both palettes can be changed in format and fixed at the boundary of the work area.

2.1.10.1 Direct Positioning

First an element icon has to be selected with the mouse and then it can be placed in the work area. An icon of the element follows the mouse arrow. By clicking with the mouse, the element is placed.

In case of a mistaken placement the icon can be moved by clicking this element and moving it with the mouse button pressed. It is also possible to move an icon by using the menu **Edit/Move**. But first the icon is to be selected by **one** click. The selected icon is **red**. The display area is placed underneath a grid. The grid points are the points of reference for the final positioning, i.e. an element can be moved not continuously over the display area, but only from grid point to grid point. The grid can be disabled by pressing the **CTRL - key**. It is finally placed by clicking the left mouse button.

2.1.10.2 Positioning of Modules with variable Size

Some modules are variable in size or form. Conveying buffer or bulk section tracks as well as the size of shuttles and paired shuttles can be entered by means of the mouse. Once again, one of the elements from the element palette is firstly selected and brought over the display area to the desired position by means of the mouse. After clicking with the mouse at the desired position the beginning is marked. The expansion of the element in a direction can be influenced by moving the mouse. It is shown with a line. Further clicking of the left mouse button produces different way points for these items. The end of the element is set by clicking the right mouse button. The process orients itself also here at the grid put underneath at the display.

For elements with more than two way points, the last point can be removed during placing. With pressed **SHIFT key** after pressing the **left** mouse button the last way point is deleted.



For the elements *multiple workstation*, *shuttle*, *paired shuttle* and *storage* right-angled branches are not permitted. For these elements further clicking of the **right** mouse button sets the endpoint of the element.

2.1.10.3 Modification of Placing of Modules with variable Size

Placed elements can subsequently again be positioned by graphic operations (Move, Mirror, ...). The process of placed modules with several bases (accumulation conveyor,) can be subsequently redefined. If exact one element is selected, this can be placed again in the layout by **Model/Replace Module**. The parameterization and the links are preserved.

2.1.10.4 Full-Scale Placement of Modules of variable Size

Completely parameterized accumulation conveyors can serve as reference in order to determine the capacity from the length of the representation. For this the entry of **default values** is to be selected from the context menu. Thus by the capacity and the representation, the capacity per unit length is calculated.

From now on, the capacity of a new accumulation conveyor is registered automatically from the length of the representation. This applies to all modules with a capacity such as conveying section, bulk section, etc.

The menu item can be selected only if exactly one module is selected, this module supports this functionality (accumulation conveyor, conveying section, ...) and the module is completely parameterized.

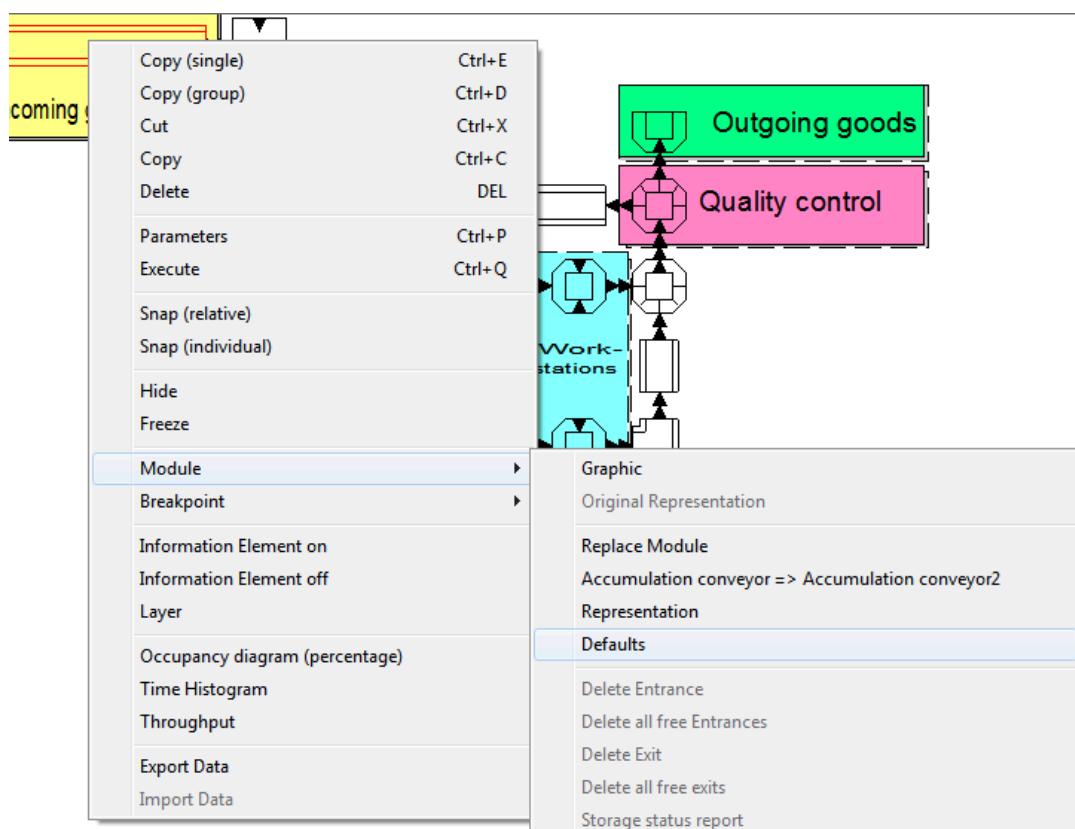


Figure 2.19: Setting default values via context menu



Furthermore with the selection of default values also the speed and the object length of the defaults are reinitialized.

2.1.10.5 Subsequent Change of the Module Type

The type of a module can be subsequently changed to a limited extent. This requires the new module type substantially coincide with the original module type or even have more functionality.

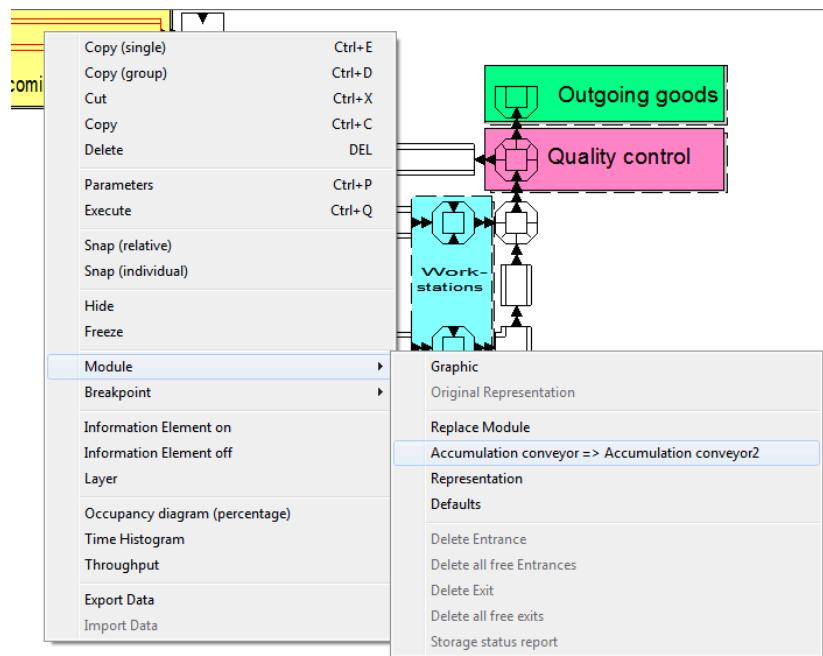


Figure 2.20: Change module type

When transferring the parameters as many as possible data will be transferred to the new module.

The following table shows which conversions are supported:

From	To
Workstation	Multifunctional Workstation
Assembly	Multifunctional Workstation
Disassembly	Multifunctional Workstation
Accumulation Conveyor	Accumulation Conveyor2
Conveying Section	Accumulation Conveyor2
Discharger	Crossing
Distributor	Crossing
Infeed Element	Crossing
Combining Station	Crossing

2.1.11 Linking of Modules

The next step in building an MFS is to link the modules. Thus, the exits of the modules are linked to the entrances of the successor. The connection between two modules is clearly defined by using junctions.



To link the modules the menu item **Edit/Linking active** has to be selected.

Shortcuts

Toolbar:



Toolbar:

Modeling

Keyboard:

F9 or Ctrl+right mouse button

In the mode **Linking active**, the junctions are displayed in green. Additionally the button is displayed as depressed and the menu item is indicated with a check mark.

In the status bar the message **Junction: Select starting element** appears. If the user now selects a module by using the left mouse button, the system states **Junction: Select end of junction**. The destination module is now defined giving the user the possibility to influence the track. Thus, it is not necessary to always choose a direct link (straight line). After having chosen the destination module, the message **Define way points (M1, delete (Shift+M1), end M2)** appears. By using the left mouse button, further points can be set, with the <Shift>-key and the left mouse button the last point can be deleted. The input can be terminated by using the right-mouse button and the line will be drawn.

The starting point of a junction is specified by the type of module it is connected with. With some modules (shuttle, conveying circuit ...) however the cursor position during selection determines where this junction is placed at the module. These starting points can be changed later. For this the starting point can be dragged while pressing **Shift** and **left mouse button**.

If all exits of the initial component and/or all entrances of the target component are already busy, these are increased dynamically. This number is maintained after the deletion of junctions. A change (in particular a reduction) of the exits/entrances of a module can be carried out in the module parameters (see module parameter chapters).

2.1.12 Linking of Controls with Elements

The mode **Linking active** is also necessary when changing control lists (work area, failure ...).

Selecting a control displays it in blue. Now you can select a module/junction to assign it to this control. The element is displayed in red. After selecting a control, all modules/junctions, which belong to this control, are displayed in red. Now you can add or remove modules/junctions by mouse-click to/from this control. Linking can be completed by clicking in the work area without selecting an element.

With the help of multiple selection, several modules can be added to a control at the same time. For this a rectangle is to be dragged with the mouse while pressing the **SHIFT key**. All modules, which are in contact with this rectangle, are assigned to the control.

DOSIMIS-3 supports 2 different modes when connecting controllers:

- First mode:
If another control is selected, then it will appear in blue and becomes the active control. All selected elements will be assigned to the new active control. Now, if a control should be connected with another control (eg. disrupting the flow time measurement or superposition of failures), the **Shift key** is pressed in addition.



- Second (alternative) mode:

When a control is active (drawn in blue), then all selected controls will be connected automatically (without pressing the shift key). Another control can become the active control, if no element was selected previously (press escape or click on the white area of the model).

In new created models the alternative mode is active. The connecting mode can be changed at Model/Configure.

2.1.13 Properties of Junctions

If the user wants to see the properties of a junction, the appropriate junction has to be selected. The user is not allowed to make changes to the number of the junction. For this purpose, a window is opened by double-clicking the junction. Already placed junctions can be modified via the menu **change way points**.

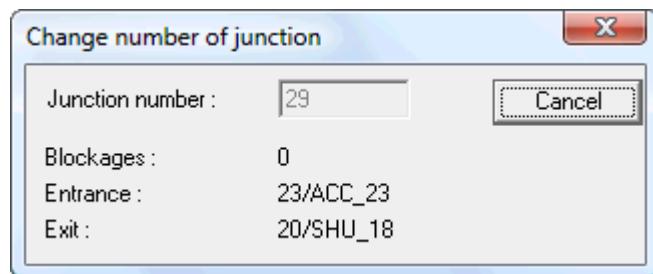


Figure 2.21: Change number of junction

2.1.14 Deleting of Junctions

A linking of modules is deleted, as the junction is selected and removed afterwards from the layout. The reference in the module remains however, so that no parts of the parameterization are lost. Open references in the list of the junctions are marked as *not linked*. If the module is connected again with another module, the first unused connection is used again. The following remarks are described by the example of the entrance junction and are valid also according exit junction.

4 <input type="checkbox"/> Entrance(s)			2 <input type="checkbox"/> Exit(s)		
No.	Junction	From module	No.	Junction	To module
1	24	ACC_100	100	1	27
2	not linked			2	62
3	26	ACC_99	99		TUT_137

Figure 2.22: List of entrances and exist

The number of connections can be reduced, as the number of entrances in the parameter mask of the module is reduced. The junctions at the entrances with the highest numbers are deleted and the parameters are adapted accordingly.

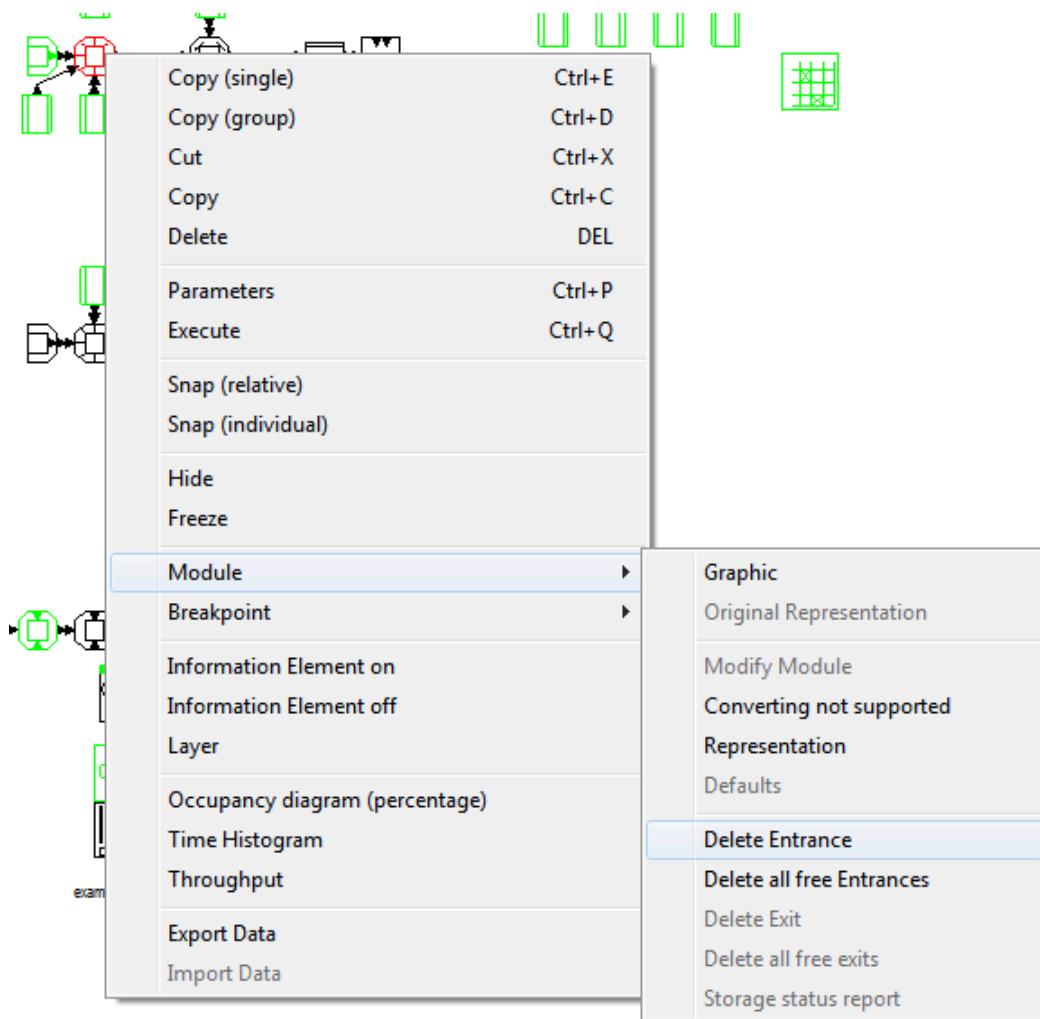


Figure 2.23: Delete entrance

Alternatively the number of not occupied connections can be reduced by the context menu of the module. This method works however differently. The first reference not connected with another module is deleted and afterwards the appropriate parameters of the module are adapted. In addition the appropriate junction must have been deleted first. If several entrance references are to be corrected, this method must be called several times.



2.1.15 Object Transfer

The main importance of junctions is the fact that they transmit information from one module to the next one. Thus, the modules can react to states or trigger reactions of the adjacent modules in case of changes of the state.

Four different states can be distinguished: ANNOUNCED, OCCUPIED, WAITING and FREE. For a better understanding, a link can be regarded as the junction of three Boolean variables ANNOUNCED, OCCUPIED and WAITING. The state FREE is reached if both OCCUPIED and WAITING are set to FALSE.

Given the following situation



Figure 2.24: Example of links between modules

with both links first being in the state FREE.

If an object now advances to the end of module 1, the variable ANNOUNCED in junction 1 is set to TRUE. If the junction is not blocked, the state of WAITING is also set to TRUE and the object announces its arrival to module 2. The following module (here module 2) then decides if the object is allowed to enter or not. If module 2 is ready to let the object enter, the variable WAITING in junction 1 is set to FALSE and OCCUPIED is set to TRUE. If the object has fully entered module 2, OCCUPIED is set to FALSE again.

Under certain circumstances it is even possible to have WAITING and OCCUPIED set to TRUE at the same time. For example, if modules 1 and 2 are tracks with different conveying speeds and $\text{speed1} > \text{speed2}$. If two objects are conveyed in track 1, in junction 1 WAITING is set to TRUE as soon as the first object wants to enter track 2. If the object is accepted, WAITING is set to FALSE and OCCUPIED is set to TRUE. Because of the different conveying speeds the second object wants to enter module 2 before the first object has left. Now WAITING (because the second object wants to enter) and OCCUPIED (because the first object is not fully conveyed yet) are set to TRUE.

Additionally for execution described above an examination takes place with some elements of whether the last object left the element completely (the value of "OCCUPIED" junction must be FALSE for all outputs), much before operational sequence is initiated. Thus, for example,

- in a workstation the work can only then be begun,
- in a crossing the next output can only then be determined,
if the last object from the element has been driven out.

This examination is not made, if the element has a **length of 0**. Then the next step continues immediately. The loading length 0.0 is admissible with workstation, assembly, disassembly, distributor, combining station, crossing, discharger and infeed element.



If a module has several entrances (e.g. 2) the entrance accepting the object is determined by the right-of-way strategies. If in module 2 as well as in module 1 an object is waiting, the right-of-way strategy defined in module 3 decides which object is to be delivered. Module 3 sends the message **ready to enter** to one of the two other modules; this module then delivers the waiting object to module 3. The other module receives the message **not ready to enter** and cannot deliver its object.

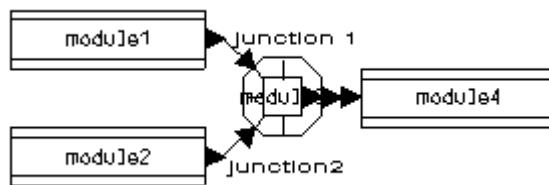


Figure 2.25: Example of links between modules

2.2 Menu File

If an MFS is loaded, then the following submenu appears:

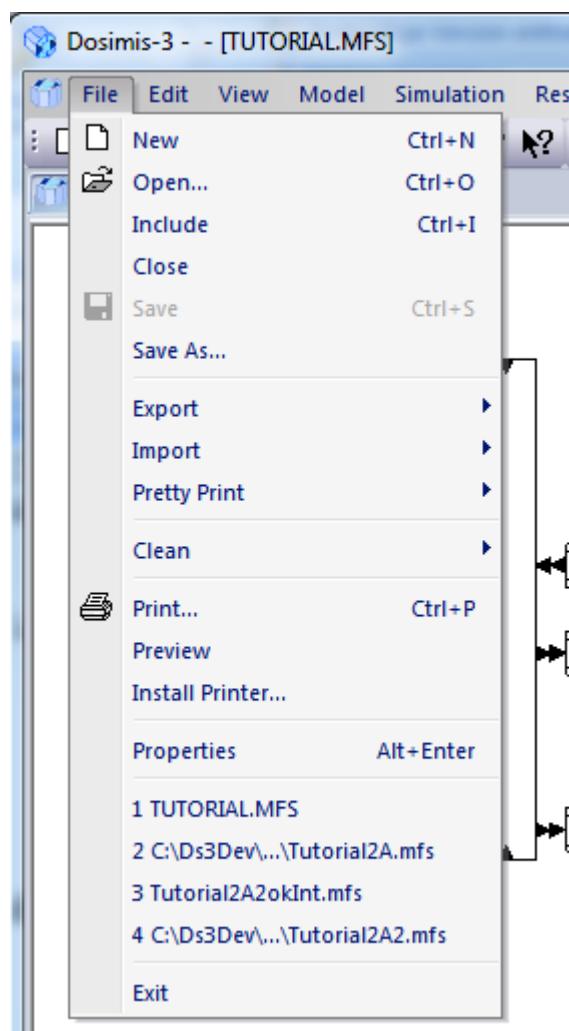


Figure 2.26: Submenus „File“, if an MFS is loaded



You can open several models at a point of time. You can create a **New** model from the menu or **Open** an existing model.

The option **Include** allows a combination of several material flow systems. Different systems can be summarized into one system by entering the name of the model to be inserted. The layout of the inserted model appears in the display area and can be shifted with the instruction **Edit/Move** to its final place, since all items of the new model are automatically selected. Since combining partial models can sometimes cause conflicts, a query on assigning of names for all affected components assigned to decision tables (network attributes, element variables) appears. See also Paste (from clipboard).

Close has can be used to complete the work with the active material flow system. Possible changes, which have been made since the last save, will not be saved. It is recommended to save the new data with **Save**. Here the command **Close** only signifies that the current model is closed, DOSIMIS-3 however remains open. When the user wants to continue with another MFS, he can do this with **Open** without leaving the editor. When selecting **New** he can start modeling a new MFS.

If the user changes the current MFS, the new or changed data can be saved by using the operation **Save**. The system then responds with *data saved* and the user can continue working on the graphical model. It is recommended to save the model regularly to avoid losing data in case of unexpected events. The **Save** option only affects the *name.mfs* and the *name.dar* files. The *name.slg* and *name.tra* files remain unchanged. As long as no changes are made, the menu **Save** and the corresponding button remain enabled.

Shortcut

Toolbar:



Keyboard:

CTRL+S

If the name of the current MFS should be changed, this is done via the option **Save as....** The old name is replaced by the new one. The file will be saved under a new name.

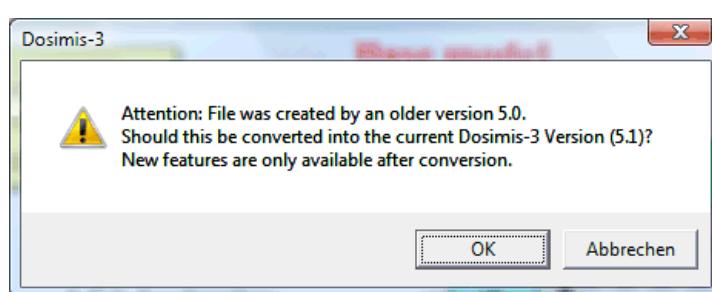


Figure 2.27: Warning during saving of older models

When the user saves a model, which was created with an older version of DOSIMIS-3, the message above can prevent the model from being overwritten in the new format. Since with saving in a new format, additional or changed information is added, the model cannot be converted back into the old version.

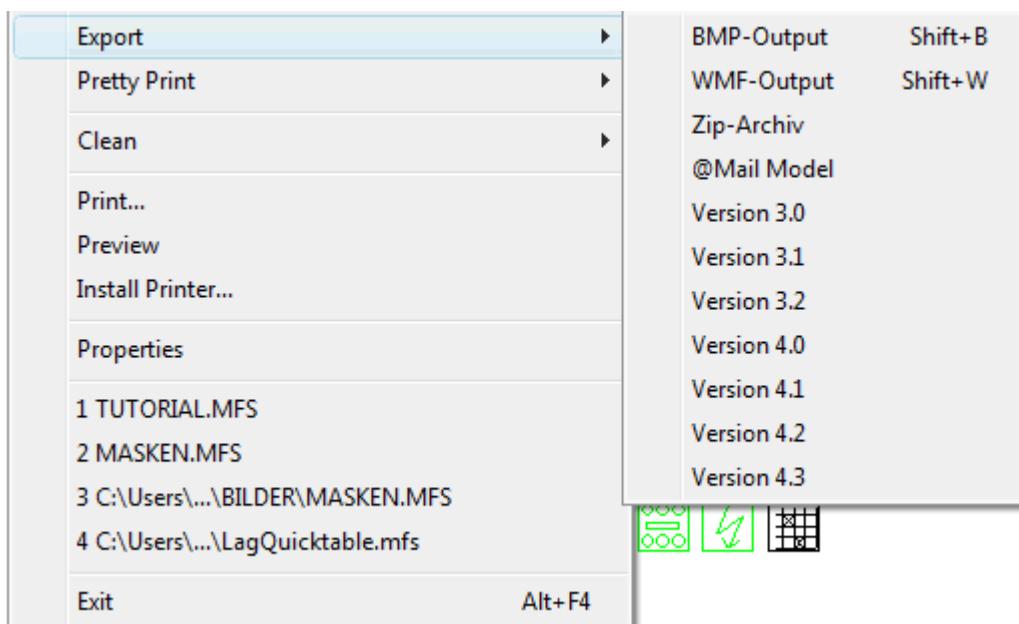


Figure 2.28: Menu „Export“

Under **Export** the graphic of the layout can be copied as a **bitmap** or **Windows Meta file** into the clipboard. From this, these diagrams can be transferred into other programs by the standard menu option **Insert**.

With the help of the menu option **Zip Archive**, all files, which belong to the material flow system, are stored into a Zip file. These are apart from the parameter files, generation data of the sources, if these exist, the bitmap files and some more. The file carries the names of the material flow system and ends with *zip*. This entry is disabled, when the program Minizip.exe could not be found in the program folder.

This file can be sent away by **@mail model**. For this the appropriate browser must be configured (tested with Outlook and Netscape). The default address Dosimis-3.support@sdz.de is entered. This entry is disabled, when the program Minizip.exe could not be found in the program folder.

Furthermore, the possibility exists of storing the model in the format of older versions (**Version 2.3**, **Version 3.0**, **Version 3.1** and **Version 3.2**). This occurs then without the additional information, which was added with the current version to the material flow system.



Figure 2.29: Menu „Pretty Print“

By activation of the menu option **All** the parameters of the material flow system are written into a file in a more readable form. If the menu option **Pretty Print/Decision tables** was selected, only the structures of the decision tables (also network attributes, global decision tables, etc., however, no conveying engineering parameters) are written. By **Pretty Print/Selection** the parameters of the selected items are written. The file name results from



the file name of the material flow system and the extension *.pty* and can be changed by the user.



Figure 2.30: Menu „Clean“

The submenu **Clean** offers two options to delete temporary files created during simulation from the project folder. With **Model** only those files are deleted which belong to the current model. With **Directory** all temporary files of the current folder are deleted, regardless of whether they belong to the current material flow system or not. This is fixed by the extension of the files. These are aside from the backup files also the statistic file and the trace file. A collection of the DOSIMIS-3 files can be found at the [end](#) of the documentation.

The menu **Print...** refers to the print output of DOSIMIS-3 layouts and diagrams of results.

Shortcut

Toolbar:

Keyboard: no shortcut

The option **Preview** gives a view of the model of how it would be printed.

The menu **Install Printer...** is used to select a printer.

Through the menu option **Properties**, defaults of DOSIMIS-3 can be parameterized. Apart from the scaling of the Undo function, the simulator and the interfaces, statistics are created for each model.

To leave the simulator DOSIMIS-3 click **Exit**. A safety check prevents inadvertent deletion of a model.



2.2.1 Properties User Interface

The representation of the times can be switched by **Show Minutes** between the formatted outputs and the pure minute display. The input can furthermore take place in both formats. This switch is considered only when creating the dialog.

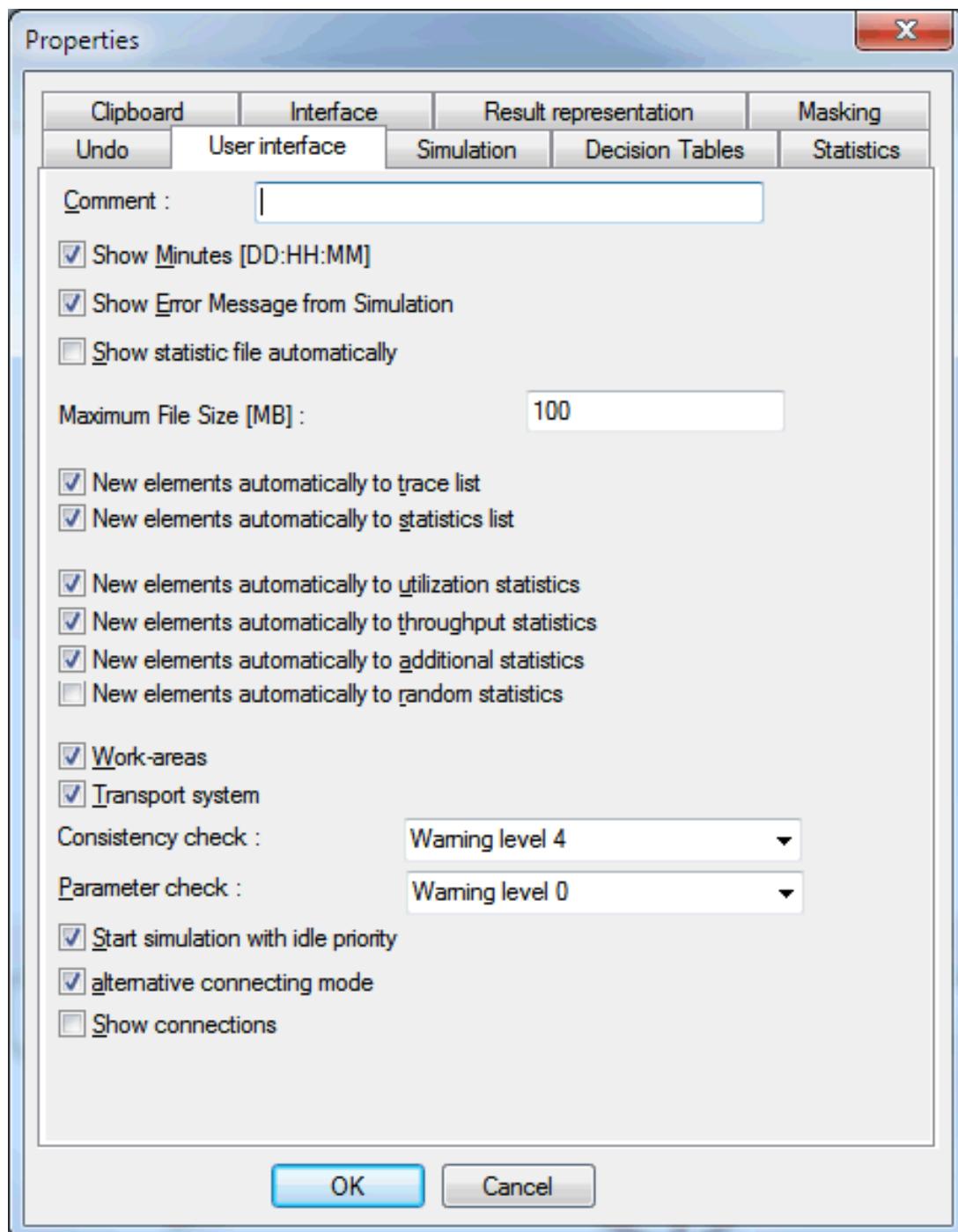


Figure 2.31: Properties of user interface

Before reading the results a safety query takes place for large files, since with the simulation large quantities of data result accumulate, based on experience. Since this can be obstructive, the possibility exists of influencing this value as a function of the requirement and computer configuration. In particular with automated simulation (optimization or via Excel) the usual query can be deactivated.



Furthermore, the statistics lists to be automatically assigned for a new element can be fixed.

Both switches **Work areas** and **Transport system** are for the overview of the input dialogs. If the model should not contain work areas or transport systems, these switchers can be deactivated. All input fields according to these parameters will then be hidden. In every case all input can be activated afterwards.

Additionally a warning level can be defined for the **Consistency check** (from one to four). This means, that aside from the errors during the parameter check, warnings will be printed out, which give hints to possible sources of errors.

The switch **start simulation with low priority** ensures that the simulation run is not so much system taking. The run will be at least a little bit longer, but the computer is not as heavily stressed, so you can still work in parallel with other programs.

The **alternative connecting mode** changes the behavior when connecting controllers with modules, or among themselves.

Show connection means that during the mode *connecting active* the elements connected to an active control are highlighted not only by color but also be marked by a connecting line.



2.2.2 Properties Masking

With the dialog *Masking* some information's of the model can be faded in permanently.

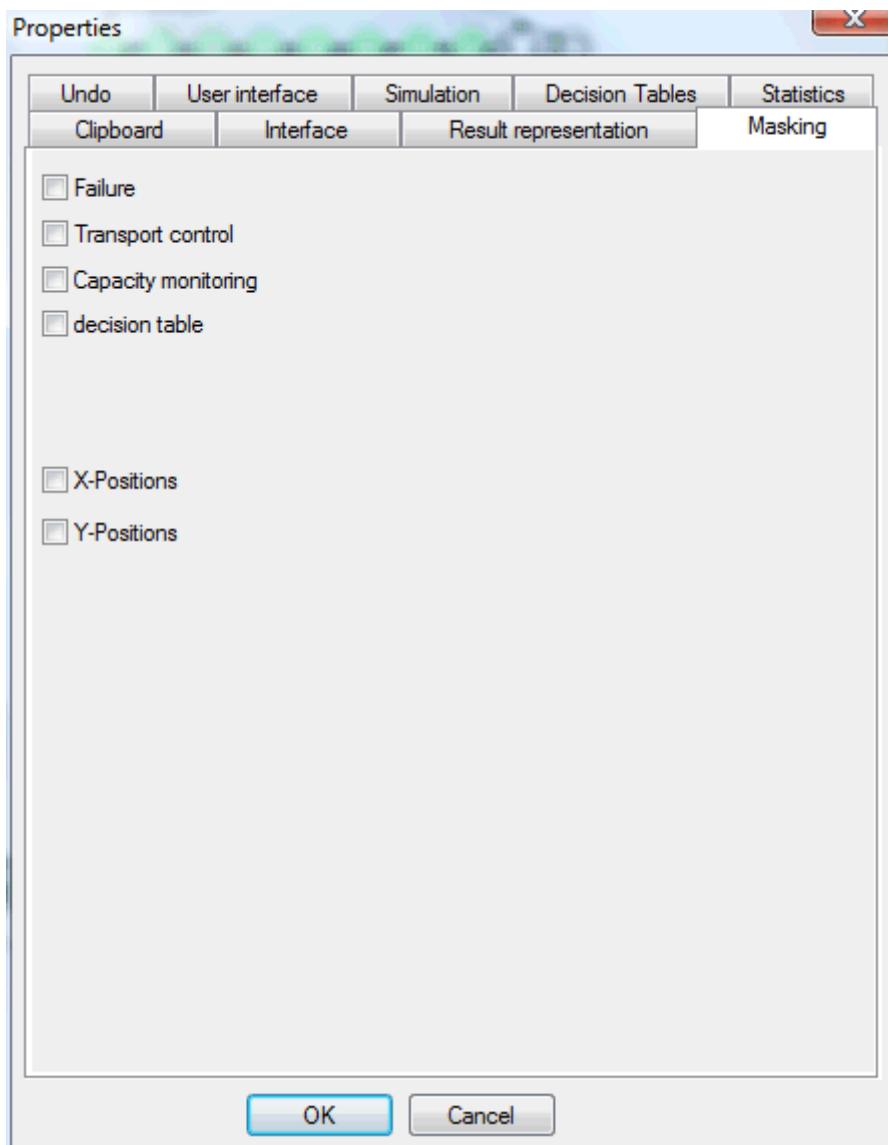


Figure 2.32: *Masking*

Between the names of some groups of elements the coordinates of the positions of a shuttle can be faded in.



2.2.3 Properties Simulator

If calculation program and user interface of DOSIMIS-3 are situated in different directories, the path to this simulator can be made here. Additionally a directory can be indicated, in which the temporary files of the simulation (statistics and internal message log) are written. Thus the data-intensive operations can be shifted on the local computer.

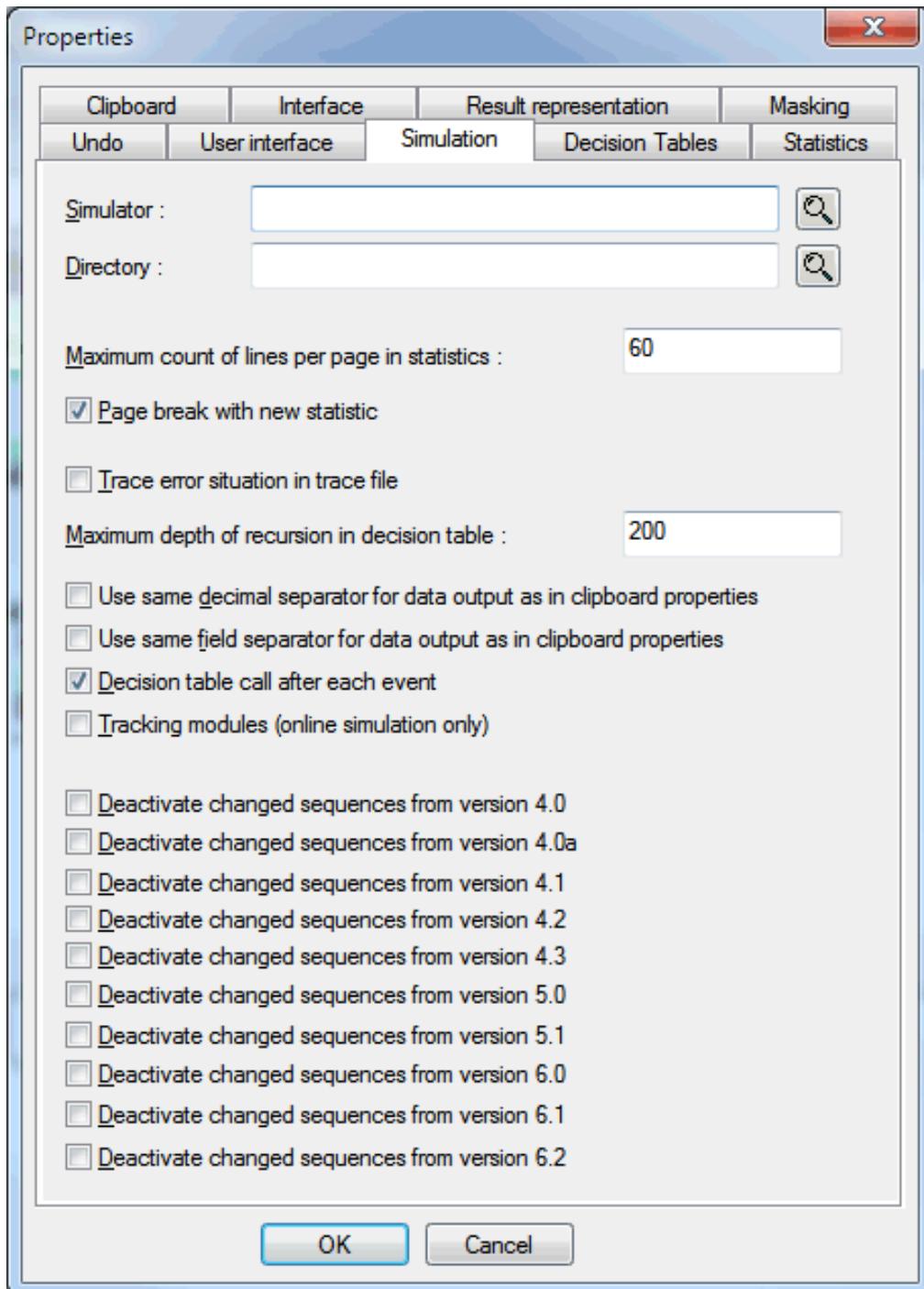


Figure 2.33: Properties of simulator

With the fields **Maximum count of lines...** and **Page break...** the output of the statistic file can be influenced.



When the check box **Trace error situations in trace file** is activated, in the case of error the output is taken in the error file and also as a text output in the trace file. So the status at the point of time of the error can be analyzed during animation.

The **Maximum depth of recursion** in decision tables can be limited with this parameter so that unintentional loops can be detected without a crash of the simulation program.

During the output on a file the separators are taken from the system configuration. When different separators are defined for the output to clipboard, these can be used for the output to a file by the switch **Use same separator for data output as in clipboard properties**.

After each event of an activated module of a decision table the table will be evaluated. To disable this and for the analysis only of the relevant events like entering, exiting, ... this can be disabled by the switch **Decision table call after each event**.

If the switch **Tracking modules** is activated, the way of an object through the material flow will be logged in online - simulation for each object. For each module, in which the object was, the time of the entrance and the exit is recorded. This information can be seen during online - simulation in the variables windows.

The switch **Deactivate changes sequences from version 4.0 (version 4.0a, ...)** deactivates changed behavior of the actual version. Thus, old models lead to the same simulation output with the new version. The changes can be found in the release notes of the respective version.



2.2.4 Properties Results

Under the properties of the result representation three result diagrams can be selected, which can be accessed faster.

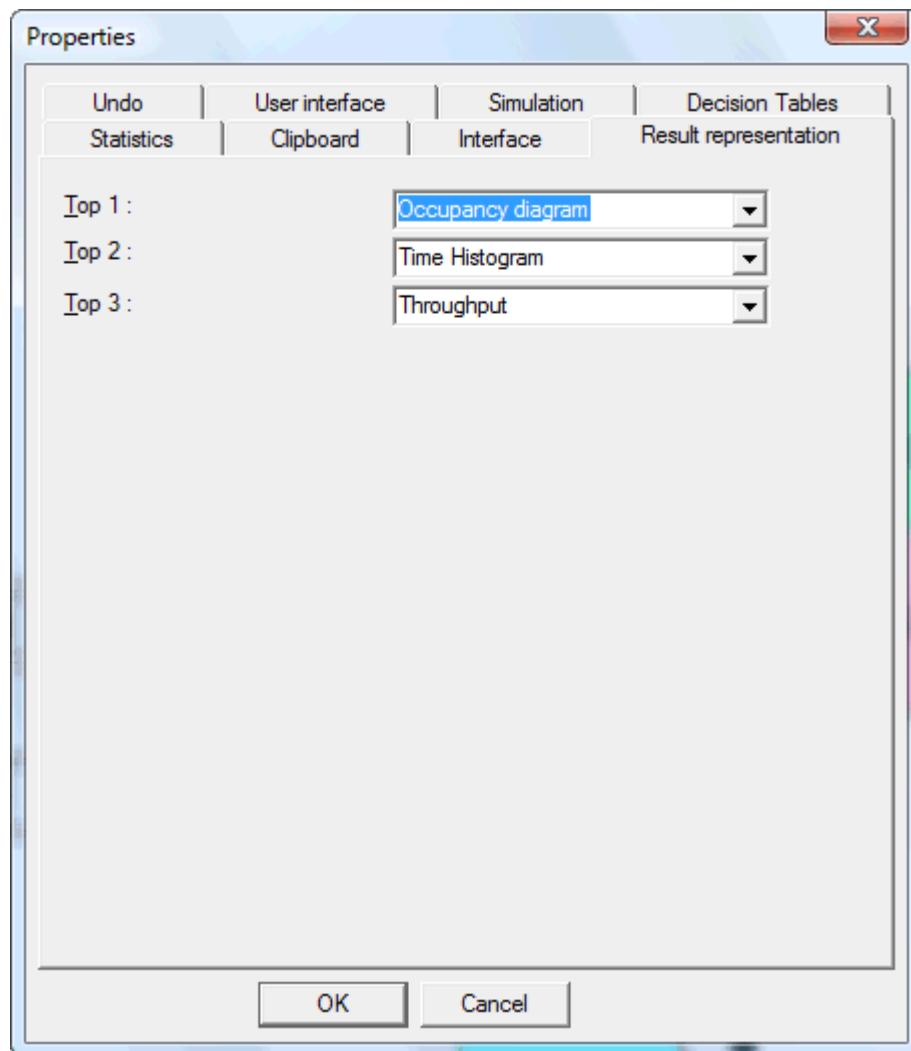


Figure 2.34: Properties result representation

These can be found in the upper area of the menu result. Additionally these entries are available for fast access in the context menu of each Dosimis-3 element.



2.2.5 Properties Clipboard

The export of data causes problems, because different symbols are used as separators depending on setting of country. To configure this, the symbols for **field separators** or **decimal separator** can be set.

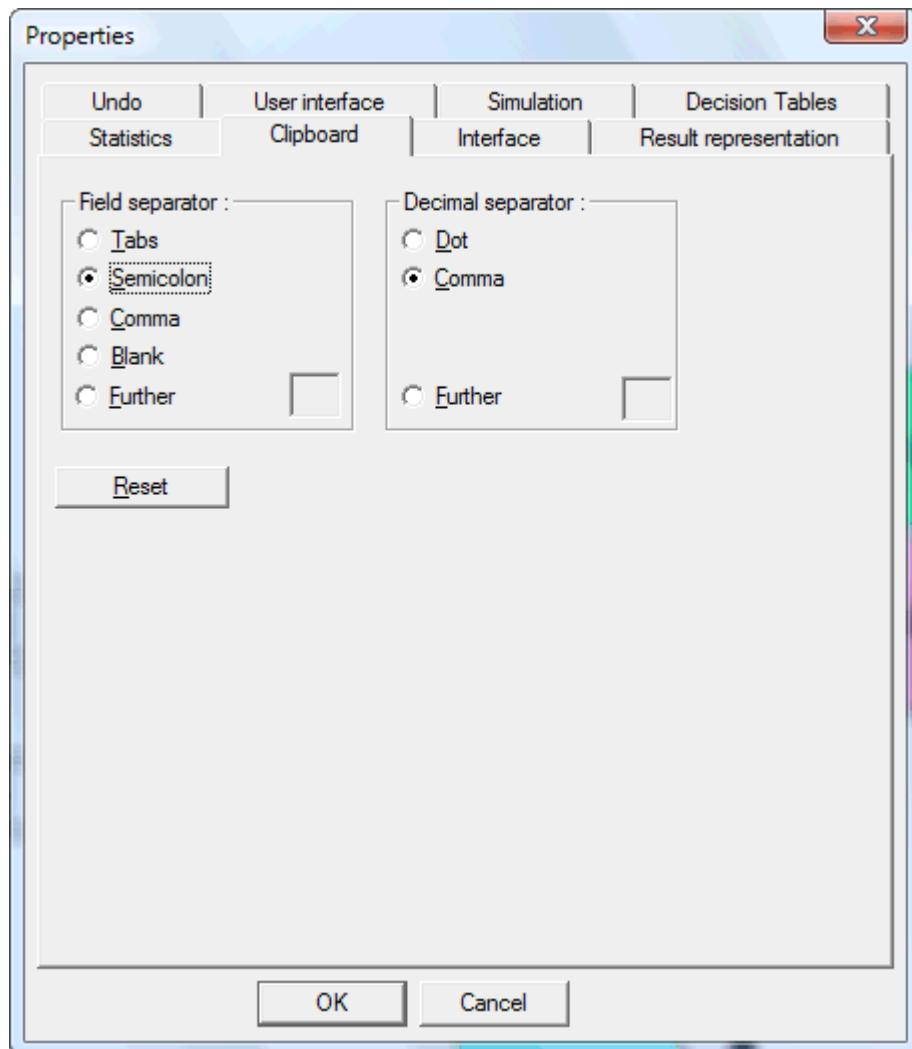


Figure 2.35: Properties of Clipboard

With the button **Reset** the symbols of the current country setting are used to initialize the symbols.



2.2.6 Properties Decision Table

The properties of the decision tables refer on the one hand to the representation of the editor. Normally the position and the partitioning of the sub windows are individual for each decision table and used with each call. This leads however with changes within different decision tables to the fact that they "jump". This can be deactivated by the switch **Unique representation of all decision tables**.

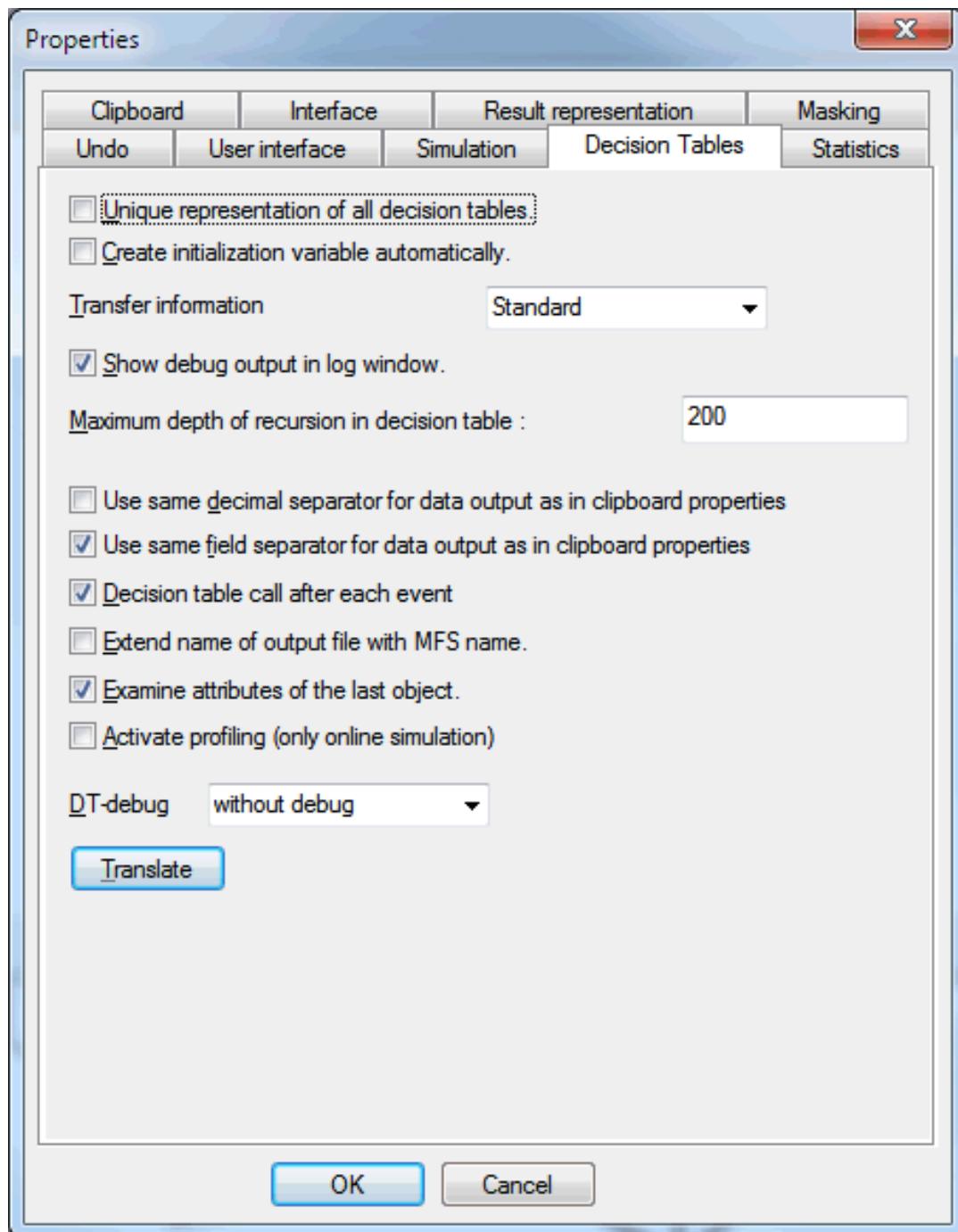


Figure 2.36: Properties of decision tables

If with the definition of an initialization, an assignment to a new initialization variable is made, this variable can be defined implicitly if the switch **Create initialization variable**



automatically is activated. The third state means, that before creating a variable automatically a security request will be made.

The **Maximum depth of recursion** in decision tables can be limited with this parameter. Thus, unintentional loops can be detected without a crash of the simulation program.

During the output to a file the separators are taken from the system configuration. When different separators are defined for the output to clipboard, these can be used for the output to a file by the check box **Use same separator for data output as in clipboard properties**.

After each event of an activated module of a decision table the table will be evaluated. To disable this and for the analysis only of the relevant events like entering, exiting and so on, this can be disabled by the switch **Decision table call after each event**.

Activate profiling offers the possibility of examining time consumption more exactly during the computation. Calls of decision tables as well as the use of the Quicktable are statistically seized. You can take further information from the chapter [Online – Simulation](#) out of the documentation of the decision tables.



2.2.7 Properties Statistic

Statistics are created for each model. From this you can examine, which Dosimis-3 version created the model and the name of the original model, from which this data was derived. Additional information regarding user interface and model size are available.

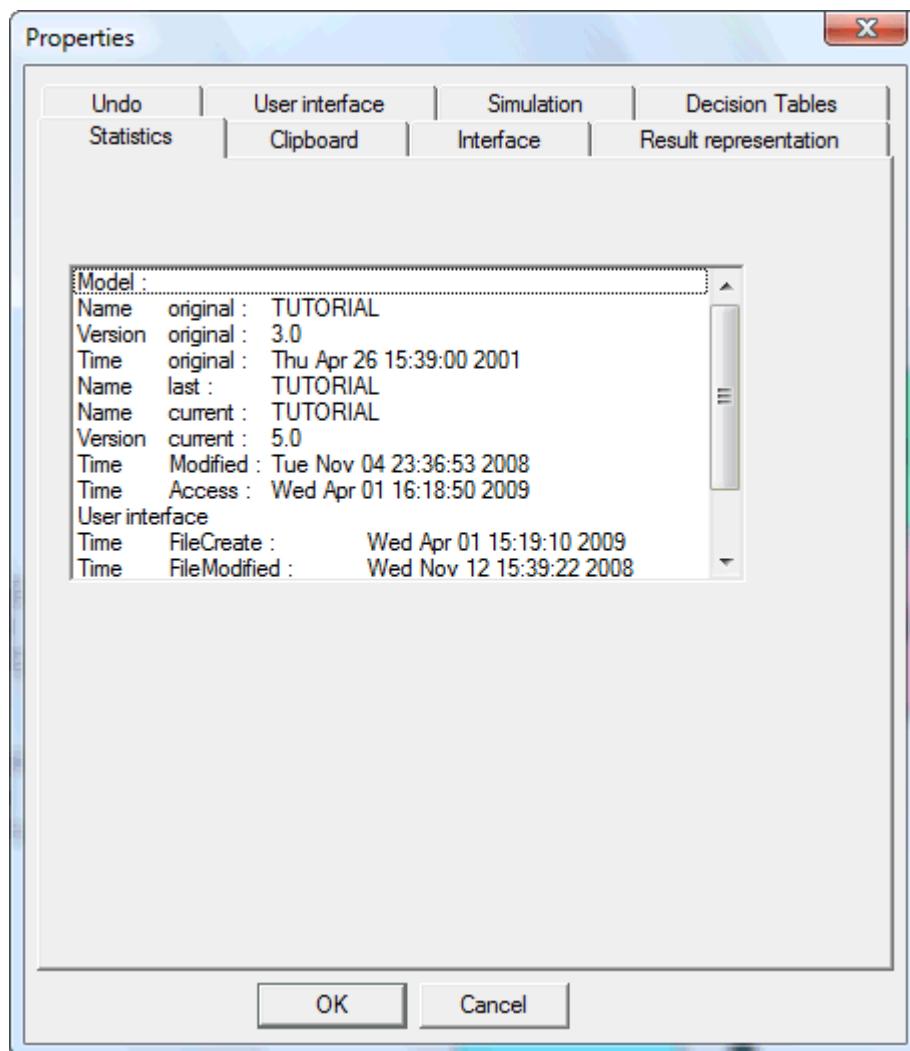


Figure 2.37: Properties of statistics



2.3 Menu Edit

After selection of an existing element, the menu **Edit** allows the following commands.

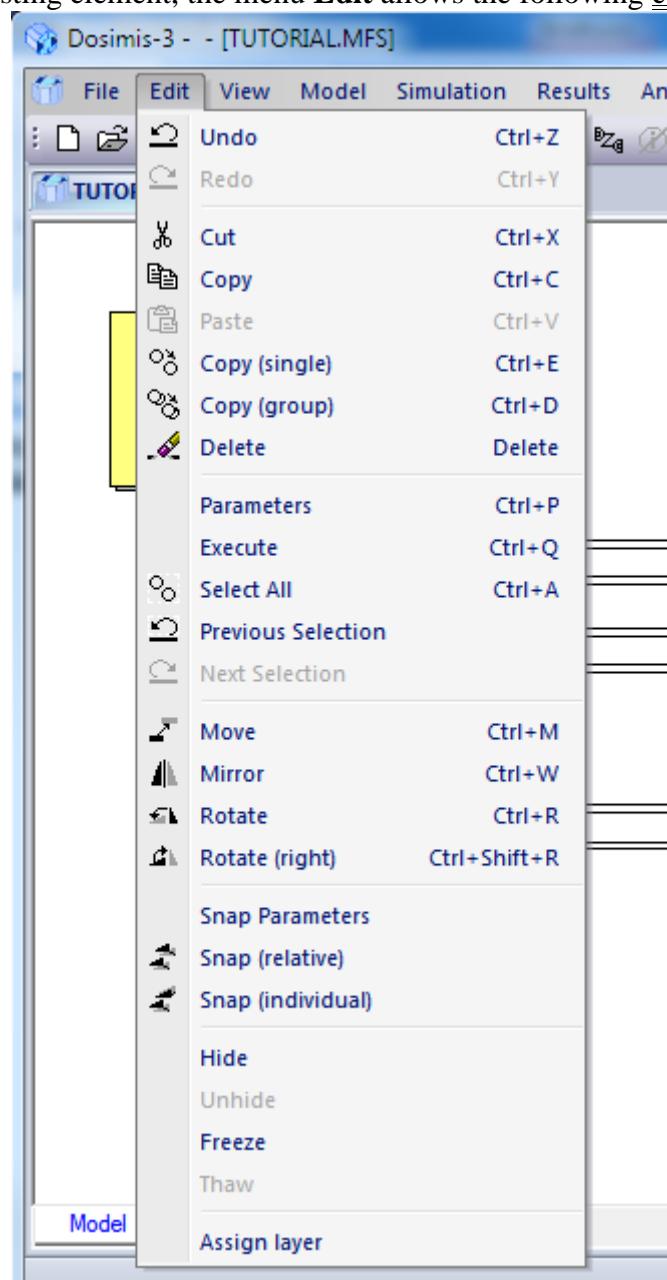


Figure 2.38: Editing of elements

Most commands can be accessed by the toolbar **Edit**.



Figure 2.39: Toolbar „Edit“



2.3.1 Undo

The last operation can be undone.

Shortcut

Toolbar:



Keyboard:

CTRL+Z

With each operation that modifies the characteristics of the model, an image of the model is saved. This occurs within the temporary area of the computer. Since this can particularly affect time noticeably with large models, the possibility exists of reducing the frequency by excluding particular operations. This occurs by means of menu option *File/Properties*.

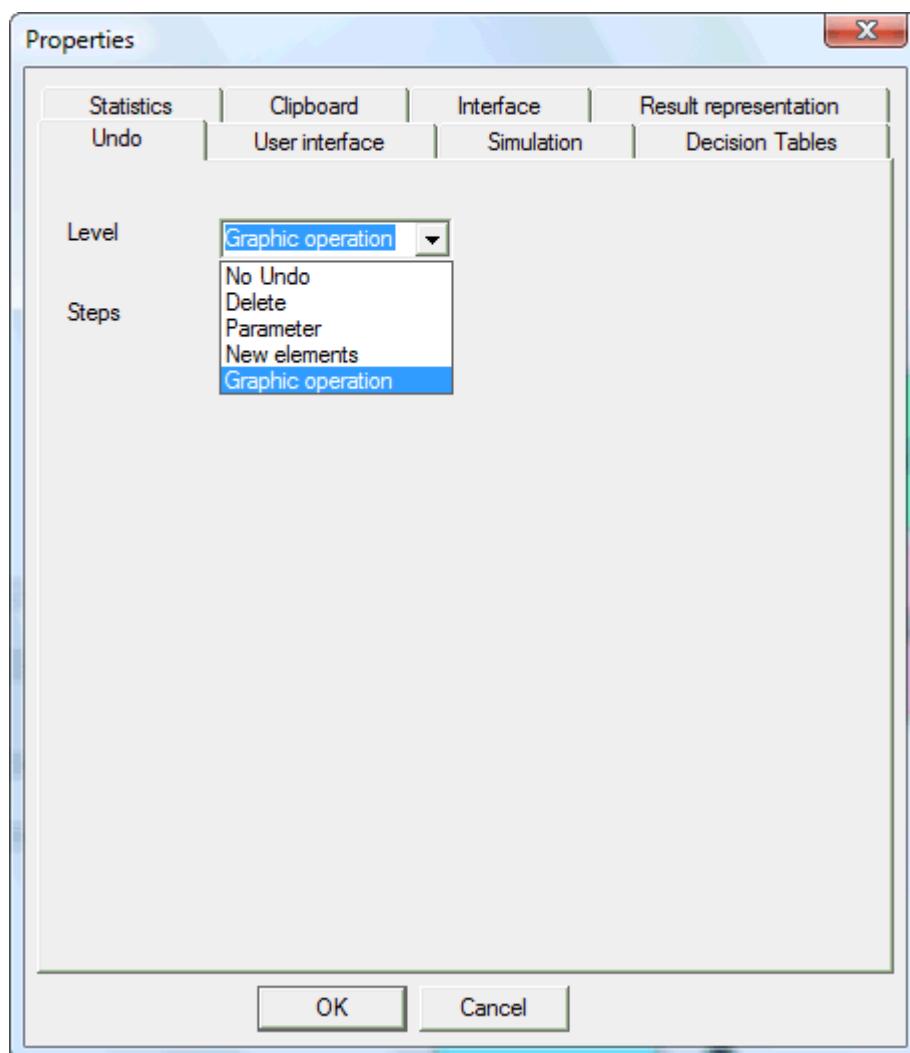


Figure 2.40: Scaling of the Undo function

According to standard, *graphic operations* are adjusted, so that after each relevant operation a security step is executed. If one selects in the protection level for example *Parameter*, then the graphic operations and placing of new items into the layout does not occur, i.e., before these operations no protection takes place. Accordingly the further levels are to be selected, so that in extreme cases with *No Undo* this functionality is switched off.



In the field *Steps*, the number of operations that can be canceled are defined. This reduces the space requirement of the hard disk.

2.3.2 Redo

The last undo operation can be undone. This operation is available till the next change on the model.

Shortcut

Toolbar:



Keyboard:

CTRL+Y

2.3.3 Cut (into Clipboard)

The selected item is deleted and copied into the clipboard. Further components of model are also considered. Further information is under *copying (into the clipboard)*.

Shortcut

Toolbar:



Keyboard:

CTRL+X

2.3.4 Copy (into Clipboard)

The selected items are copied into the clipboard. Additionally further components are added, which do not have a graphic representation in the model layout. Therefore the following rule applies. References to global decision-table data (network attributes, global decision tables...) are also copied if one of the selected items directly or indirectly refers to these. Same rule applies for model variable (Integer variable, Float variable...).

Shortcut

Toolbar:



Keyboard:

CTRL+C



2.3.5 Paste (from Clipboard)

If items were copied into the clipboard, these can be inserted into the model. Since combining partial models with decision tables (network attributes, element variables) can cause conflicts, a query to rename these conflicts appears in addition to inserting from the clipboard. According to the occurrence of those names a type is provided. (-) meant that this name is used in the original model. Entries of the type (+) are names from the inserted model, which are unique. Ambiguous names of the inserted model are provided with an exclamation mark (!). Modifications of the last two types mean that the references are modified for these names in the decision tables. In the case of an empty input the appropriate variable is removed from the new model. In case of name conflicts the variable from the original model is referenced.

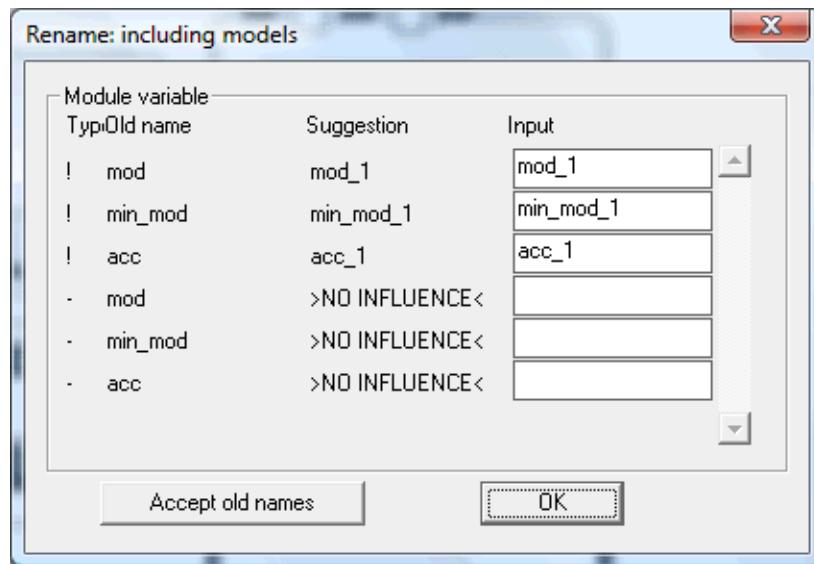


Figure 2.41: Solving conflicts during the combination of models

The button **Accept old names** makes sure that all names which are used in both sub model appear only once in the combined model.

Shortcut

Toolbar:



Keyboard:

CTRL+V

2.3.6 Copy (Single)

The **Copy (single)** function can be used to copy an element including the parameters. The orientation and the size of the new element can be changed. All references are put back. If references are to be copied along, the instruction *copy (group)* should be used.

Shortcut

Toolbar:



Keyboard:

CTRL+E



2.3.7 Copy (Group)

The **Copy (group)** function copies a group of selected elements including the parameters. It is not possible to change the orientation and the size of the elements. The decomposition (solution) of references (e.g. component lists of failures) occurs according to the following rule: If the reference within the new group can be dissolved, this is done. If however this is not possible, the reference is searched for in the other part of the model.

Example: If the user copies a failure by itself, all components of the old failure are also in the component list of the new failure. If however the components are in the group to be copied, their copies are in the component list of the new failure. The references to the old components are no more available after the restore operation.

Shortcut

Toolbar:



Keyboard:

CTRL+D

Beware of the different behavior of direct copying and copying via clipboard. Some information can be lost there.

2.3.8 Delete

With the **Delete** function the chosen element (module, junction, work area ...) can be removed. This command can only be selected if modules are selected.

Shortcut

Toolbar:



Keyboard:

DEL

To prevent the user from unintentional deletions, a safety check appears.

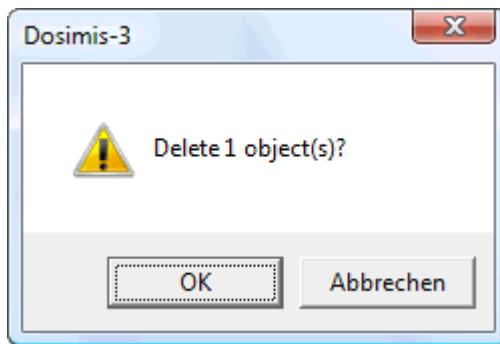


Figure 2.42: Safety check for deletion



2.3.9 Parameter

The button **Parameter** opens a window, to enter parameters for the selected module (elements, breaks...). If several modules are selected all corresponding parameter windows are opened one after another. Alternatively, you can double-click a module.

Shortcut

Toolbar:	no symbol
Keyboard:	CTRL+P or Double-click

2.3.10 Execute

Use this command to execute the standard method of the current element. Not every element possesses a standard method. This command is available only by single selection and when the element supports this function. Alternative a double-click with pressed SHIFT key can be made.

Shortcut

Toolbar:	no symbol
Keyboard:	CTRL+Q or SHIFT + Double-click

Example for elements with standard methods:

Global control	Directly starts the dialog of the decision table.
Evaluation	Shows the result of the evaluation.
Quick table	Opens the quick table and shows the data, when an initializing file is assigned.

2.3.11 Select all

You can select all elements of the layout with this command.

Shortcut

Toolbar:	
Keyboard:	CTRL+A

2.3.12 Previous Selection

With this instruction the items of the layout are selected, which were in the last selection.

Shortcut

Toolbar:	
Keyboard:	no shortcut



2.3.13 Next Selection

With this instruction a retrogressively made selection is again removed.

Shortcut

Toolbar:



Keyboard:

no shortcut

2.3.14 Move

The **Move** function can be used to move an already placed element to another place in the view area. For this, start and end point is to be selected with the mouse. For displacement of elements, it is sufficient to select an element with the left mouse button and to replace the objects with pressed left mouse button.

All elements are moved relative to the snap values. The consideration of this grid can be switched off by pressing of the CTRL-Key.

Shortcut

Toolbar:



Keyboard:

CTRL +M

Alternatively selected elements can be moved by the 4 arrow keys.

2.3.15 Mirror

With this command, you can reflect the marked data. You determine the reflector axis for this with the mouse. The final position of the reflected elements can be tested by pressing the SHIFT key. The mirror line can be displaced by pressing the CTRL -key.

Shortcut

Toolbar:



Keyboard:

CTRL +W

2.3.16 Rotate

With this command, you can rotate the marked data. You determine the pivot around which rotation is performed by 90 degrees clockwise for this with the mouse. By pressing the SHIFT key, you can test the final position of the rotated elements.

Shortcut

Toolbar:



Keyboard:

CTRL +R



2.3.17 Rotate Clockwise

With this command you can turn the marked data in the clockwise direction. For this you determine the center point with the mouse, around which turning is performed by 90 degrees in the clockwise direction. By pressing of the SHIFT key you can check the final position of the rotated items.

Shortcut

Toolbar:



Keyboard:

SHIFT+CTRL + R

2.3.18 Snap Parameters

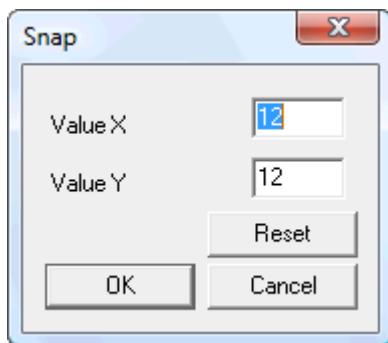


Figure 2.43: Snap parameters

With the aid of the registered snap parameters, the work area is partitioned into pixels. New elements are aligned with their insertion points on these points. Already placed elements can be brought subsequently to this grid with the command **Edit/Snap (individual)**. If the change of the parameters is temporary, the previous values can be restored by pressing the button **Reset**.

2.3.19 Snap (Individual)

With this command, you can align the selected elements to the current grid. Each element is moved individually, so this can lead to different displacing length.

Shortcut

Toolbar:



Keyboard:

no shortcut

2.3.20 Snap (Relative)

With this command, you can align the selected elements to the current grid. The first element is displaced in accordance with the grid and all further ones are displaced around the same value.

Shortcut

Toolbar:



Keyboard:

no shortcut



2.3.21 Hide

With this instruction can the selected items will be hidden. The items are not any longer drawn, but they can be selected however.

Shortcut

Toolbar:	no symbol
Keyboard:	no shortcut

2.3.22 Unhide

With this instruction hidden items are made visible.

Shortcut

Toolbar:	no symbol
Keyboard:	no shortcut

2.3.23 Freeze

With this instruction selected items can be frozen. These elements remain visible but they cannot be selected.

Shortcut

Toolbar:	no symbol
Keyboard:	no shortcut

2.3.24 Thaw

With this instruction all frozen items are released again for selection.

Shortcut

Toolbar:	no symbol
Keyboard:	no shortcut

2.3.25 Assign Layer

With this instruction layer can be assigned to the selected elements. Information accord layer are described in section toolbars TOP_MODELLIEREN_SYMBOLLEISTE.

Shortcut

Toolbar:	no symbol
Keyboard:	no shortcut

Elements can be assigned to a layer in two different ways. If elements are selected, the selection dialog for layers appears. Here the layer can be assigned to several elements at the same time.

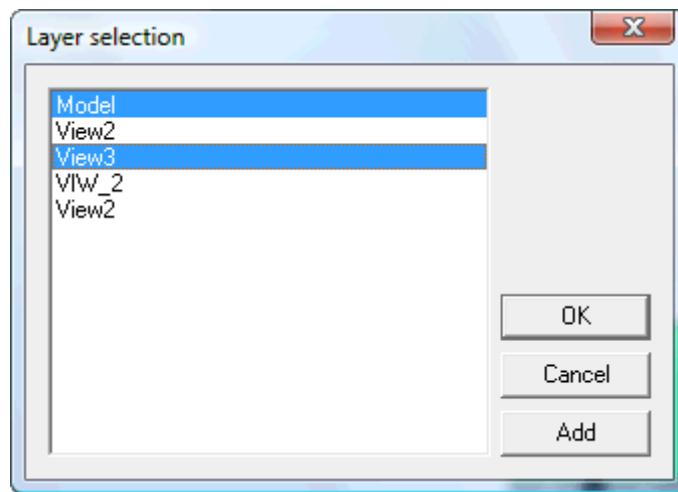


Figure 2.44: Layer selection

If the dialogue is left by pressing **Add**, the selected layers are added additionally to the already existing layer of the selected elements. In the other case the existing layers are removed first. The layers can however also be individually managed in each element. For this each element possesses a sub dialog, in which this is made possible.

2.3.26 Characteristics of selected or unselected Elements

To select/deselect elements or group of elements use the left mouse button on the element. *Shift* and *Control* are modifiers with the following meaning:

Click on	Shift	Ctrl	Action
Selected element	No	No	No effect
	Yes	no	Element is not selected
	Yes	Yes	No effect
	Yes	Yes	All elements under the cursor are selected
Unselected element	No	No	Element is selected
	Yes	No	Element is additionally selected
	No	Yes	Element is additionally selected
No element	Yes	Yes	All elements under the cursor are additionally selected
	No	No	No element is selected
	Otherwise	Otherwise	No effect

DOSIMIS-3-specific information:

- The layout of the module symbols orients itself to a grid with 24 points in X/Y direction.
- The anchor points of the modules should be adjusted to this grid. This can be achieved with parameter of adjustment.
- The expansion of the modules, which extended over several grid squares (e.g. shuttle, accumulation conveyor...), is also adjusted to the value 24.
- The co-ordinates are displayed in the status bar. The menu option **View/Zoom/Cords** is switched on. This information is updated with each mouse movement and thereby prevents the display of other messages. Therefore it should be activated only to control the co-ordinates.



2.4 Menu View

By working on a graphical model of a material flow system sometimes it is necessary to change the view of the model. The menu item **View** serves for it. The menu **View** displays the following submenus:

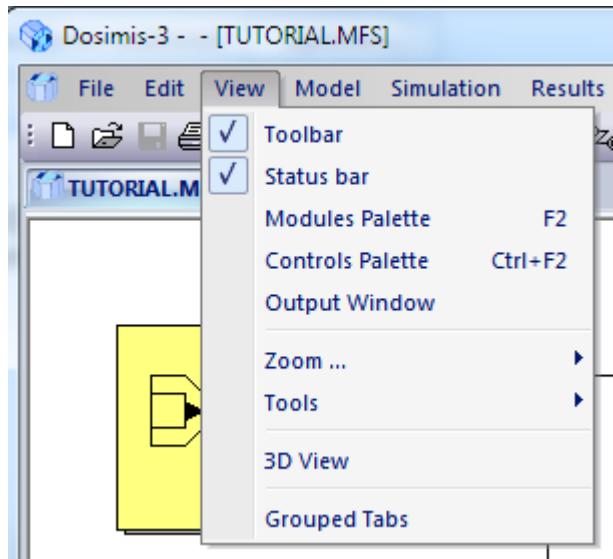


Figure 2.45: Submenus „View“

The submenu **Toolbar** displays or closes the toolbar described above. It is left to the user to use the toolbar and submenus via the menu bar or through the keyboard (consult MS-WINDOWS literature).

The submenu **Status bar** displays or closes the status bar. To get messages about modules or about the status of the simulator it is advisable to keep the status bar active.

The submenu **Modules palette** opens or closes the modules palette (see Figure 2.17).

The submenu **Control palette** opens or closes the control palette. The control palette contains elements, which do not transport objects. Mostly there are modules with superior controls (e.g. work area, failures, decision tables). Additionally these are modules which make the work more comfortable (evaluation or throughput-time measurement).

Zoom... allows a larger or reduced view of the material flow system. By selecting this submenu the menus shown in the next figure appear.

Tools enables the user to show or hide the displayed toolbars. See also chapter [Toolbars](#).

3D View leads to a three-dimensional view of the layout. See chapter [3D-Visualization](#).



2.4.1 Menu Zoom...

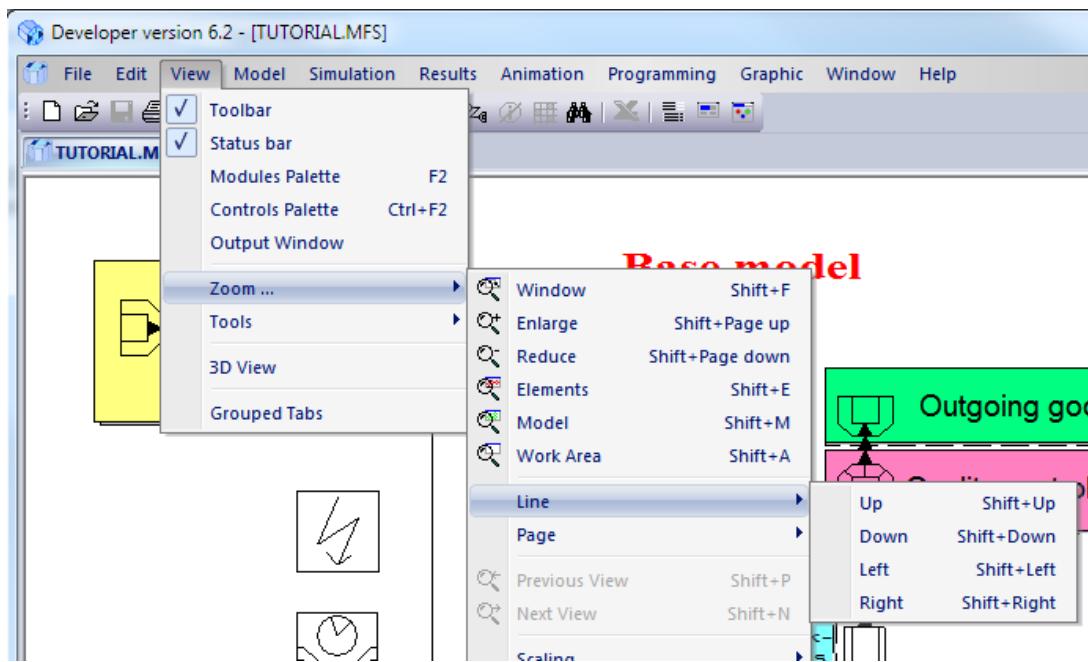


Figure 2.46: Submenu Zoom

Navigating within the model takes place with the help of the Zoom menu. All instructions are available however also as shortcuts and/or in a toolbar. Furthermore the visible pane can be moved with pressed mouse wheel within the work area.

2.4.1.1 Window

Use this instruction to magnify any rectangular section of the model.

Shortcut

Toolbar:	
Keyboard:	Shift + F

The submenu **Window** enables the user to have a closer look at an area of the drawing. The message *Select a rectangle area* first appears in the status bar. Now the user moves the cursor to a corner of the area to be zoomed. Then the user moves the mouse with pressed left button to the opposite corner. The message *Ready* appears in the control bar. Now the area inside the rectangle is zoomed.

2.4.1.2 Enlarge

Enlarges the center of the visible area.

Shortcut

Toolbar:	
Keyboard:	Shift + PgUp

Alternatively the mouse wheel can be used.



2.4.1.3 *Reduce*

Shows the model on a reduced scale.

Shortcut

Toolbar:



Keyboard:

Shift + PgDn

Alternatively the mouse wheel can be used.

2.4.1.4 *Elements*

Shows elements that are visible in the current view.

Shortcut

Toolbar:



Keyboard:

Shift + E

2.4.1.5 *Model*

The entire layout is displayed in a window.

Shortcut

Toolbar:



Keyboard:

Shift + M

The command **Model** means that the whole MFS is loaded on the screen, thus making optimal use of the screen area.

2.4.1.6 *Work Area*

Shows the complete work area.

Shortcut

Toolbar:



Keyboard:

Shift + A

By using **Work Area**, the whole area that can be modeled is displayed. This might be useful to give an overview or to select new areas for zooming.

If the user has already zoomed a part of the whole view and now wants look at another part of the MFS in the same scale he can use the option **Line** and **Page**. Using these two options the model scrolls up or down, to the right or to the left. These operations can also be done by using the scrollbars to the right or bottom of the work area.



2.4.1.7 Line

Shifts the layout by a small unit of distance in the selected direction.

Shortcut

Line up	Shift + Page-Up
Line down	Shift + Page-Down
Line left	Shift + Page-Left
Line right	Shift + Page-Right

Similar to this shifting (line by line) it is possible to use the appropriate arrow keys on the scrollbars.

2.4.1.8 Page

Shifts the layout by a larger unit of distance in the selected direction.

Shortcut

Page up	Ctrl + Page-Up
Page down	Ctrl + Page-Down
Page left	Ctrl + Page-Left
Page right	Ctrl + Page-Right

Even here it is possible to shift the layout by clicking between slider and arrow key on the scrollbar in the right direction.

2.4.1.9 Previous View

Draws the area after a modification with the last adjustment.

Shortcut

Toolbar:	
Keyboard:	no shortcut

2.4.1.10 Next View

Undoes the last zoom instruction.

Shortcut

Toolbar:	
Keyboard:	no shortcut



2.4.1.11 Scaling

Shows the work area in a predefined scaling. Therefore screenshots of the work area are always equally large, which is otherwise only guaranteed, if the work area is maximized.

Shortcut

600x480	Alt + Numpad 1
800x600	Alt + Numpad 2
1024x768	Alt + Numpad 3
1152x864	Alt + Numpad 4
1280x960	Alt + Numpad 5

2.4.1.12 Coordinates

Shows the coordinates of the cursor in the status bar. This information is updated with each mouse movement and thereby prevents the display of other messages. Therefore it should be activated only to control the co-ordinates.

Shortcut

Keyboard: no shortcut

2.4.1.13 Redraw

Redraws the visible area.

Shortcut

Keyboard: F5



2.5 Menu Model

The menu **Model** contains the following submenus:

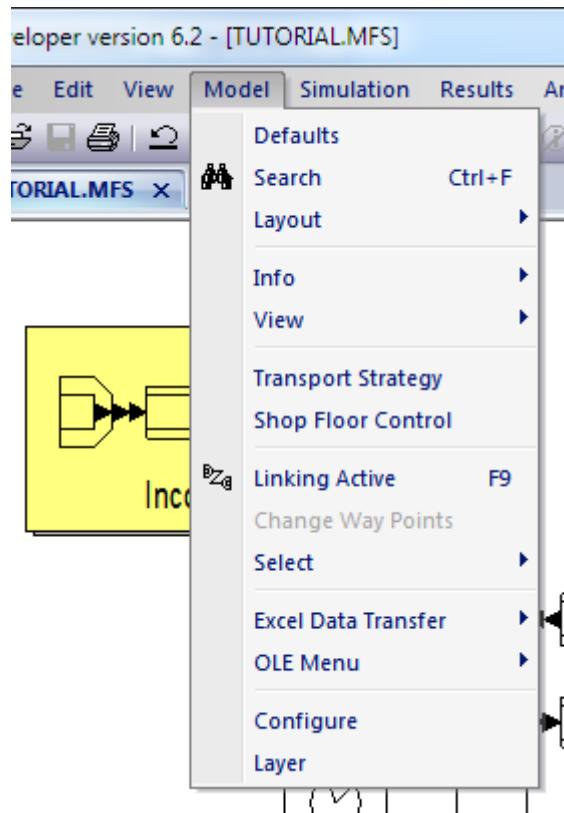


Figure 2.47: Submenu „Model“

The submenu **Defaults** enables an efficient parameterization of some components. (See also Chapter 3: Material Flow). The *conveying speed* and the *object length* of the element can be pre-set with this submenu, as also the settings of *forward control*.

Search makes it possible to quickly identify a special element. In addition, faults, global controls (GC), capacity monitoring, work areas, animation text, junction and decision tables can be searched.

Layout makes it possible to carry out layout operations on the entire model.

Info shows element parameters in the layout.

View turns different groups of DOSIMIS-3 elements on or off.

Global DT-Data opens a window to set parameters of decision tables. (See: User manual for setting-up decision tables)

Shortcut

Symbol:
Toolbar:



Modeling

Transport Strategy to define the strategy for guided vehicles. (go [there](#))



Working plans leads to the dialog for parameterizing of work plans and orders (go [there](#)).

Linking active enables the junction of elements (see section: Linking of modules) and/or the end of this mode. In this state, components are also assigned to super ordinate elements (failures, operating ranges).

Shortcut

Symbol:
Toolbar:



Modeling

Way points: Modify the track points of the connecting junctions. If more than 2 junctions are selected and if the **SHIFT key** is pressed, then all track points of the selected junctions are deleted. This works even when different items are selected. If exactly one element is selected, which can possess more than 2 track points (such as accumulation conveyor), the menu entry **Replace element** appears. This element can again be repositioned in the layout. The parameterization and the connections are preserved.

Select enables selecting by means of submenus from predefined lists and elements according to types.

The **Excel Data Transfer** enables the exchange of parameter via Excel tables. This is dealt with in a later chapter.

OLE Menu lets you use and exchange data from or with different applications.

Configure offers the possibility of adapting the standard behavior of the user interface.

With **Layer** the drawing layers are managed. If no element is selected, here new layers can be created.

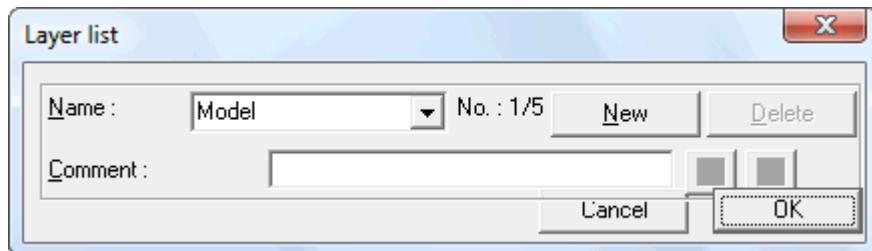


Figure 2.48: Parameters of layer

Each layer possesses a name and can be described more precisely with a comment. Furthermore these can be moved within the list with the help of the arrow keys.

A view possesses a list of layers, which are active in this view. Likewise an object (module, control, graphic) possesses a list of layers, which are assigned to this object. An object is drawn in a view if at least one layer of the object agrees with one layer the view.

2.5.1 Search

Through the menu **Search** it is possible to search for items of the model by number or name. The found items will be selected. If there is no item of the indicated number, an error message is displayed. In case of decision tables the component is selected, whose main decision table



corresponds to the search word. The components of the upper section can be looked up both for number and for name. For items within the lower area only one of these two search criteria is applicable.

In addition to the numbers and names it is possible to search for strings in the initializations, conditions or actions of decision tables. To do this, select **Main decision tables** under *Looking for* and **Parameter (contains)** under *Search for*. You can then search each decision table with the search string by pressing the button **Parameter**.

Note: Global and Transport decision tables cannot be looked up here.

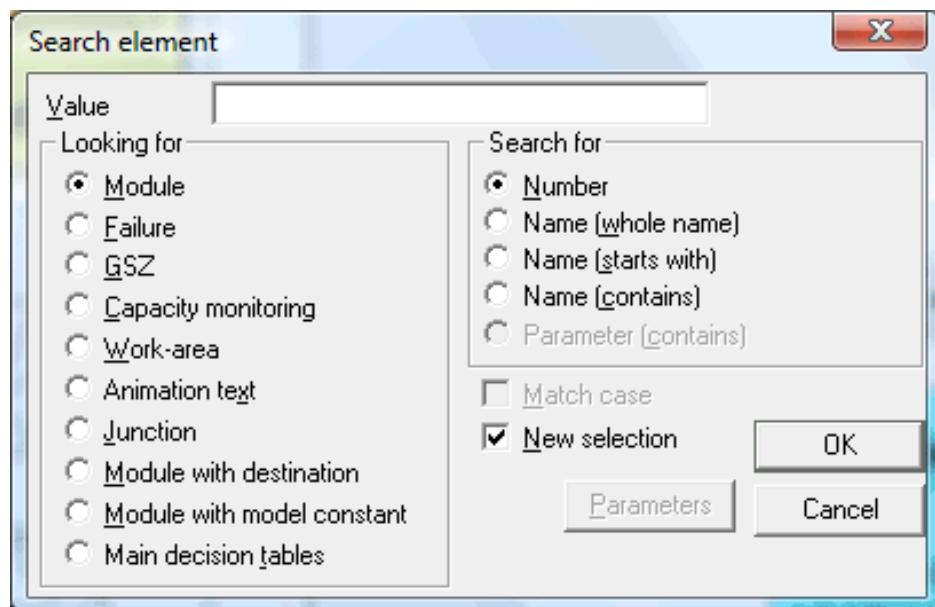


Figure 2.49: Options during „Search“

There is the possibility of looking for the number and the name. The option also exists of selecting all items that begin with or contain the search text. If **New selection** is activated, then all found items become the current selection, otherwise they are added to the current selection.

Looking up elements can be activated also via the toolbar.

Shortcut

Symbol:



Toolbar:

Modeling

Keyboard:

Ctrl + F

With the key **F3** the last search can be repeated.



2.5.2 Submenu Layout

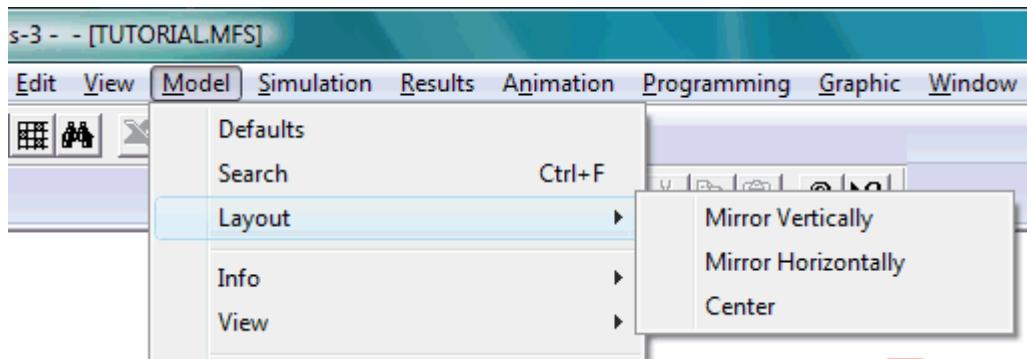


Figure 2.50: Submenu „Layout“

The submenus **Mirror Vertically** and **Mirror Horizontally** flip the MFS up and down or right and left in the screen. The submenu **Center** centers the model in the work area.

2.5.3 Submenu Info

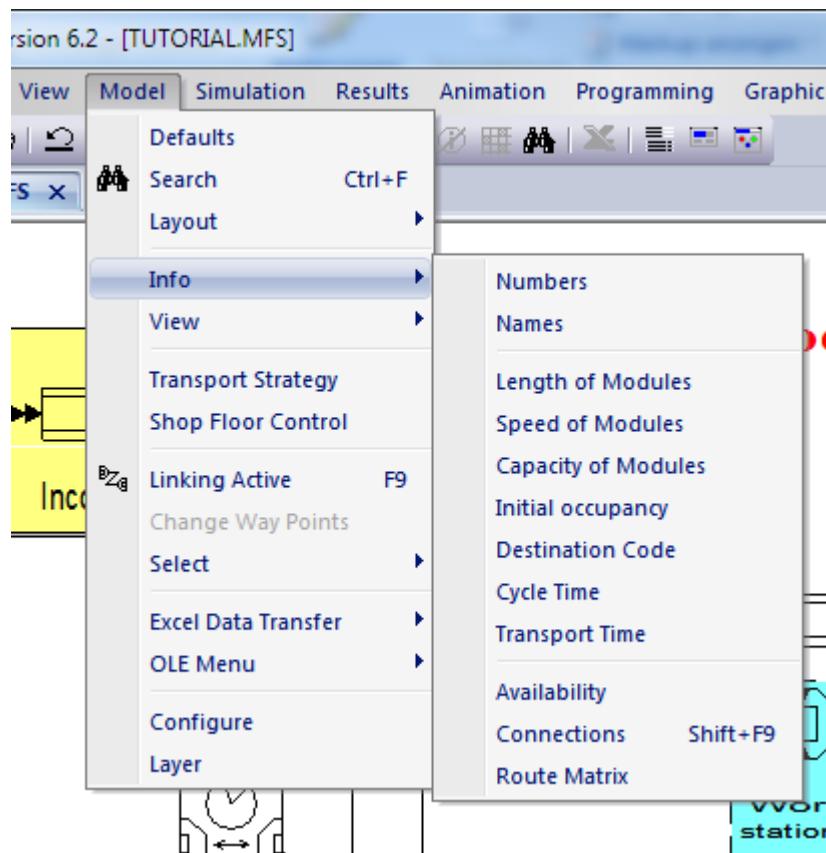


Figure 2.51: Submenu „Info“

Info... offers the submenus shown below. They give different information about the elements of the MFS; this information is shown in the element. The submenus are self-explanatory. **Destination code** is the number of the targets in the transport disposition. Initial allocation indicates the number of objects initialized in the components.



Initial Occupancy shows the number of the initialized objects of a module.

The **Cycle time** indicates the duration between the entry and/or exit of two objects in this module. Apart from the relationship of module length (segment length) to speed in work stations the mean operating time is also used. In the case of a forward control the time to enter the successive module is also considered.

The **Transport time** is the time, which is needed to move through the module. For components with several entrances and exits, the average value for all relations is used. The entry time into the successive module is not considered.

Additionally the option exists to insert the **Availability** into the failures if the parameter permits this (periodic and random failures).

With the submenu **Connections**, the connections between controls and elements are shown in the layout.

Shortcut

Keyboard: SHIFT+F9 or SHIFT + right mouse button

Connections from decision tables to activated elements are in red and to reference elements are in blue.

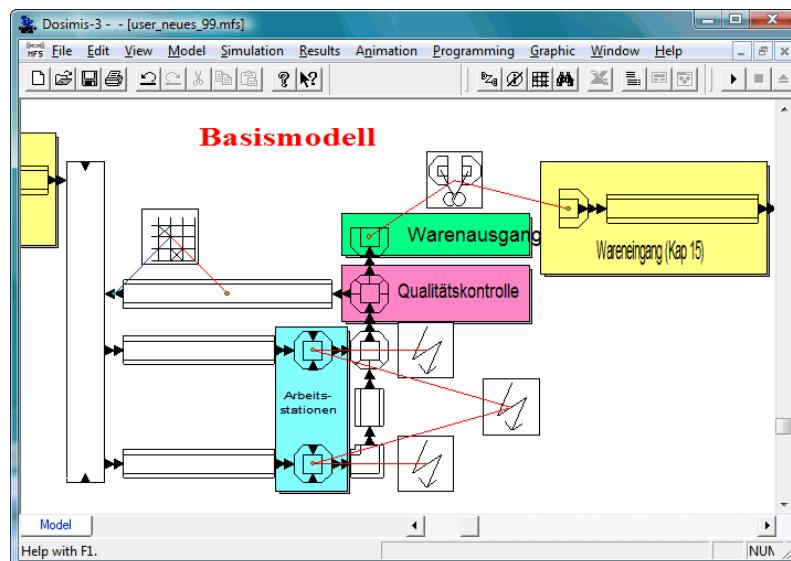


Figure 2.52: Connections „without selection“

If no element is selected, all relations are shown.

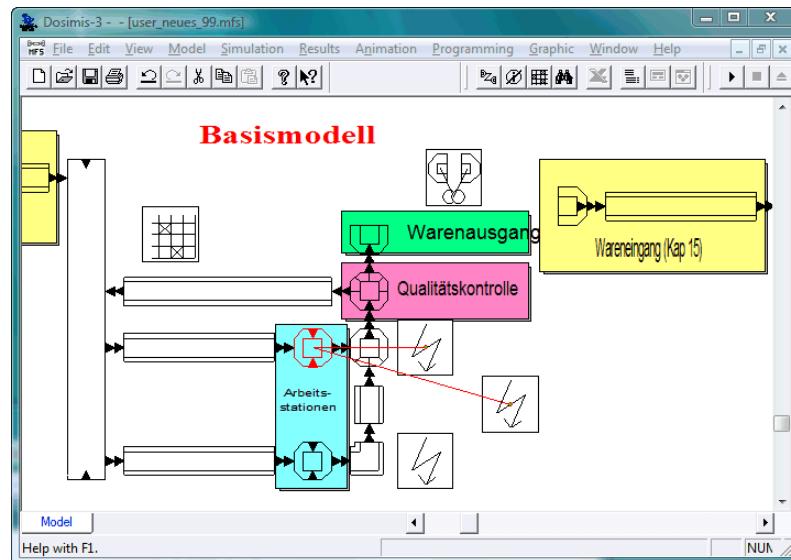


Figure 2.53: Connections „with selection“

If an item is selected, only the connections associated with this item are shown. Elements, which are used in a global decision table, are marked with an open green connection.

With the command **Track matrix**, the fastest way of the transport system will be displayed as calculated by the track matrix.

If a submenu from **Info...** is active in the model at the moment, then it can be hidden by a repeated selection of the appropriate submenu item. Alternatively a button is available in the toolbar.

Shortcut

Symbol:



Toolbar:

Modeling or Result

Keyboard:

no shortcut



2.5.4 Submenu View

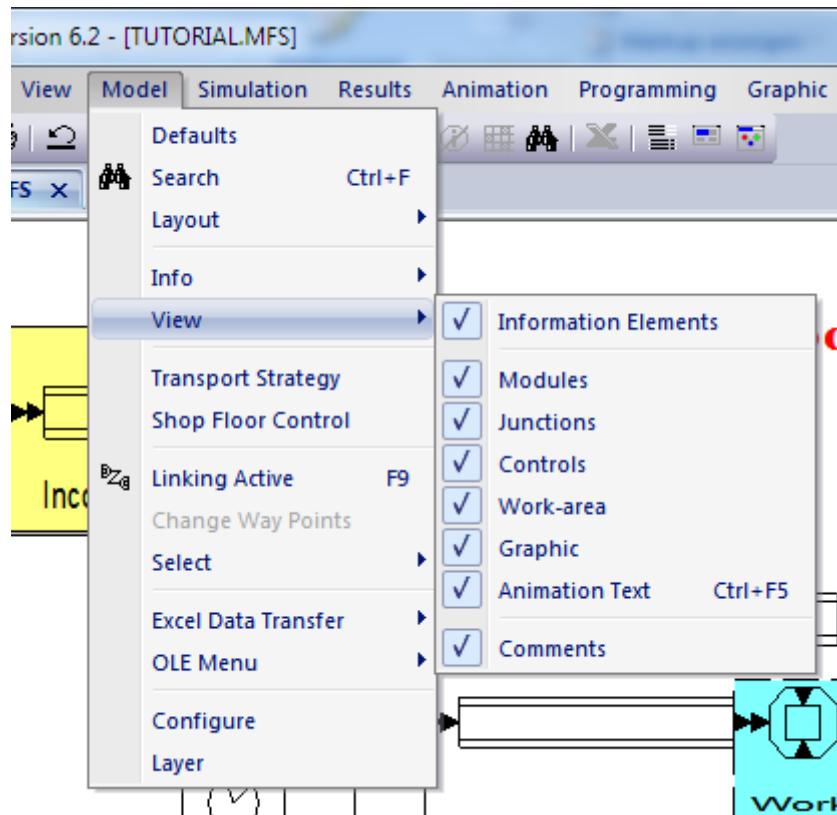


Figure 2.54: Submenu „View“

The entries below are like a switch and can be turned on and off. If the switch is turned on, the corresponding elements are shown in the work area, otherwise they are not displayed.

Information elements: Elements defined as information elements.

Elements: Modules of the MFS.

Junctions: Junctions of the MFS.

Controls: Control elements, e.g. GC, failure.

Work area: Show or hide all work areas including their working places

Graphic: Graphic elements created via elements palette.

Animation text: Animation text is automatically turned off at start of animation. You can turn it on for purposes of modification. This happens automatically at start of animation. With this menu option these can be shown again, so that these appear again for handling.

Comments: Hide graphics created by the graphic palette. This corresponds to the menu item *Hide all graphics* from the graphics menu.



2.5.5 Submenu Select

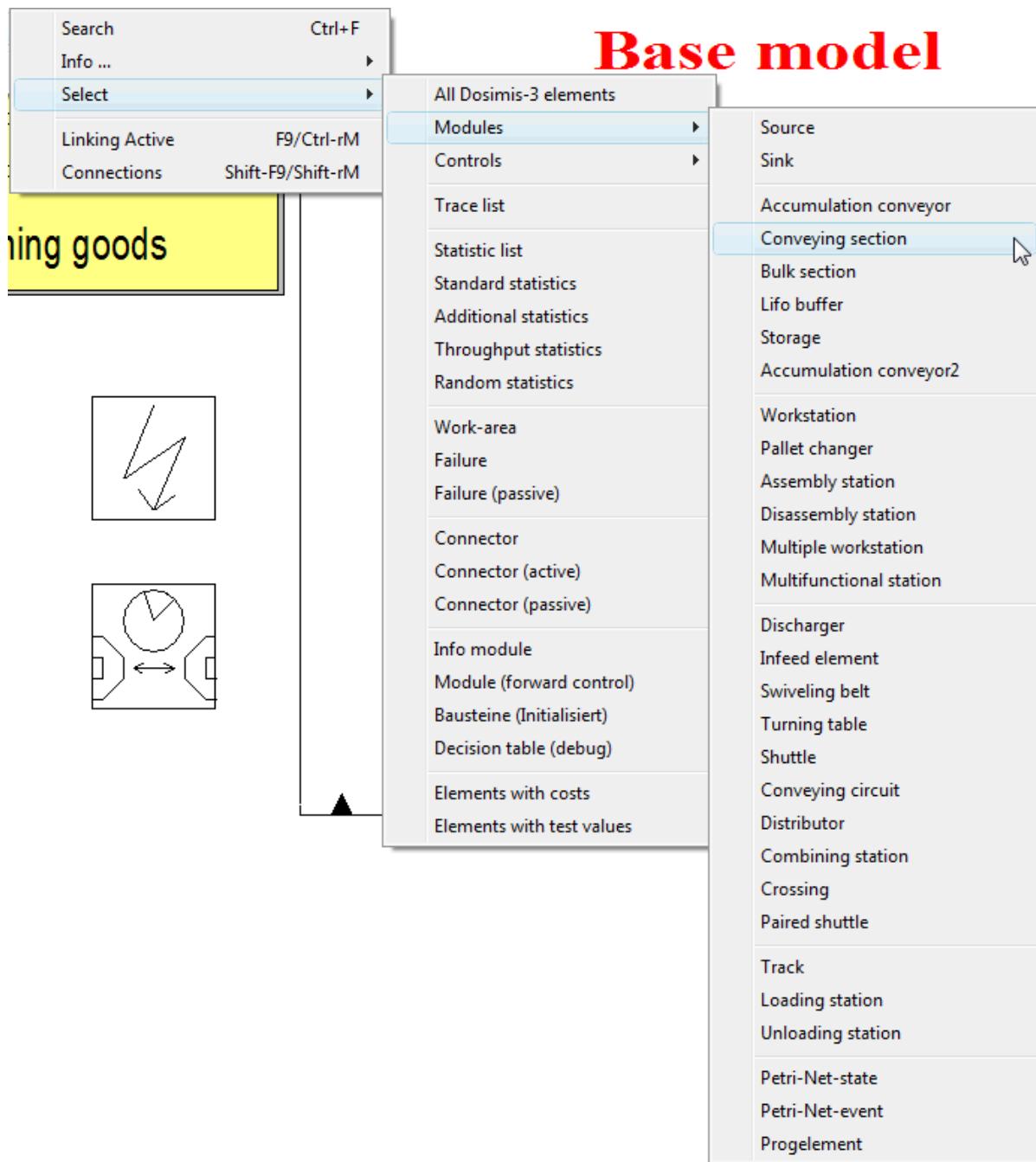


Figure 2.55: Submenu „Select“

Additionally controls and components can be selected according to type and characteristics (parameters). The selection takes place if the characteristic shown in brackets fits.

With the menu **Select** the items that correspond to the selection criteria are selected. The selection is always additive in nature, i.e. the found items are added to the current selection. Thus selections can be arranged, on which the operations are executed (e.g. common parameterizing or definition of the trace list (see menu option Simulation)).



This can be **All DOSIMIS-3 Elements** (elements and controls). This can also be one of the three standard lists (Trace, **Statistics** list and **Additional statistics list**). The controls and elements can be selected according to type and characteristics. The selection takes place if the characteristic indicated in parentheses applies.

2.5.6 Submenu OLE Menu

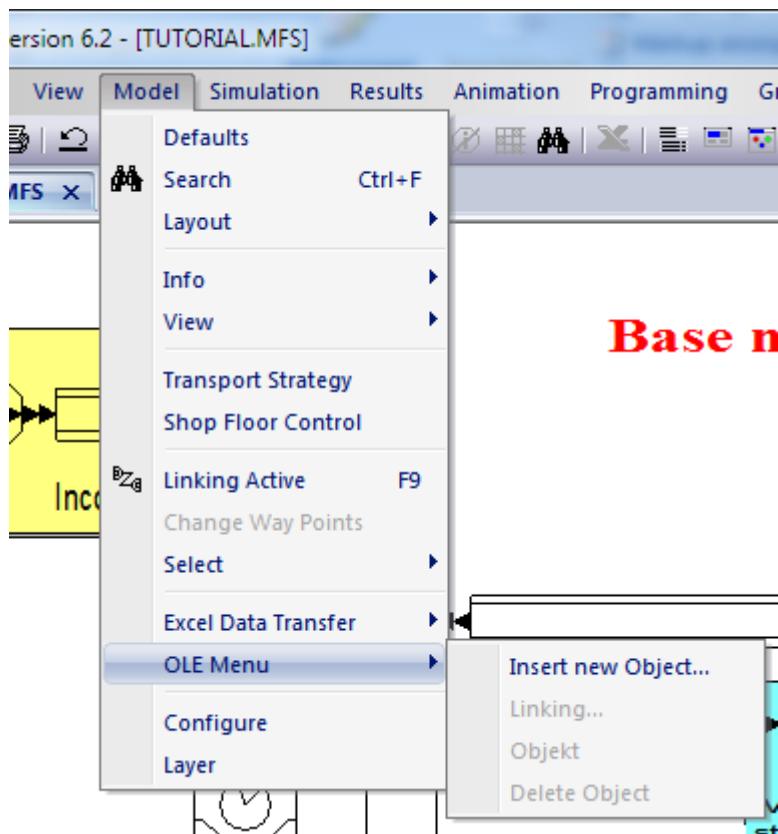


Figure 2.56: Submenu „OLE-Menu“

OLE is the abbreviation of **Object Linking** and **Embedding**. This means the linking and embedding of objects, i.e., you can use and exchange data from or with different applications.

With the menu option **Insert new Object** you can generate a new OLE object or insert it from a file.

With the menu option **Linking** you can edit a linked OLE-Object.

With the menu option **Object** you can activate a linked or embedded OLE-Object.

With the menu option **Delete object** you can delete a selected OLE-Object.

All these operations are temporary and therefore not saved with the model.

All objects inserted in such a way are only temporary and are not available at a later stage. They are only meant for the formatting of the layout such as for presentations. If diagrams should be used in the layout, this can be done by bitmaps. These are available in the graphics palette.



2.6 Menu Simulation

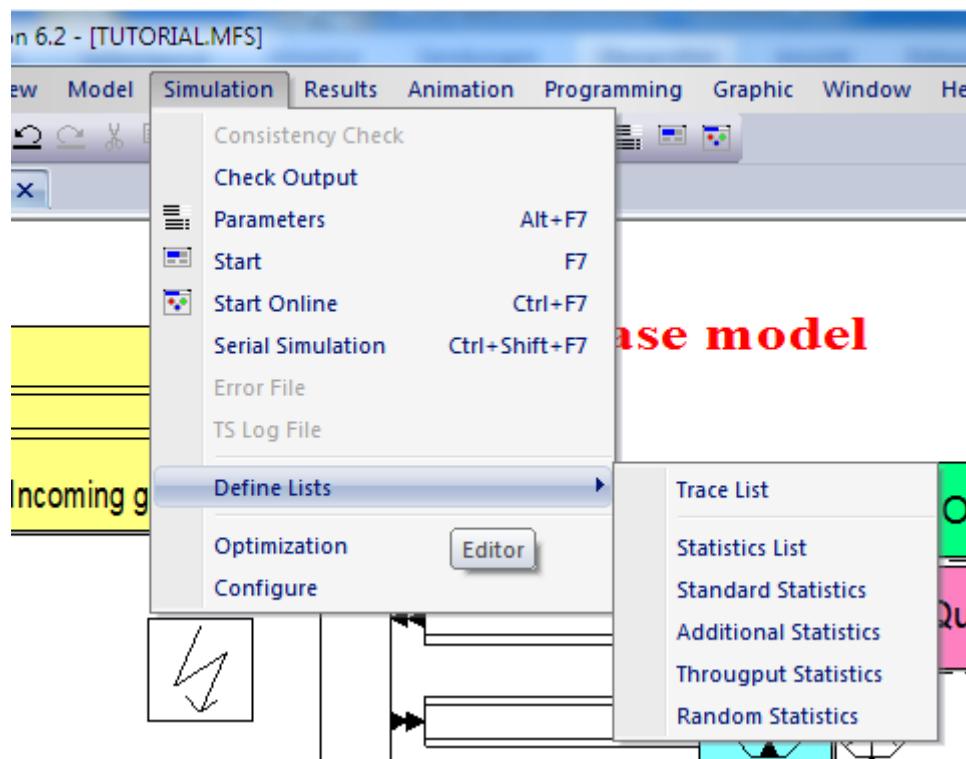


Figure 2.57: Submenus „Simulation“

Detailed information can be found in the chapter [Simulation run](#).



2.7 Menu Results

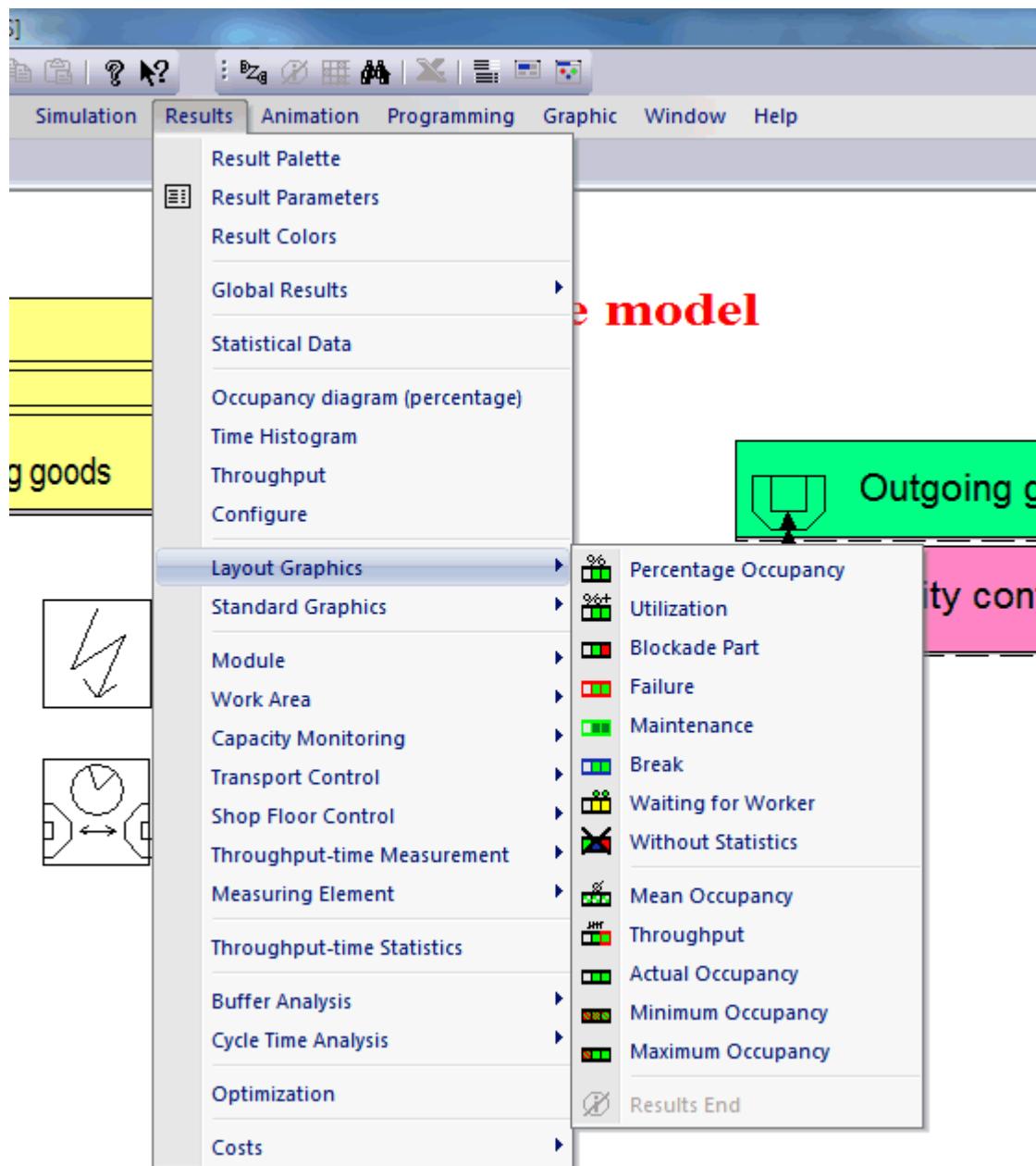


Figure 2.58: Menu „Results“

Under the submenu **Result parameters** you find a small dialog to affect the selection of data (statistic moment, interval statistic). Detailed information can be found in the chapter [Simulation results](#).



2.8 Menu Animation

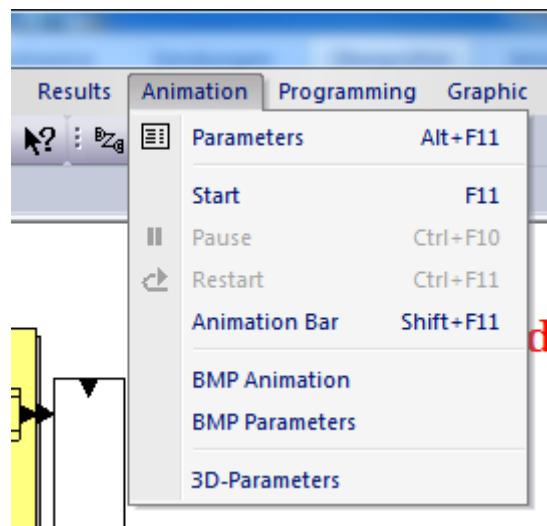


Figure 2.59: Menu „Animation“

The main menu **Animation** is used to start and modify the animation. A description can be found in the chapter [Animation](#) of this manual.

2.9 Menu Programming

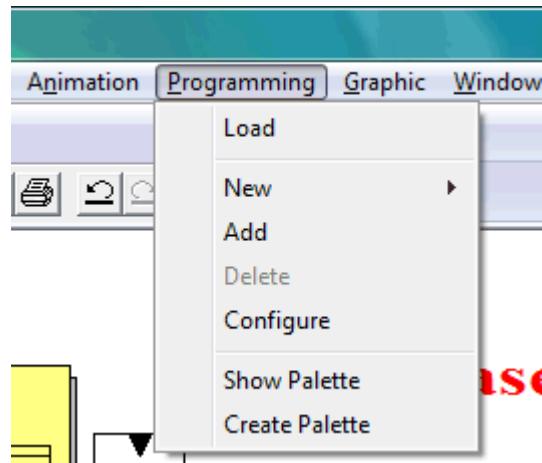


Figure 2.60: Menu „Programming“

A detailed description can be found in the chapter [Programming](#) of the manual.

2.10 Menu Graphics

The menu **Graphics** is used to process graphic comments. Here rectangles, lines etc. can be placed to structure the model. Detailed information can be found in the chapter [Graphic comments](#) of this user manual.



2.11 Menu Window

The menu **Window** arranges the opened windows according to the selected order. Here the possibility exists to open a second window of the model, for example, in which another window of the model is to be seen.

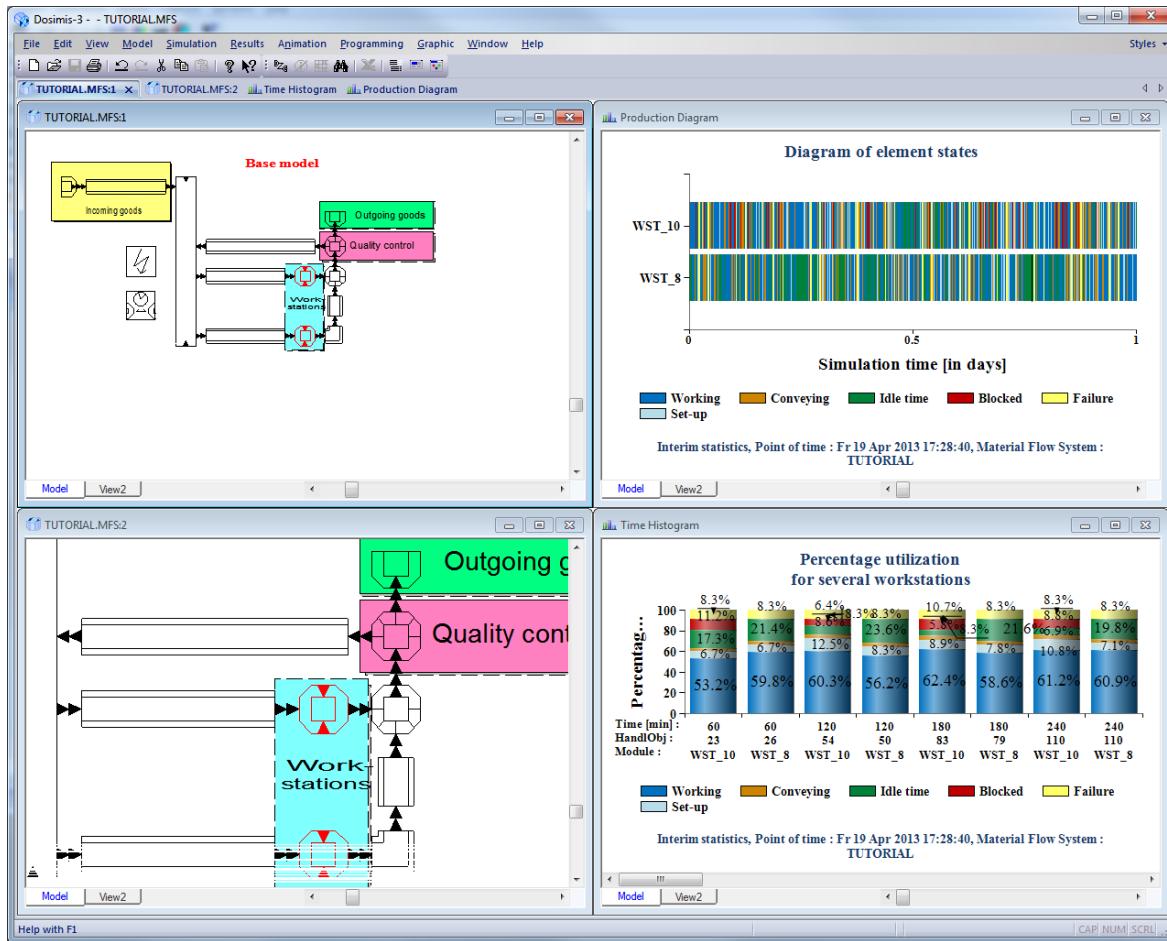


Figure 2.61: Windows in DOSIMIS-3

The handling is the same as the handling of other MS-Windows tools. Therefore these options are not described here.



2.12 Menu Help

The main menu **Help** contains the submenu **About DS3...**. The **About** dialog contains the version number of the current program as well as further information about DOSIMIS-3. The version number consists of the main number (in this case 6.2) and a build number (in this case 639).



Figure 2.62: Information about DOSIMIS-3

Using **Help** is as normally done. Additionally, whole sections can be printed from the Contents view via the option **Print**.

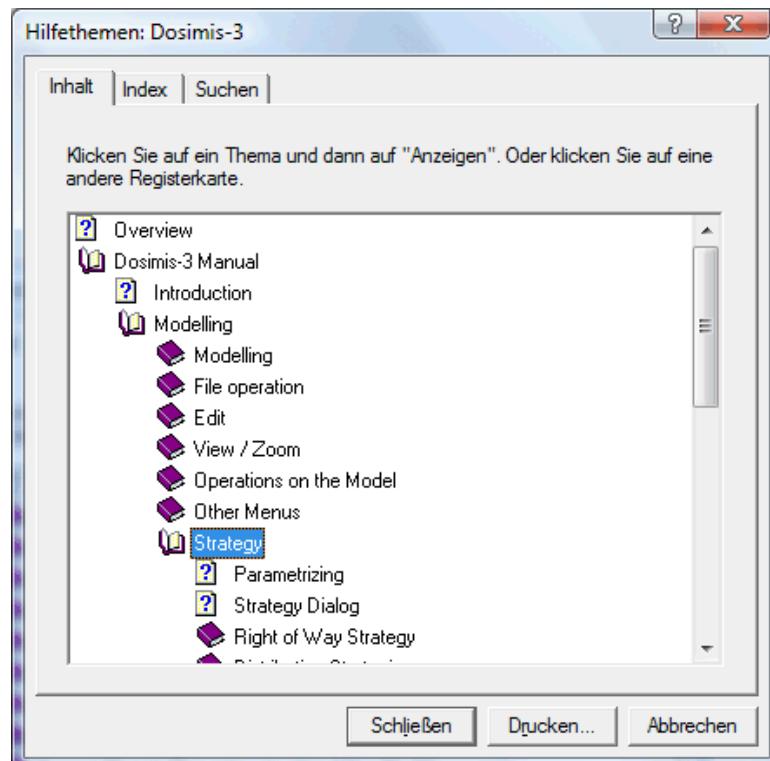


Figure 2.63: Information about DOSIMIS-3



You can jump forwards and backwards within a topic by using the two browser buttons



The Help window usually always appears in the foreground. This can be changed via the menu *Options/Always in the foreground.*



3 General Parameters of a Material Flow System

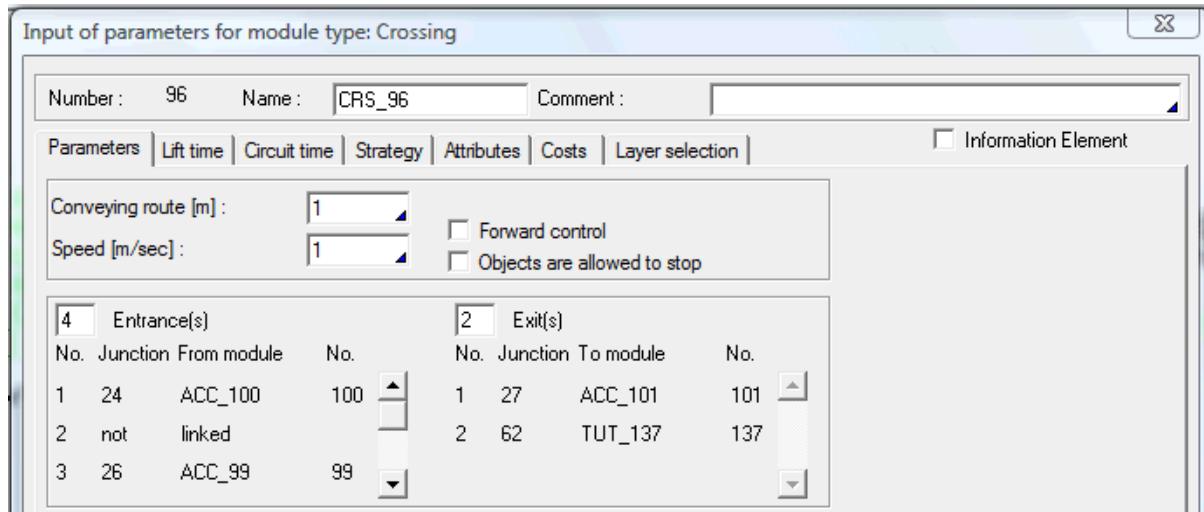


Figure 3.1: Input dialog of modules

In the upper area of the input dialog the name and a comment can be set. Additionally other parameters of the first sub-dialog have to be set. These are mainly speed und length of the module and simple strategies. There the links of the module with their neighbors are displayed. The number of entrances or exits can be changed in this dialog.

The behavior can be specified in further sub-dialogs depending on the module type. An input of these parameters is not necessary.

The following sub dialogs exist:

Strategy:	Right-of-way and Distribution strategy
Lift time:	Input of lift time (see <u>crossing</u>)
Circuit time:	Input of circuit time (see <u>crossing</u>)
Initialization:	Initialization of module (accumulation conveyor, conveying section, bulk section, LIFO buffer and track (transport system))
Double cycles:	Definition of double cycles of a paired shuttle
Strategy single cycle:	Right-of-way and distribution strategy of single cycles of a paired shuttle
Attributes:	Parameters of Attributes
Costs:	Parameters of Cost simulation
Break Points:	Parameters of break - points (Online simulation and animation).
Layer selection:	Parameters for structuring



3.1 Strategy Dialog

The right-of-way and distribution strategies of a module can be set in the sub-dialog **Strategy**. Only those strategies supported by the module are shown.

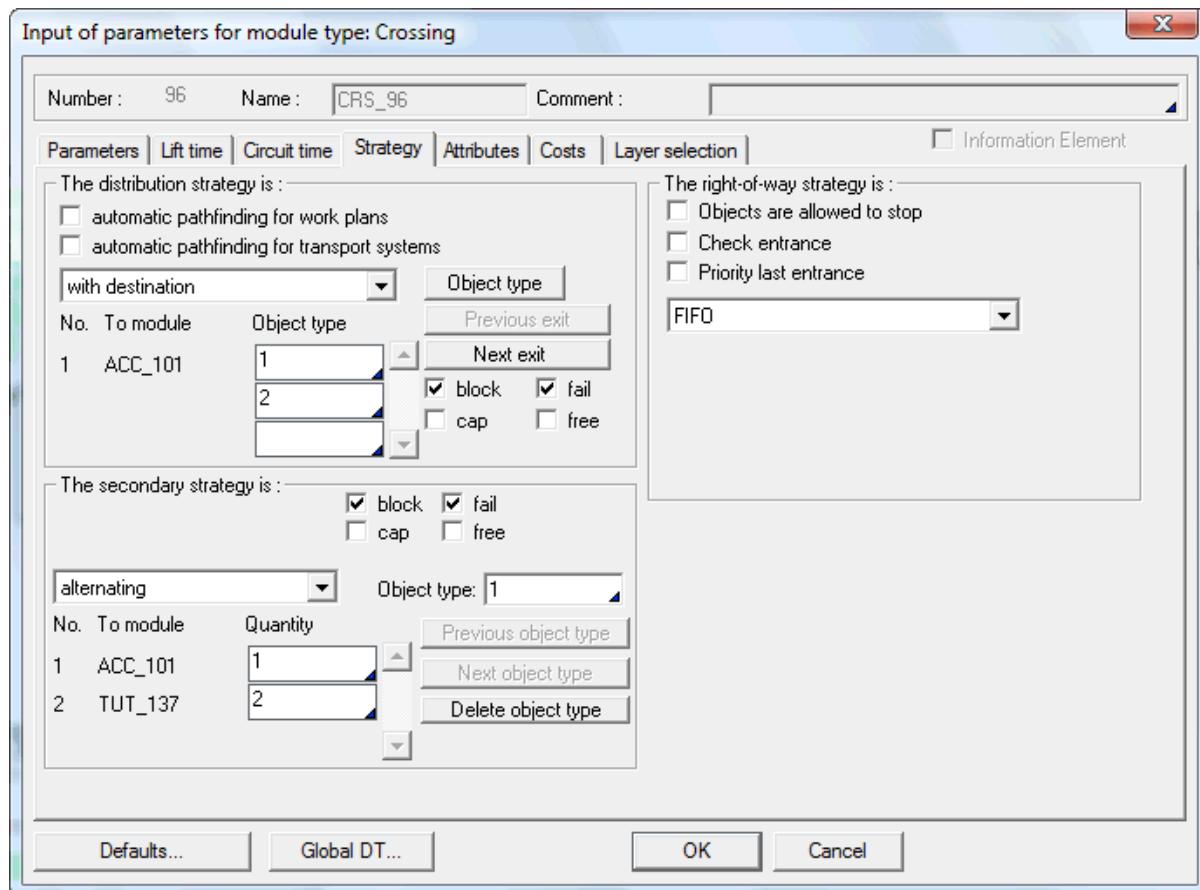


Figure 3.2: Input dialog of right-of-way and distribution strategies



The table shows the possible number of inputs/outputs of different elements. Depending on the number of inputs or outputs, right-of-way and/or distribution strategies can be assigned to some elements.

Module type	Number of entrances	Number of exits	Right-of-way strategy	Distribution strategy
Source	0	1	-	-
Sink	1	0	-	-
Accumulation Conveyor	1	1	-	-
Conveying Section	1	1	-	-
Bulk Section	1	1	-	-
Accumulation Conveyor2	1	1	-	-
LIFO-Buffer	1	1	-	-
Storage	Variable (≥ 1)	Same as entr.	-	-
Workstation	1	1	-	-
Assembly	Variable (≥ 2)	1	-	-
Multiple Workstation	1	1	-	-
Disassembly	1	Variable (≥ 2)	-	-
Multifunctional workstation :	Variable (≥ 1)	Variable (≥ 1)	-	-
Pallet changer	2	2	-	-
Crossing	Variable (≥ 1)	Variable (≥ 1)	Entrance ≥ 2	Exit ≥ 2
Turntable	Variable (≥ 1)	Variable (≥ 1)	Entrance ≥ 2	Exit ≥ 2
Shuttle	Variable (≥ 1)	Variable (≥ 1)	Entrance ≥ 2	Exit ≥ 2
Paired shuttle	Variable (≥ 1)	Variable (≥ 1)	Yes	Yes
Distributor	1	Variable (≥ 2)	-	Yes
Discharger	1	Variable (≥ 2)	-	Yes
Swiveling Belt	1	2	-	Yes
Conveying Circuit	Variable (≥ 1)	Variable (≥ 1)	-	Yes
Combining Station	Variable (≥ 2)	1	Yes	-
Infeed Element	Variable (≥ 2)	1	Yes	-
Petri Net State	Variable (≥ 1)	Variable (≥ 1)	-	-
Petri Net Event	Variable (≥ 1)	Variable (≥ 1)	-	-
Loading Station	2	1	-	-
Unloading Station	1	2	-	-
Track	1	1	-	-

Table 3.1: Elements: Number of entrances and exits



3.2 Right-of-way Strategies

A right-of-way strategy has to be defined if objects are waiting at more than one entrance. Only those entrances whose succeeding modules are not disturbed or waiting are considered. The link to the entrance must also not be closed.

The timing of the decision made in favor of one strategy is dependent on the state of the junction through which the last object leaves the module. If objects are waiting at several entrances, a decision is to be taken as soon as the link reaches the state **OCCUPIED**, i.e. when the object leaves the module. In case the module is free-space-controlled, the system delays the decision until the object has completely entered the successor. Thus, the link must retain the state **FREE**. A decision once taken is never changed.

If two objects with their entrances fulfill the same conditions for the right-of-way rule, the entrance with the lower entrance number is defined.

The user can choose among various right-of-way strategies. The following six strategies can be chosen for most of the modules with more than one entrance.

In general the following menu appears in the parameter dialog:

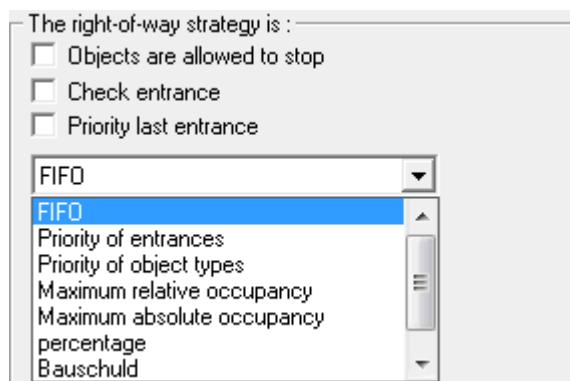


Figure 3.3: Right-of-way strategies

For modules with several entrances and exits the switch **Objects are allowed to stop** determines whether the object is to be taken over independently of the possibility of delivering it immediately. Otherwise the examination whether the object can be delivered (distribution strategy) takes place before overtaking of the object (see module bunch).

If **Check entrance** is selected, the strategy works after all the synchronous events of the predecessor elements are carried out.



If the check box **Priority last entrance** is selected, it is firstly examined, whether at the entrance, from which an object was taken over last time, a further object is waiting. Then by this entrance objects are taken over as long as no more objects are waiting there, the module at that entrance is disturbed or the junction is blocked.

see [FIFO](#)

see [Priority of entrances](#)

see [Priority of object types](#)

see [Maximum absolute occupancy](#)

see [Maximum relative occupancy](#)

see [Minimum free capacity](#)

see [Percentage](#)

see [Bauschuld](#)

see [Self-defined](#)

see [Shortest path](#)

The strategies function in the following way:

3.2.1 Right-of-way Strategy FIFO

The **FIFO-Strategy** checks the stand-by times of all objects waiting at the entrances. The object with the longest waiting time is allowed to enter the module next. (FIFO = First-In-First-Out has the same meaning as FCFS = First-Come-First-Served).

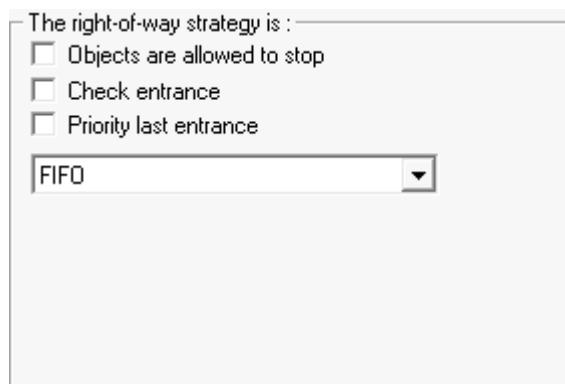


Figure 3.4: FIFO (default parameters)



3.2.2 Right-of-way Strategy Priority of Entrances

Priority **of entrances** assigns fixed and clear priorities to the entrances. A low number indicates a connection of higher priority. For smooth operation of the simulation, the entrances must have defined priorities. The object enters the module which is standing by at the entrance with the highest priority, i.e. at the entrance with the lowest *number* (thus 1 is the highest priority).

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

Priority of entrances

No.	From module	Priority
1	ACC_100	1
2	not linked	2
3	ACC_99	3

Figure 3.5: Priority of entrances

3.2.3 Right-of-way Strategy Priority of Object Types

Priority **of object types** assigns fixed and clear priorities to each object type that might pass through the module. The object with the highest priority is allowed to enter the module.

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

Priority of object types

Object type	Priority
1	1
2	2

Figure 3.6: Priority of object types



3.2.4 Right-of-way Strategy Maximum relative Occupancy

The strategy **Maximum relative occupancy of predecessors** defines the percentage occupancy of the preceding module by means of module capacity and current actual occupancy. The object standing by at the exit of the module with the highest percentage occupancy is allowed to enter the next module.

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

Maximum relative occupancy ▾

Figure 3.7: Maximum relative occupancy

3.2.5 Right-of-way Strategy Maximum absolute Occupancy

In contrast to the previous strategies the strategy **Maximum absolute occupancy of the predecessors** chooses the object of the module with the highest current actual occupancy. The capacity of modules is not important.

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

Maximum absolute occupancy ▾

Figure 3.8: Maximum absolute occupancy



3.2.6 Right-of-way Strategy Minimum free Capacity

Contrary to the previous strategy the strategy **minimum free capacity** of the predecessor module selects the object at the entrance, with which exhibits the few free capacity.

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

Minimum free capacity

Figure 3.9: Minimum free capacity

3.2.7 Right-of-way Strategy Percentage

With alternative entrances the **percentage** right-of-way strategy determines with the help of a random process the entrance, from which the next object is to be taken over.

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

percentage

No.	From module	Perc. Share
1	ACC_100	10
2	not linked	10
3	ACC_99	80

Figure 3.10: Percentage



3.2.8 Right-of-way Strategy Bauschuld

In contrast to the previous strategy the strategy **Bauschuld** selects the object according to a fixed algorithm.

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

Bauschuld

No.	From module	Perc. Share
1	ACC_100	10
2	not linked	10
3	ACC_99	80

Figure 3.11: Bauschuld

3.2.9 Right-of-way Strategy Self-defined

The strategy **Self-defined** allows the user to introduce any right-of-way strategy desired. The user can either resort to decision tables (see User manual for setting-up decision tables)

The right-of-way strategy is :

Objects are allowed to stop
 Check entrance
 Priority last entrance

Self-defined

[Edit right-of-way DT](#)

[Delete DT](#)

Figure 3.12: Self-defined



3.2.10 Right-of-way Strategy Shortest Path

For infeed modules, the strategy **Priority of transit section** is offered. Here an object is only allowed to enter the main conveying direction if the transit conveyance is not hindered by an object. Therefore, during entrance, it is checked whether an object is arriving from the predecessor module in the main conveying direction. If this happens, the entrance is delayed, otherwise slipping-in is allowed. Only predecessor modules are considered. Modules further back are not considered so that with short conveying distances and high conveying speed, it is possible for an object to arrive at the main conveying entrance before the entering process is finished.

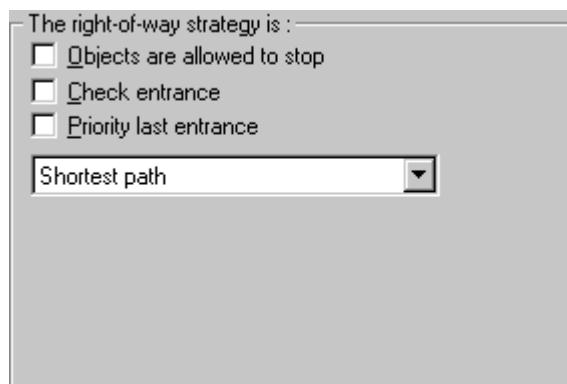


Figure 3.13: Shortest path

For turntables an additional strategy is offered, which takes into account the position of the table. Here the object at the entrance in the most advantageous position is selected. This means the **shortest turning time** is given priority.

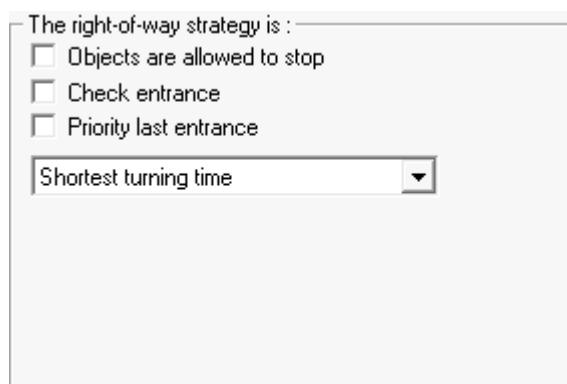


Figure 3.14: Shortest turning time

A similar right-of-way strategy exists for shuttles and paired shuttles. Here the current vehicle-position is taken into account and the object resulting in the **shortest running route** for the vehicle is selected.



3.3 Distribution Strategies

The following standard distribution strategies are offered for most modules having more than one exit. The choice of the strategy is made in the parameter dialog.

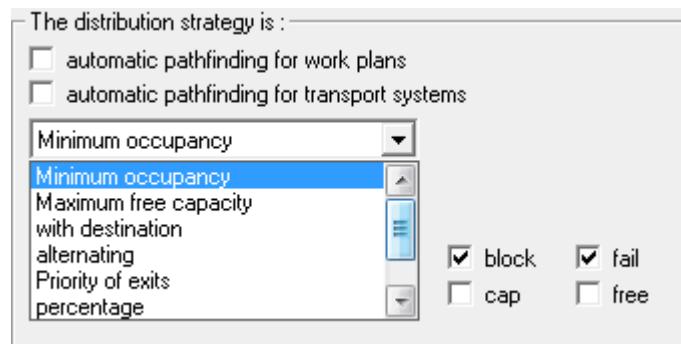


Figure 3.15: Distribution strategy

The distribution strategies are needed to determine the exit which is used by an object to leave the module. This is based on the assumption that an object can get from any entrance to any exit. The number of possible connections is only restricted for crossings because every single path would have to be defined by the user.

A decision in favor of one distribution strategy, if possible, is made immediately after the object has fully entered the module; this means the state 'free' is assigned to the entrance link. If a decision for a strategy has been made for the system, it cannot be changed, even if another strategy is chosen later (e.g. to clear a jam).

If a stop is not allowed in the element, the definition of the exit occurs automatically with the definition of the entry since it must be found out whether the object can pass through the element. See also the chapter *Module Bulks*.

During the analysis of the successive elements some of the successive elements can be excluded by means of the four options to the right. The following table shows the consequences of selecting the options.

block	Elements, which can be reached by a blocked junction, are not taken into consideration.
fail	Elements which are faulty are not taken into consideration.
cap(acity)	Elements which are full are not taken into consideration.
free	Elements that are not receptive are not considered.

Unlike **capacity**, **free** also considers whether the object can enter immediately. The capacity of the successive element can be ok, but if a *forward control* is activated, it is not receptive as long as the old object has not been completely removed.



If two exits have the same conditions for being taken by an object, each distribution rule defines the exit with the lowest exit number as destination for the object.

The individual strategy functions are as follows:

- see [minimum relative occupancy](#)
- see [minimum absolute occupancy](#)
- see [Maximum free capacity](#)
- see [alternating](#)
- see [Priority of exits](#)
- see [Bauschuld](#)
- see [percentage](#)
- see [destination-oriented](#)
- see [Self-defined](#)
- see [Shortest path](#)

3.3.1 Distribution Strategy Minimum relative Occupancy

If objects are distributed according to the strategy **Minimum relative occupancy of successive modules**, the exit is chosen whose successive module has the lowest percentage occupancy.

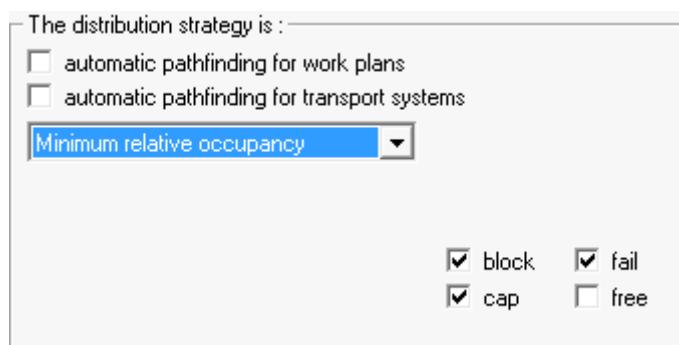


Figure 3.16: Minimum relative occupancy

3.3.2 Distribution Strategy Minimum absolute Occupancy

If the objects are distributed due to the strategy **minimum absolute occupancy** of the successive module, that exit is chosen whose successive module has the lowest absolute occupancy.

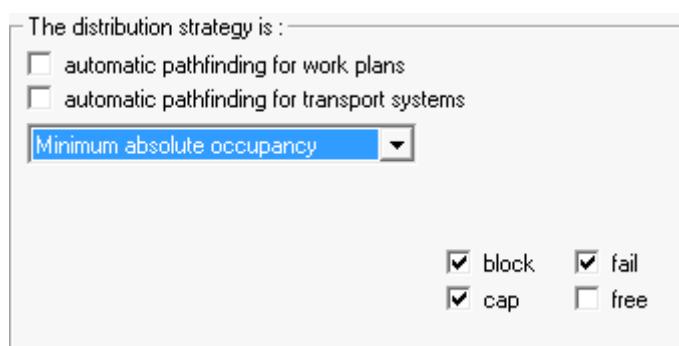


Figure 3.17: Minimum absolute occupancy



3.3.3 Distribution Strategy Maximum free Capacity

The distribution of objects based on **Maximum free capacity of successive modules** directs the object to the exit having a successor with maximum free capacity. This number is computed as the difference between capacity and current number of objects. The order of objects (e.g. in a conveying route) is not taken into account.

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

block fail
 cap free

Figure 3.18: Maximum free capacity

3.3.4 Distribution Strategy Alternating Distribution

For **alternating distribution** a lot size defining the number of objects to pass this exit in a consecutive order is determined. If the lot size is completed, the next exit according to its exit number is selected. If all exits have finished their assigned number, the first exit is selected again and the distribution process starts from the beginning. If the number of exits is restricted because of blockages, breakdowns or breaks, and the exit that is currently completing its number of objects is closed, the process is interrupted. No object is distributed until the exit is reopened and the alternating distribution can be continued.

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

No.	To module	Quantity
1	INF_139	1
2	ACC_112	2

block fail
 cap free

Figure 3.19: Alternating



3.3.5 Distribution Strategy Priority of Exits

With **Priority of exits**, each exit must be given a priority number. The distribution of objects is done according to a priority table and depends on the state of the successive modules. The exit having the minimum priority number and a successive module with free capacity is selected.

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

Priority of exits

No.	To module	Priority
1	INF_139	1
2	ACC_112	2

block fail
 cap free

Figure 3.20: Priority of exits

3.3.6 Distribution Strategy Percentage Distribution

The **Percentage distribution** assigns the object to an exit according to a random number. For every exit a probability is defined on a percentage basis. These percentages add up to 100% for all exits. If the number of exits is restricted, only the data of admissible exits are considered and added up. The random number produced is uniformly distributed between 0 and the computed sum.

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

percentage

No.	To module	Perc. Share
1	INF_139	10
2	ACC_112	30

block fail
 cap free

Figure 3.21: Percentage



3.3.7 Distribution Strategy Bauschuld

The distribution according to **Bauschuld** results in the output of an object being determined according to a fixed algorithm. This requires the specification of a percentage value for each output, whereby the total of the individual values does not necessarily have to result in 100 %. The Bauschuld algorithm ensures that after a fixed number of objects the indicated relation is exactly achieved.

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

No.	To module	Perc. Share			
1	INF_139	10	<input type="checkbox"/> block	<input checked="" type="checkbox"/> fail	
2	ACC_112	30	<input type="checkbox"/> cap	<input type="checkbox"/> free	

Figure 3.22: Bauschuld

3.3.8 Distribution Strategy Destination-oriented

The strategy **Destination-oriented** allows further restrictions on the number of exits to be considered. There are the following alternatives: For every exit, the object types that may pass it are defined. It is also possible to enter a destination parameter instead of the object type. This is done via the selection of the alternative **destination parameter** in the selection. Additionally to the type of the object can also be assigned to a destination parameter by means of a decision table; the user is free to define a parameter. Additionally a freely defined **Integer attribute** can be selected as criterion. This must be defined with the help of a decision table and assigned to the object.

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

No.	To module	Object type			
1	CRS_94	1	<input type="checkbox"/> block	<input checked="" type="checkbox"/> fail	

Figure 3.23: Destination-oriented

For an object, the number of possible exits is given by its type and the current situation. However, it is still possible to have more than one permissible exit, making an additional distribution strategy necessary. This strategy is called **secondary strategy** and it defines unambiguously the exit to be chosen. Therefore, the destination-oriented strategy plays a special role.

At first object types or destination parameters are defined in the list of exits; i.e. it is defined which object types pass which exit. The usage of wildcards '*' is possible. The first exit found



with an asterisk in the object list will be taken, when the current object type is not explicitly mentioned. If an object that has had no exit assigned to it tries to leave the module with the **destination-oriented** strategy, an error message is produced and the object leaves the module at exit 1.

In another area of the parameter-input window a secondary strategy can be determined. However, the user cannot choose the destination-oriented strategy now, because this would not make any sense. The strategies **percentage distribution** as well as **Priority of exits** are entered in the same way as described under the first strategy.

Slight changes are necessary due to **alternating** distribution. In order to guarantee a clear object transport at all times, the number of objects that pass one exit has to be defined dependent on the criterion, defined in the primary strategy (object type, destination parameter or integer attribute).

The screenshot shows the 'The secondary strategy is:' section of the parameter input window. It includes checkboxes for 'block' (checked), 'fail' (checked), 'cap' (unchecked), and 'free' (unchecked). A dropdown menu shows 'alternating' is selected. An 'Object type:' dropdown shows '1'. Below these are two rows for 'No. To module': row 1 has 'ACC_101' and 'Quantity 1'; row 2 has 'TUT_137' and 'Quantity 2'. To the right of these rows are buttons for 'Previous object type', 'Next object type', and 'Delete object type'.

Figure 3.24: Second strategy is alternating

In contrast to the primary strategy **Priority of exits**, with the secondary strategy, the priority can be specified for each type of object individually. If the priorities should apply for all objects, a wildcard (*) can be used as type of object.

The screenshot shows the 'The secondary strategy is:' section of the parameter input window. It includes checkboxes for 'block' (checked), 'fail' (checked), 'cap' (unchecked), and 'free' (unchecked). A dropdown menu shows 'Priority of exits' is selected. An 'Object type:' dropdown shows '1'. Below these are two rows for 'No. To module': row 1 has 'ACC_101' and 'Priority 0'; row 2 has 'TUT_137' and 'Priority 0'. To the right of these rows are buttons for 'Previous object type', 'Next object type', and 'Delete object type'.

Figure 3.25: Second strategy: Priority of exits



3.3.9 Distribution Strategy Self-defined

The strategy **Self-defined** offers the same possibilities as the analogous right-of-way strategy. The user can introduce any distribution rule desired. To do so, he can resort to decision tables (See: User manual for setting-up decision tables)

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

Self-defined

Edit distribution DT
Delete DT

block fail
 cap free

Figure 3.26: Self-defined

3.3.10 Distribution Strategy shortest Route

In addition to those strategies mentioned above, other module-specific rules can be defined.

Another distribution strategy can be chosen for *shuttles* and *paired shuttles*. Here the current vehicle position is taken into account and the exit resulting in the **Shortest running route** for the vehicle is selected.

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

Shortest route

block fail
 cap free

Figure 3.27: Shortest path



3.3.11 Distribution Strategy Automatic Path Finding

The distribution strategy is :

automatic pathfinding for work plans
 automatic pathfinding for transport systems

Priority of exits

No.	To module	Priority		
1	ACC_126	1	<input type="checkbox"/> block	<input checked="" type="checkbox"/> fail
2	ACC_127	2	<input type="checkbox"/> cap	<input type="checkbox"/> free
3	ACC_128	3		

Figure 3.28: Automatic path finding

If a transport system or the processing according to work plan is used in the model, the automated path finding can be used depending on the parameterization and the incoming object. Here the object is routed to the fastest way to the station of the next processing step (and/or transportation station).

Vehicle with load (load has work plan):

Path finding according to work plan	Path finding for transport systems	Destination
Yes	Anything	According to work plan
No	Yes	According to transport systems control
Yes	No	Standard distribution

Vehicle with load (load doesn't have work plan) or vehicle without load:

Path finding according to work plan	Path finding for transport systems	Destination
Anything	Yes	According to transport system control
Anything	No	Standard distribution

Object (with work plan):

Path finding according to work plan	Path finding for transport systems	Destination
Yes	Anything	According to work plan
No	Anything	Standard distribution

Object (without work plan):

Path finding according to work plan	Path finding for transport systems	Destination
Anything	Anything	Standard distribution



3.4 Choosing a Distribution Function

The distribution functions serve to describe random processes when defining time intervals. These time intervals are necessary in case of inter-arrival times, processing or assembly times.

Simulation reproduces the timing of events on the computer. In reality there are only very few processes for which timing can be exactly predicted. The timing of these events depends on random events which, however, can often be defined by percentage distribution. The processing times of objects by workers are close to normal distribution. The performance of automated storage and retrieval system can often be described as Erlang distributed.

DOSIMIS-3 provides several distribution functions for model processing or inter-arrival times:

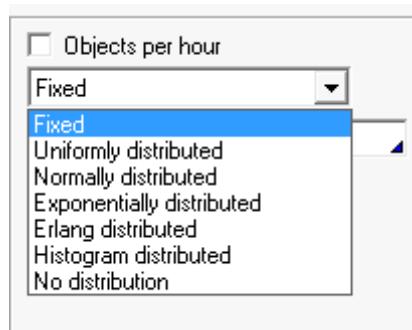


Figure 3.29: Distribution functions

The function *histogram distribution* is not available in every context.

See also:

- [Fixed distribution](#)
- [Uniformly distribution](#)
- [Normally distribution](#)
- [Exponentially distribution](#)
- [Erlang distribution](#)
- [Histogram distribution](#)

3.4.1 Fixed Distribution

The **fixed distribution** (= cycled distribution) assigns a fixed time to a process. It is not a stochastic process. The following box appears:

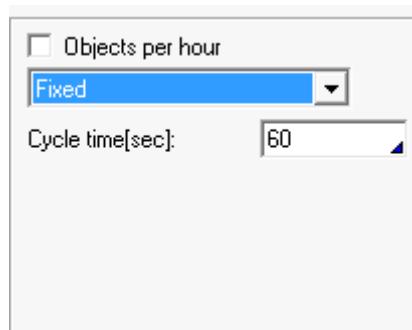


Figure 3.30: Parameters of Fixed distribution



Instead of the menu headline **Distribution of output time** other headlines can be shown. They inform about the event, e.g. for a source **Distribution of the exit time** or for a workstation **Distribution of processing time**.

3.4.2 Uniform Distribution

For the **Uniform distribution** the time is computed from a given interval with equal probabilities for each value inside the interval.

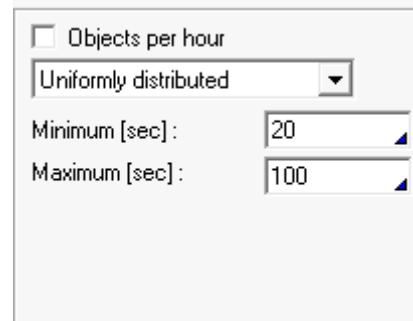


Figure 3.31: Parameters of Uniform distribution

Figure 3.32 explains this distribution for the time interval $[a, b]$ whereby the density function has the value

$$\frac{1}{a-b}$$

for all t inside $[a,b]$ and 0 for everything outside of this.

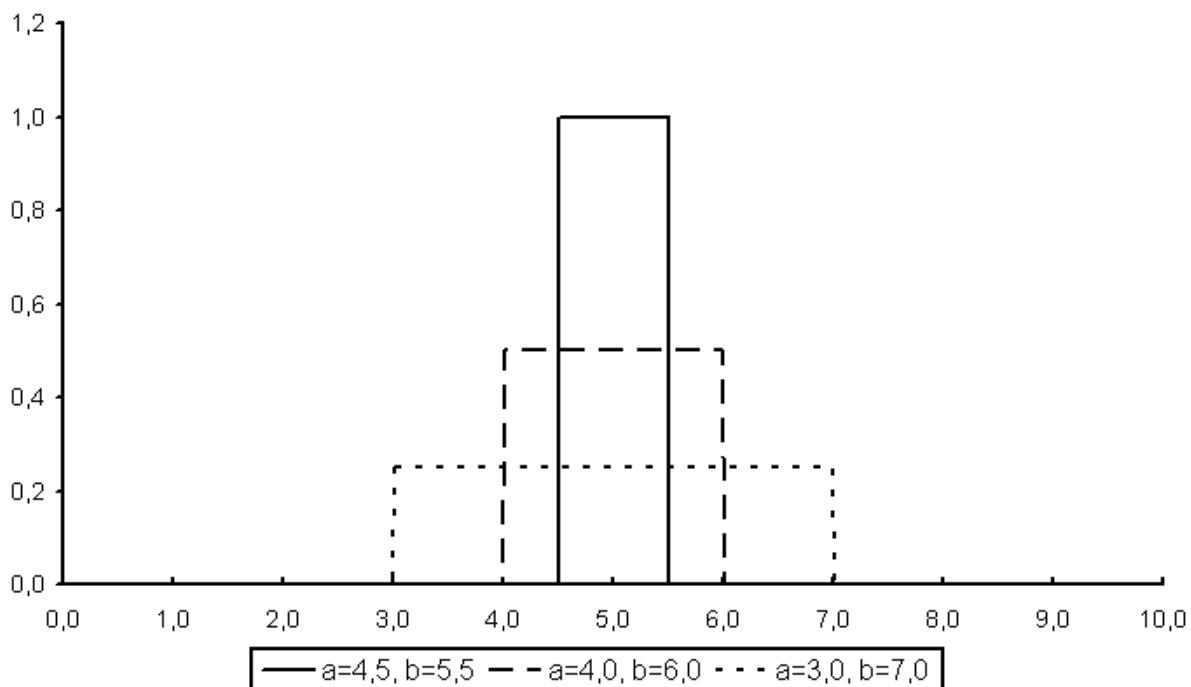


Figure 3.32: Density of a uniform distribution



3.4.3 Triangular Distribution

The **triangular distribution** (also called Simpson Distribution) calculates the time from a given time interval. It is defined by three values: Minimum, Maximum and Top value (also called *mode*).

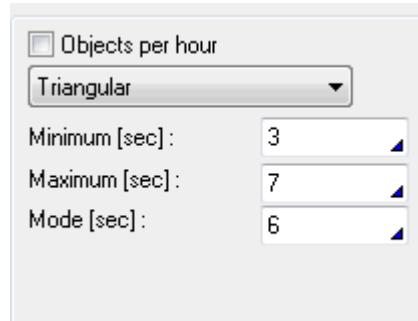


Figure 3.33: Triangle distribution

The mean value of the triangular distribution is

$$\frac{a + b + c}{3}$$

where a is the lower limit, b is the upper limit and c is the top value. The characteristics of the triangular distribution is clarified at Figure 3.34 in the time interval $[a,b]$, with the density function

$$f(t) = \begin{cases} \frac{2(t-a)}{(b-a)(c-a)} & \dots (a \leq t \leq c) \\ \frac{2(b-t)}{(b-a)(b-c)} & \dots (c \leq t \leq b) \\ 0 & \dots sonst \end{cases}$$

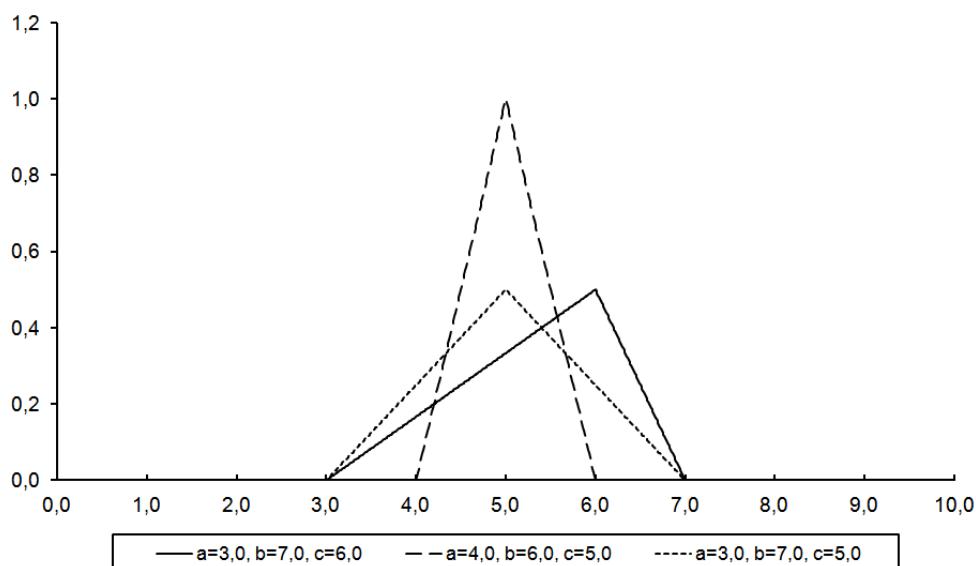


Figure 3.34: Density function of the triangular distribution



3.4.4 Normal Distribution

Normal **distribution** is one of the most important distribution functions for simulation. If, for example, the exact processing times of a workpiece are plotted against their frequency, the result is often the bell-shaped curve characteristic for a normal distribution.

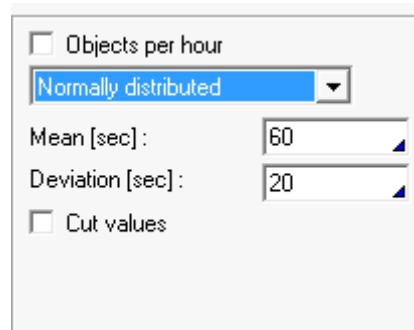


Figure 3.35: Parameters of Normal distribution

Normal distributions with an expected mean value λ and variance (standard deviation) σ have the density function:

$$f(t) = \frac{1}{\sigma * \sqrt{2 * \Pi}} * \exp\left(-\frac{(t - \lambda)^2}{2 * \sigma^2}\right)$$

In Figure 3.36 three normally-distributed curves for

$\lambda = 5.0$ [s] and $\sigma = 0.5, 1.0$ and 2.0 [s]

are shown.

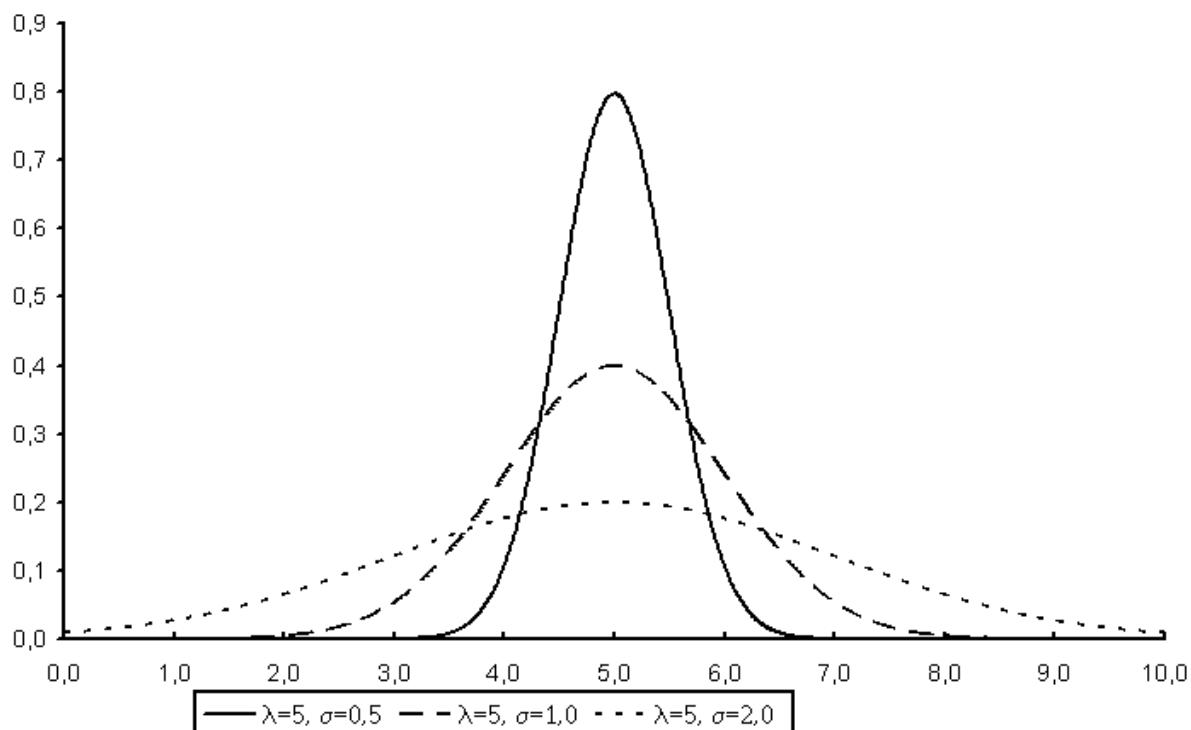


Figure 3.36: Density of normal distribution



Figure 3.37 shows the normal distribution curve for $\lambda = 5 \text{ [s]}$ and $\sigma = 1 \text{ [s]}$.

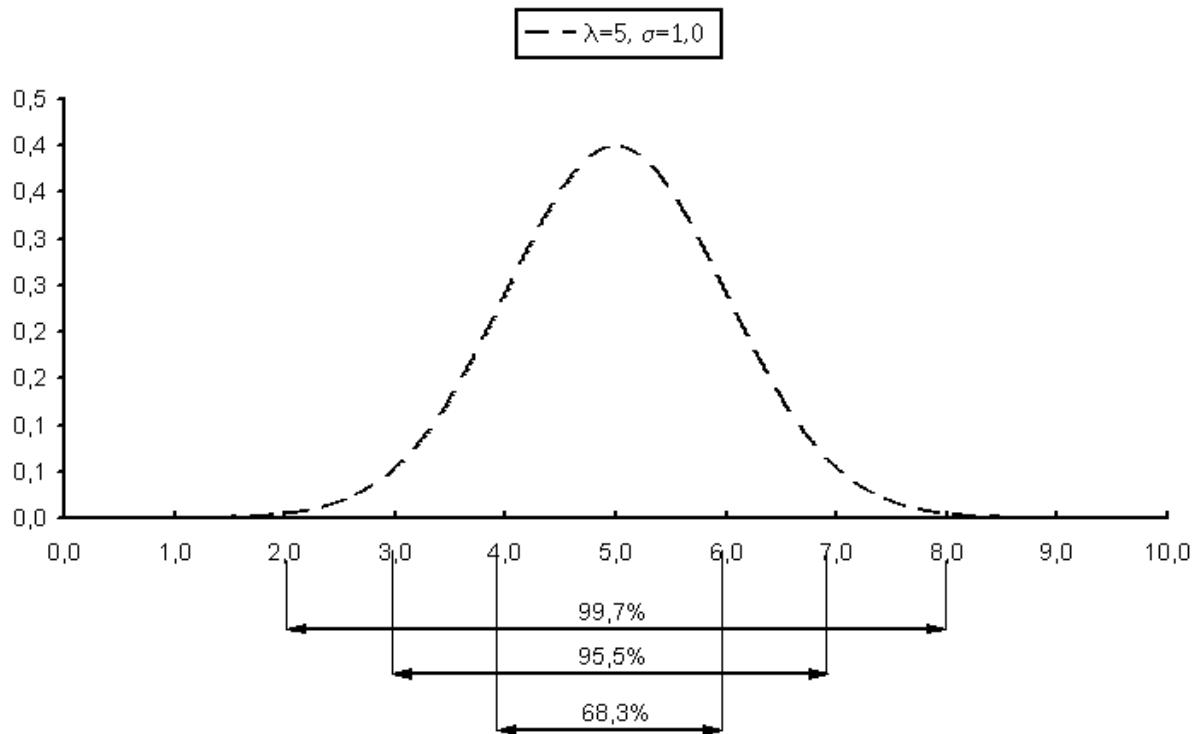


Figure 3.37: Example for a normal distribution

Explanation:

On the x-axis an appropriate scale to present the curve has been chosen. The graphic shows the probable processing time of objects at workstations based on stochastic data. The probability that an object is processed with a deviation of ± 1 second (see above) from the mean is up to 68.27%. With longer time intervals, probability increases.

Within the period ± 4 seconds the stochastic value is 99.9936% (not shown in the picture).

3.4.4.1 Usage

- Working time for one piece
- Picking time
- Processes, where humans work
- Arrival time



3.4.5 Exponential Distribution

Exponential **distribution** is clearly determined by a parameter λ .

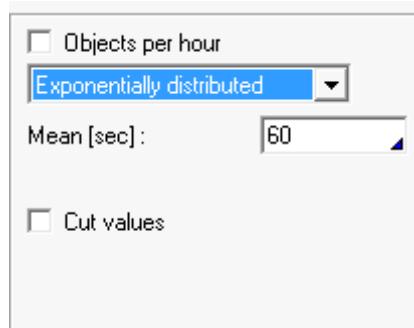


Figure 3.38: Parameters of Exponential distribution

The times are computed according to the exponential distribution

$$1 - \exp\{-\lambda \cdot t\}$$

and are distributed according to the pdf

$$\lambda \cdot \exp\{-\lambda \cdot t\} \cdot \left(\frac{1}{\lambda}\right)$$

The expected value of this distribution is

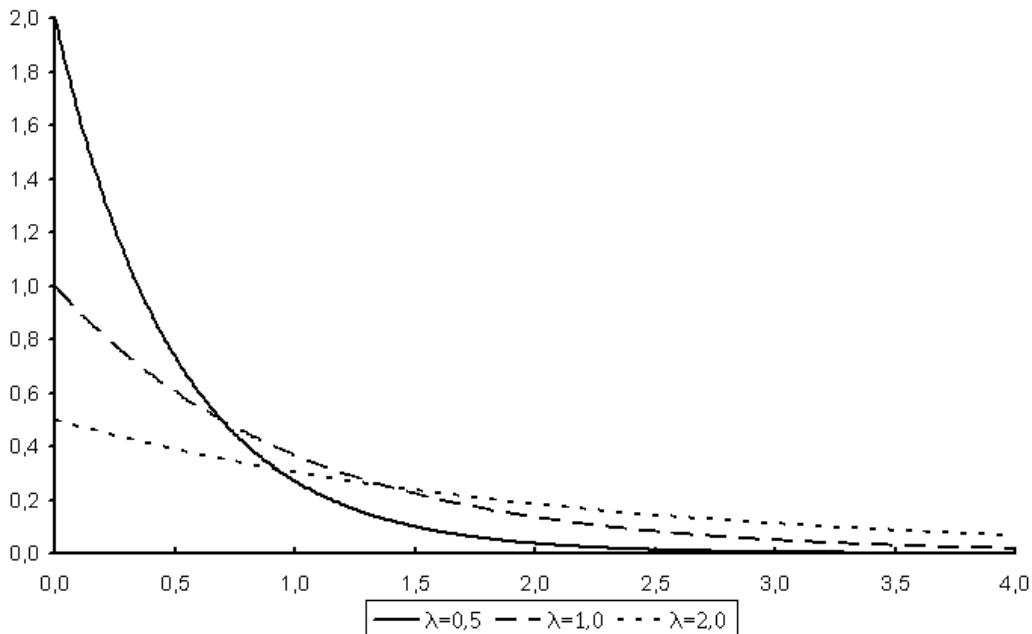


Figure 3.39: Density of an exponential distribution

3.4.5.1 Usage

- Sudden or catastrophic behavior
- The lifetime of each object is exponentially distributed, e.g. lifetime of a bulb
- Description of the length of a time interval between or before the appearance of an event, e.g. the time, until an electric component crashes.
- With random arrival processes the random variable behaves like the exponential distribution.



3.4.6 Erlang Distribution

For the simulation of pre-storage areas, it is sometimes necessary to model the behavior of sources and sinks with an **Erlang distributions**. An Erlang distribution with parameter k is a series of k identical exponential distributions.

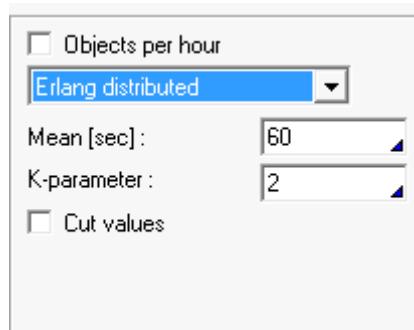


Figure 3.40: Parameters of Erlang distribution

Explanation: An Erlang distribution with k phases is a series of k identical exponential distributions, expressed in the density function of a k-Erlang distribution with an expectation

value (average) of $\frac{k}{\lambda}$ and the variance of $\frac{k}{\lambda^2}$

It complies with the formula:

$$\lambda * \frac{(\lambda - 1)^{k-1}}{(k-1)!} * \exp(-\lambda * t)$$

and is shown in Figure 3.41. Note: k and λ are parameters of the Erlang distribution.

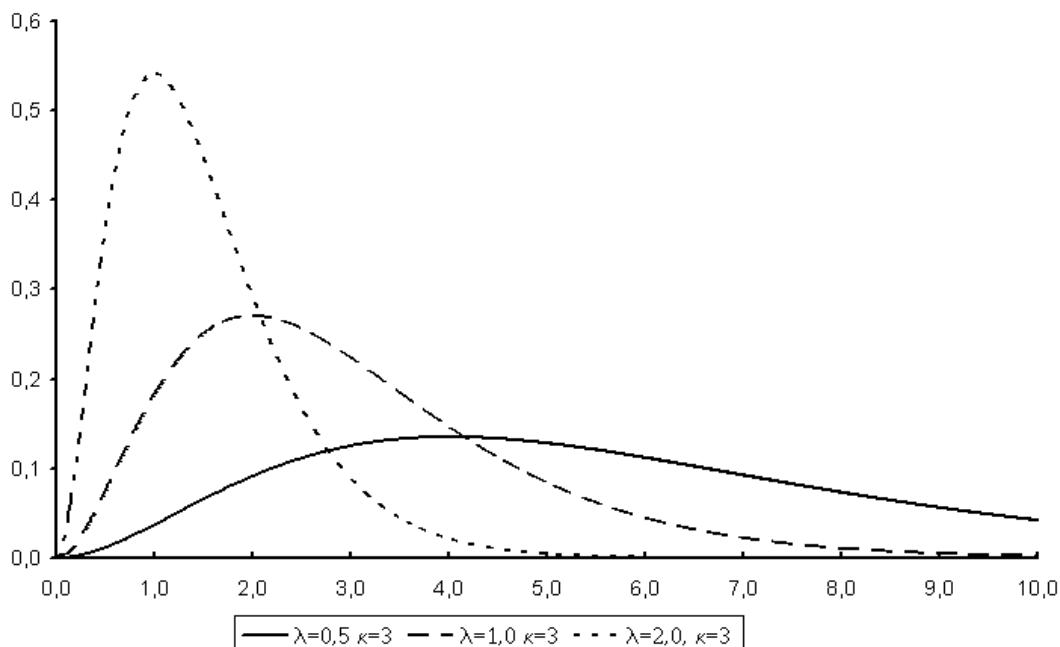


Figure 3.41: Density of an Erlang distribution



3.4.6.1 Usage

- Simulation of a storage, where the behavior of source and sink from and into the storage has to be reproduced.
- Reproducing the performance of automatic high-rack transport vehicle with ABC - zoning.
- Classical distribution in queuing theory.

3.4.7 Histogram Distribution

In addition to the five distribution functions, another function, i.e., **Histogram distribution** is offered for sources and sinks; here the intervals within which objects enter or leave the system, are indicated. For each interval the frequency has to be defined.

Histogram distributed	
Interval end	Frequency
10	2
30	3
50	0

Figure 3.42: Parameters of Histogram distribution

The determination of the next point of time takes two steps. With the help of the frequency the distance interval to the next point of time will be calculated. Then the distance will be calculated by a uniformly distributed value between the start (end of previous interval) and the end of the interval.

Example:

When starting a simulation run the first object generated in the source is delivered to the system at the time 0. The distribution function for the ejection time now defines the time at which objects will be delivered. A time within the period 0 and 30 seconds or within 50 and 80 seconds is chosen for the above-mentioned histogram. Each ejection time within an interval has the same probability. An ejection time between 30 and 50 seconds cannot arise since for this period the frequency is set to 0. Then the second object leaves the source at the time chosen. Another time is defined and the next object is only delivered after this time has elapsed. The frequencies to be defined by the user are relative values determining how often a special interval is considered over a longer period of time. The example of Figure 3.43 does not consider the interval between 30 and 50 seconds since the frequency was set to 0. Accordingly a time between 0 and 10 seconds is twice as frequent as between 70 and 80 seconds. The time between 10 and 30 seconds is three times as frequent as a time within the interval of 70 to 80 seconds. By assigning these frequency rates to intervals, the following histogram can be created:

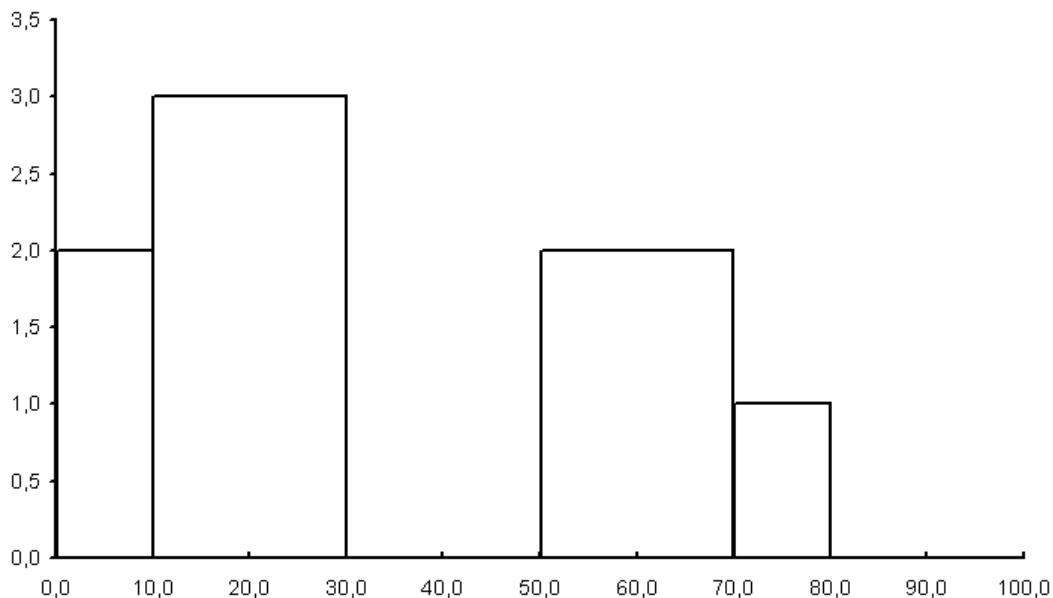


Figure 3.43: Histogram distribution

3.5 Additional Parameters of Distribution Functions

3.5.1 Limitation

Because the distribution functions normal distribution, exponential distribution and Erlang distribution can theoretically generate very large values, it is possible to limit the random number by a lower and an upper limit.

The dialog box is titled "MTTR". It contains the following settings:

- Distribution type: "Exponentially distributed" (selected from a dropdown menu).
- Mean [sec]: 300
- Cut values checkbox: checked.
- Lower limit [sec]: 20
- Upper limit [sec]: 3000

Figure 3.44: Limitation of a random process

When this feature is activated, in the simulation all values outside of the interval limits are ignored.

ATTENTION: This function should be used with **great care**, because it changes the mean value of the random function.



3.5.2 Objects per Hour

It is often much easier to set cycle times in the unit objects per hour. This is possible in source and sink. Instead of the mean value, the value of **Objects per hour** can be set. When changing the kind of input, the values are automatically recalculated.

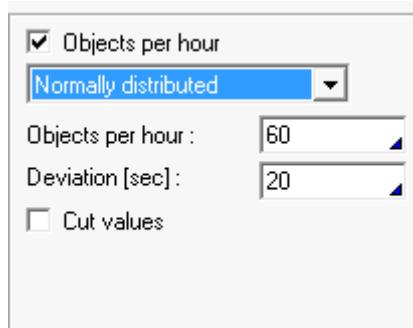


Figure 3.45: Objects per hour

3.6 Varying Distribution Function

Since time performance of the sources and sinks can vary during a considered period, the cycle times can be indicated as a function of an interval in these components. This is limited by the previous and the current interval end.

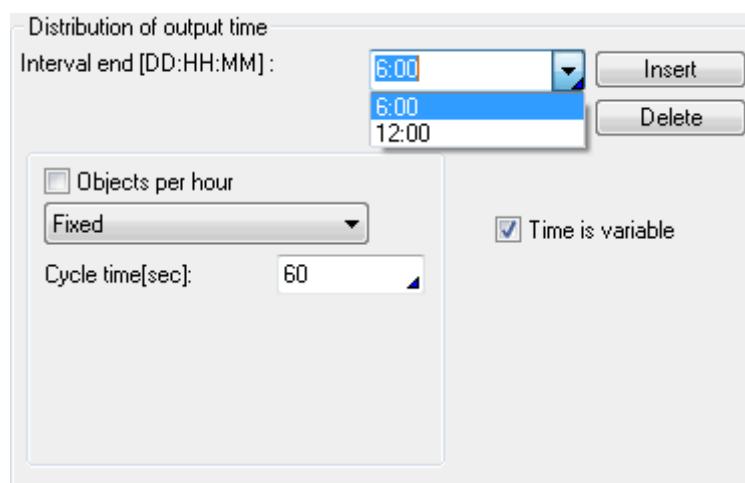


Figure 3.46: Varying distribution function

The further parameters are described now with the example of a source. The generation of the objects ends with the last interval end. In each interval the values given by the distribution function apply. When entering an interval end, the entry will be inserted at the right position. When reaching the last interval, the activities will be repeated. If this unwished, the end of the last interval should be as large as the duration of the simulation. If an interval without generation of objects is to be parameterized, then **No distribution** is to be selected from the check box for this interval.

3.6.1 Functionality

The values determined as entrance values are in accordance with the valid distribution function. If a random time lies outside the end of the interval, the distribution function of the



subsequent interval is continued. It is to be noted that the first value is cut by half. This applies also to the cycled distribution not at the average value of time after the interval end, but after half of this time.

3.7 Forward Control

For some module types such as buffer sections, bulk sections, slip-in modules and slip-out modules, workstations, distributors, merging stations, assembly and disassembly modules, pallet changer and crossings there is a conveying strategy known as forward control. This means that an object is only delivered from one module to its successor if there is enough capacity for a whole object and the object has completely entered the successive module.

For a module with a capacity of one object, forward control means that an object has to leave the module before another is allowed to enter (at one of its entrances). Therefore the exit junction of the module has to be in the state FREE and the object must have fully entered the next module.



Figure 3.47: Delivery with forward control

Given is the following situation: module 2 is forward-controlled and at junction 1 an object is waiting. Provided that module 2 has a capacity of 1, the object standing by enters the module after the other object has completely left module 2 and has completely entered module 3, thus when junction 2 is set to the state 'FREE'. If the capacity of module 2 is larger than 1, the object can only enter if enough space is available for it.

In case of transport modules (buffer section, conveying section, bulk section, LIFO buffer and block section) with more than one place, the strategy **Forward control** only affects the last place. An object can only enter the module if the last buffer place is fully available, i.e., the last object is at least in the second-last position. If, for example, module 2 is a conveying section with 3 sections of 3 m length each and the object is 3 m long as well, with the strategy forward control an object can only enter if the conveying section has one free place, i.e., 3 meters have to be available.

The range of applications of forward controlling is manifold. At branching or combining modules forward control can prevent objects from hindering one another within critical areas. With corner conveyors the next object is allowed to enter the conveyor when the object exiting has left complete.

3.8 Modules with a Length of 0

Some modules with a capacity of 1 can have a length of 0 (work station, assembly, crossing, ...). Axiomatic the process of an object can start after entering the module, when the old object has left the module completely. This check is omitted, if a length of 0 is indicated. An entering object releases the connecting junction immediately.



3.9 Module Bunch

Some modules, in which objects must not stop, have a special strategy with the determination of the new entrance. These modules are:

- Shuttle with at least two entrances and two exits
- Crossing with at least two entrances and two exits and the strategy in which objects are not allowed to stop.
- Turntables with at least two entrances and two exits

Within each of the modules listed above, checks are made before the arrival of an object to see whether, for the targeted exit in the current situation, a delivery can be made to the successive module. For this the following conditions must be given:

- The next module must be uninterrupted,
- there must be no pause at that moment,
- the junction must be accessible and
- the successor must be in a ready-to-accept state.

To ensure the latter condition, certain modules must be reserved until object delivery to avoid possible blockage during conveyance of one object from an object arriving from another entrance to the determined successive module. This holds for merging stations, slip-in modules, general turntables, shuttles and crossings which prevent an object arriving at another entrance, as well as buffer sections where emptying is prevented.

In a buffer module consisting of several modules the whole distance through the buffer module is successively determined by means of this process. If the distance through all modules can be passed without hindrance, these modules are reserved. In shuttles and on turntables the necessary vehicle positioning or turning of the tables is carried out.

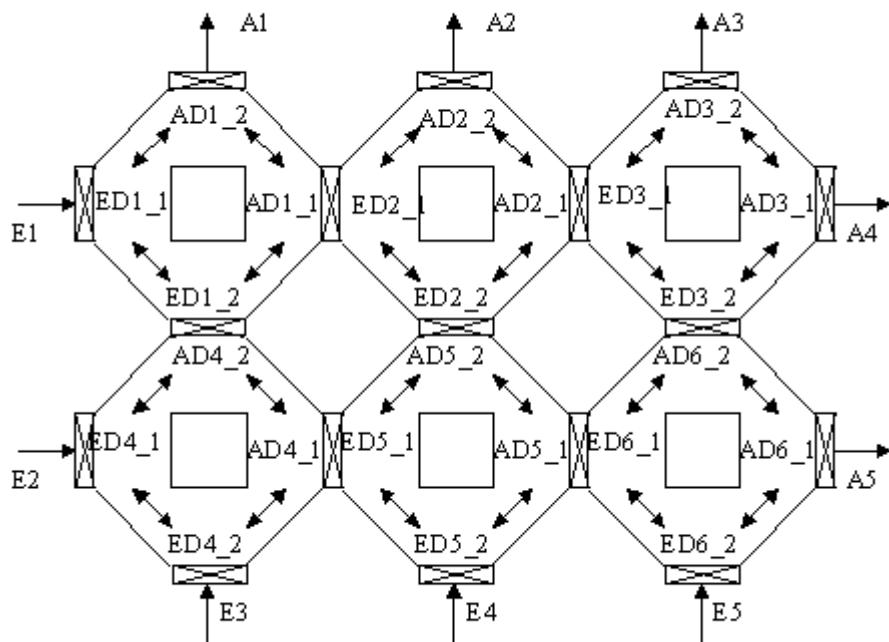


Figure 3.48: Example of a module bunch

A special property of the buffer is however that the selected right-of-way strategy must be **the same for all associated modules**. When determining the next entrances with the right-of-way mode, the entrances of the module are not considered, rather the entrances of the module bunch.

The last characteristic is clearly shown in the above figure, which shows a module buffer consisting of 6 turntables. The group/formation has 5 entrances and 5 exits which are marked *E1-E5* and *A1-A5*. When using the right-of-way strategy, entrances *E1-E5* apply for all modules in the group and not the turntable entrances *ED1_1-ED6_2*.



3.10 Defaults and Model Constants

This menu element gives the modeler the option to set default values for the parameters **object length**, **conveying speed** and **forward control** and to specify integer and real variables. Additionally variables can be defined, which can be referenced with the parameterization of the components at different locations. These can be defined also as a default value.

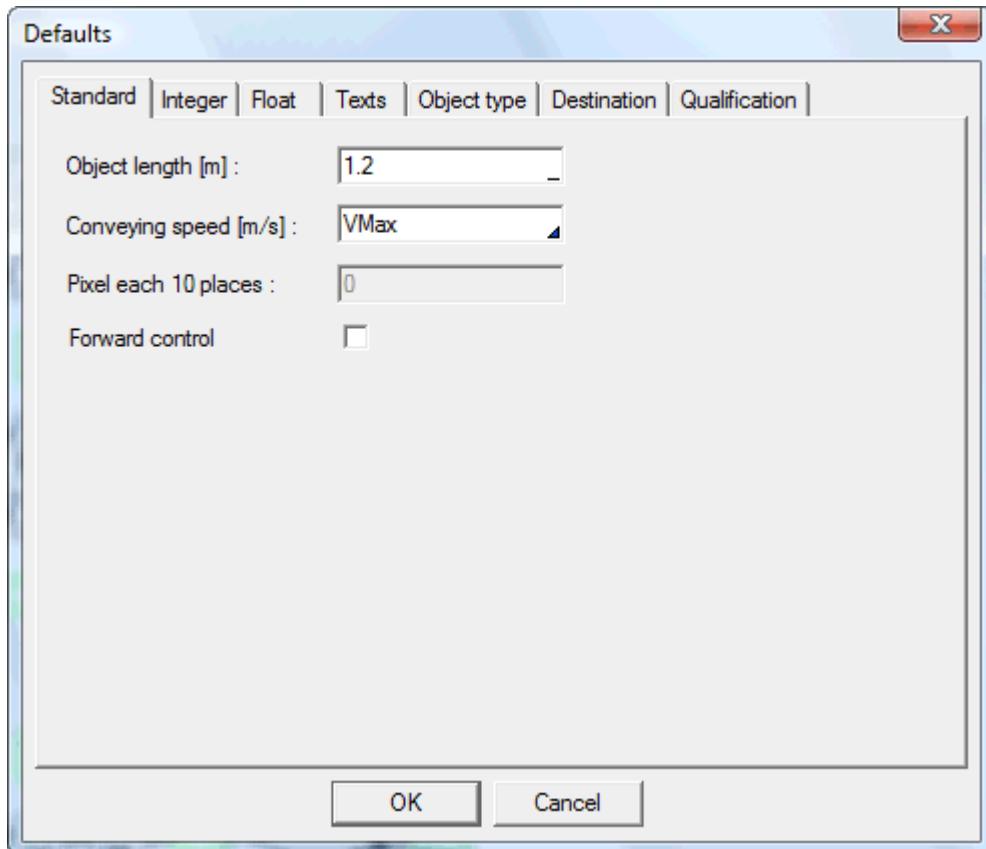


Figure 3.49: Parameters for setting of default values

The three default values are used when a new module is created within an MFS. Changing these values only influences new modules. Values of already created modules are not changed.

For accumulation conveyor, bulk section, workstation, assembly, disassembly and distributor the *speed* parameter, for crossing, turntable, swiveling belt, conveying circuit and combining element the *speed* parameters, at the pallet changer the *object* as well as the *pallet speed*, at the shuttle the *(un)loading speed*, at the discharger the *conveying* and *discharging speed* and at the infeed element the *conveying* and *infeed speed*. For accumulation conveyors and LIFO buffers the *segment length* is in addition to the speed reserved with the value entered for the **object length**. Same is valid for the loading path of the shuttle, the station lengths of the workstation, assembly and disassembly as well as the conveying path of the combining element and the distributor, the discharger and the infeed element.

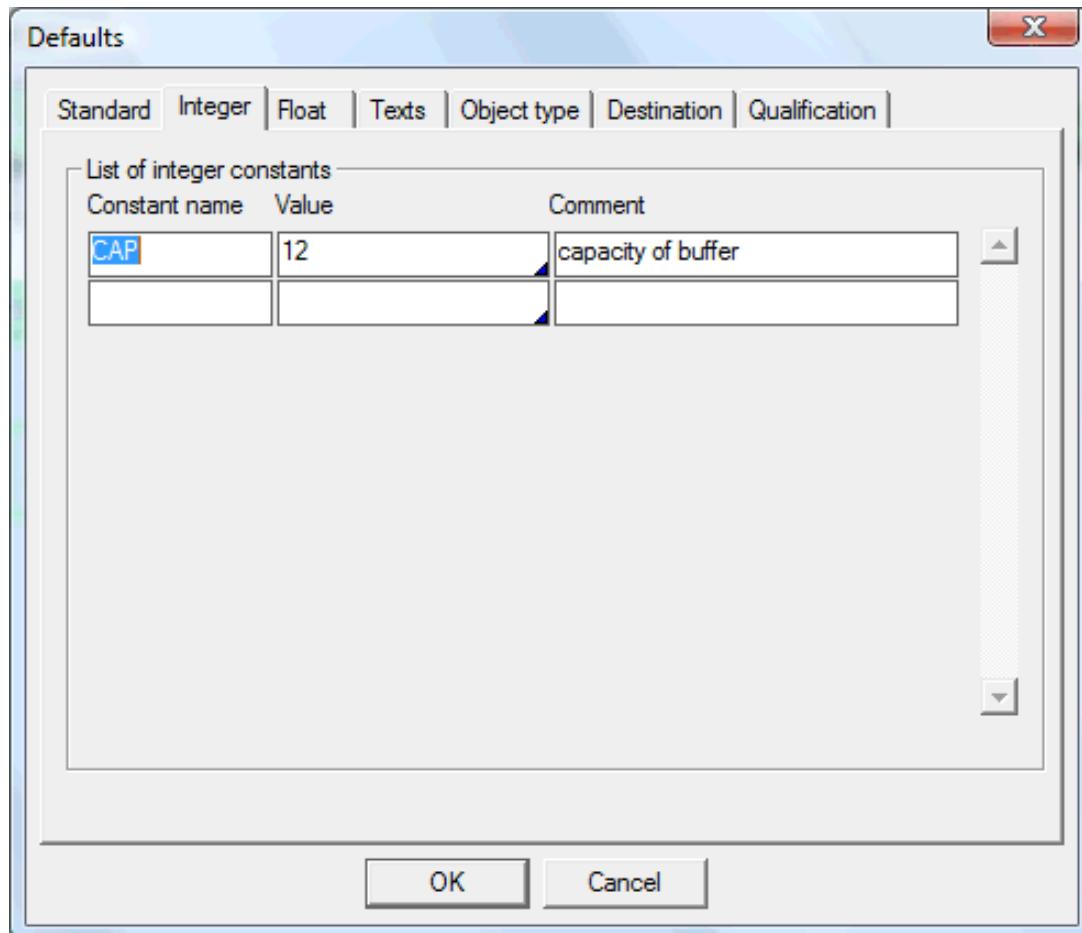


Figure 3.50: Parameters of model constants

The **integer** and **real constants** can be used within the parameter windows to set parameters of modules instead of concrete values. If a value is changed in the default window, all modules using the corresponding variable automatically reference the new value.

These constants can be used by the definition of the default values. New modules automatically refer to the constants.

The definition of the values can be made more transparent by formulas.

The comment fields do not have to be filled out. It is however recommended to use the comments for the better documentation of the constants.

The **object type** and **destination** and **worker qualification constants** serve to make the parameters of the items more transparent. The appropriate values with names can be entered, so that the data become clear. In the simulator however object types are treated identically for the same value.



Figure 3.51: Marking of input fields

Input fields, in which variables may be inputted, are marked at the bottom right corner by a blue triangle. The type of the variables is however only checked at the end of the input.



With OK the user leaves the processing of the default settings and can continue with his model.

3.11 Input Fields

Most parameter inputs take place in input fields. Here during the input it is already examined whether the parameters are permissible. For example, no letter can be entered in a field, where a number is to be entered. Erroneous entries are marked with a red border of the input mask. In addition, a tooltip with information may help to fix the problem. Many data can be specified apart from the values also by the model constants. For faster navigation, the selection of text constants, possible in an input field, can be indicated. This takes place by the context menu or the abbreviation CTRL-F. Then those model constants are indicated, which are permissible in this input field. This list is supplemented with the attributes, if they are defined.

The screenshot shows a software interface for managing work procedures. At the top, there's a header 'Work procedures' with a 'Name' dropdown set to 'ASS_1', a 'No.' field showing '1/1', and buttons for 'New', 'Delete', and 'Comment'. Below this is a section titled 'Distribution of working time' with a table. The table has columns: 'Object type', 'Distribution', 'Mean[sec]', 'Deviation[sec]', 'Strategy', 'Min.', and 'Max.'. There are four rows of data:

Object type	Distribution	Mean[sec]	Deviation[sec]	Strategy	Min.	Max.
1	Normally distribu	work	5	Minimal no.	1	3
10	Normally distribu	\$Working	5	Maximal no.	2	2
2	Normally distribu	=3600/144	5	Without		
*	Normally distribu	80	5	Maximal no.	2	3

Below this is a 'Set-up times' section with a table:

From object	To object	Distrib
1	10	Fixed
10	1	Fixed
2	*	Fixed
*	*	Fixed

A tooltip window is open over the 'Mean[sec]' field of the first row, listing options: 'VMin', 'VMax', 'work', 'setup', 'Pos1', 'Arb1', 'Ruesten', '\$Working', and '\$Setup'. To the right of the table, there are additional buttons for 'Employment of workers' and 'Employment of workers'.

Figure 3.52: Automated selection in input fields

The input often takes place in the form of lists. The input is complete, when each field contains a valid value. Inside a line you can move from field to field with the TAB key. Inside a column you can scroll with the cursor keys (up and down). The entries can be moved in the list. This happens with the pressed CTRL key and cursor keys. When opening a dialog there is only a limited number of lines that can be shown. That is why you have to consider, that there is more input in the non visible region. This can be accessed with the help of the scroll bar or by moving to the end of the list with the cursor keys.



3.11.1 Attributes

Attributes are values, which can be assigned to the objects or the modules via decision tables. These are referred to directly during the evaluation of the parameters without a further decision table being necessary. The references to attributes are marked with the \$ sign as a prefix. If the first object of the module possesses that indicated attribute, this value is being used; otherwise it is that of the module. If either inquiries are wrong or the return value is 0, an error message is shown.

3.11.2 Formulas

In many input fields formulas can be indicated. These are introduced by the symbol =. Formulas are evaluated once at the beginning of the simulation. Thus the parameters described by a formula remain constant during the simulation.

3.11.3 Percentage Values

For the normal distribution, the deviation can be indicated in percent. This is introduced by the symbol %. This value refers then to the appropriate portion of the mean value.

3.11.4 Erroneous Input

Edit fields, which can be identified as erroneous, will be marked with a red bar at the right border..

The screenshot shows a table titled "Distribution of working time" with a sub-section "Employment of workers". The table has columns: Object type, Distribution, Minimum[sec], Maximum[sec], Strategy, Min., Max., Qual., and Intrp. There are four rows of data:

Object type	Distribution	Minimum[sec]	Maximum[sec]	Strategy	Min.	Max.	Qual.	Intrp.
1	Normal	work	5	Minimal no.	1	3	2	10
1	Normal	\$Working	5	(Maximum[sec])				12
2	Uniformly	=3600/144	10		10			
*	Normal	80	5		Error: upper bound <= lower bound			5

A tooltip "Error: upper bound <= lower bound" is visible near the bottom right of the table, indicating an input validation error.

Figure 3.53: Erroneous Input

If you place the mouse over the edit field and wait a little bit, so the tooltip with information about this error appears.



3.12 Comments and Notes

All items of DOSIMIS-3 possess a field, in which an optional comment can be placed.

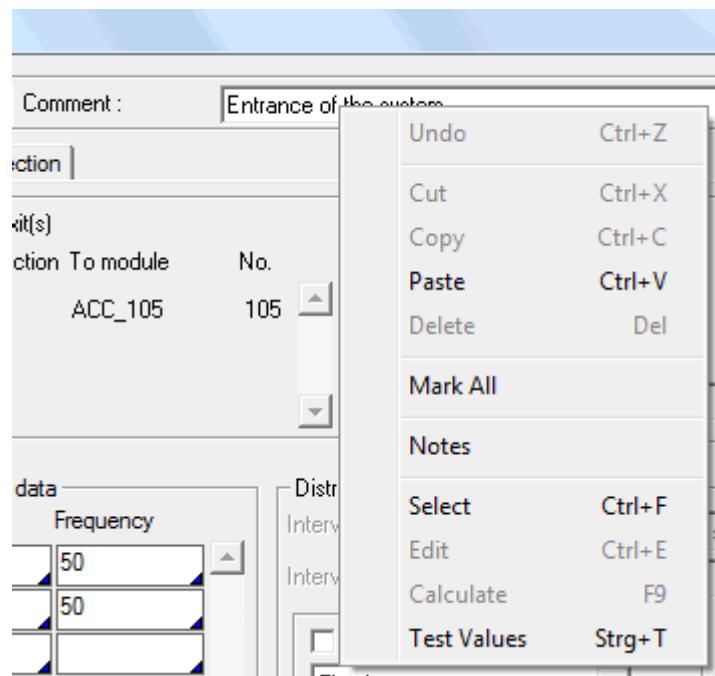


Figure 3.54: Context menu of input fields

Additionally a note can be indicated in many fields. This can be entered in the comment fields and in all fields, in which element variables can be used. These fields are marked at the bottom right by a blue triangle.

A note is multiline information, in which a parameter can be described more exactly. The input mode is made available by the context menu of the field. The input is terminated by the pressing of the return key. Line breaks can be made by pressing *Ctrl + input key* simultaneous.

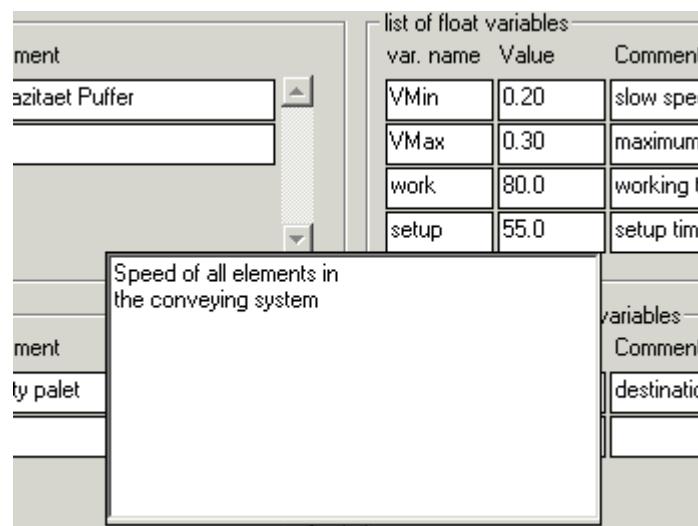


Figure 3.55: Definition of notes



Fields, for which notes were defined, are marked on the top right with a triangle. Additionally the note can be seen via tool tip without explicitly changing into the input mode. For this the mouse pointer is to be placed over the field for a short period of time.

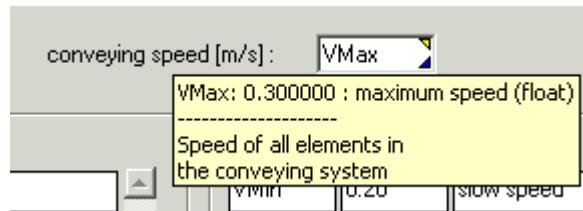


Figure 3.56: Tool tip of input fields

3.13 Test Values

In many input fields entered values can be marked as test values. This takes place with the context menu of the input field.

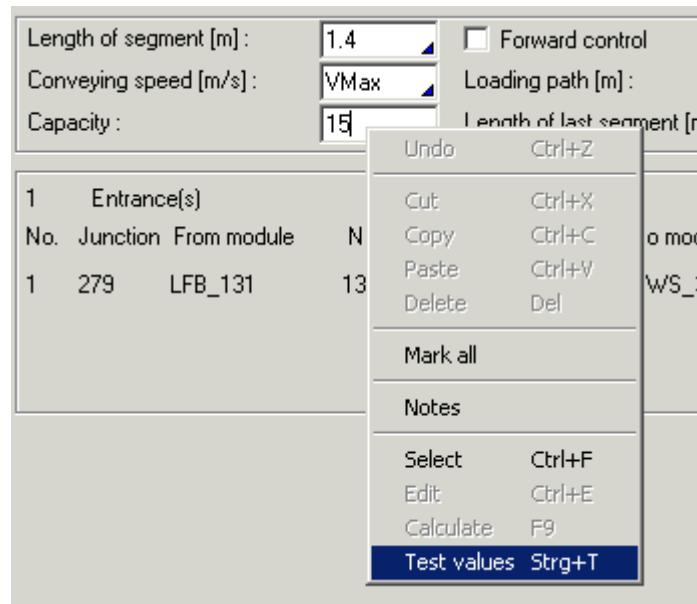


Figure 3.57: Context menu of input fields

A test value is a temporary change of a parameter for the purpose of an experiment. Further changes of this parameter are interpreted as test values as well. Undoing this mode takes place –in the test-value dialog (see below).

Fields defined as test values are marked in the upper left corner with a triangle. Additionally the value can be seen via tool tip, without explicitly changing into the input mode. For this the mouse pointer should be placed over the field for a short period of time.

If a parameter is defined as a test value, simulation takes place in the test mode. The simulation graphs are then drawn in green.

Elements, which possess a test parameter, can be selected via the menu Model>Selecting/Elements with test values.

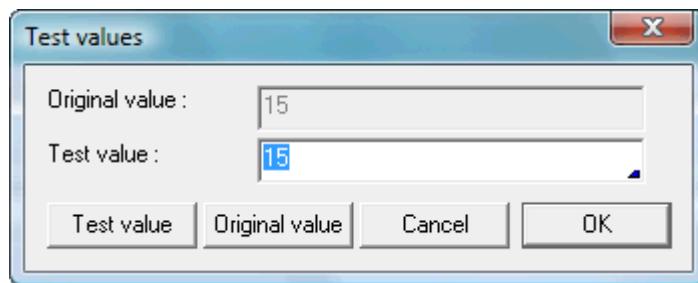


Figure 3.58: Definition of test values

The meanings of the four buttons are:

- Test value: The test value is taken as final parameter. The test mode for this parameter is waived.
- Original value: The original values are taken as final parameters. The test mode for this parameter is canceled.
- OK: The test value is taken as the parameter. The test mode for this parameter remains.
- Cancel: Everything stays with old values.

3.14 Lists

In Dosimis-3 many parameters are managed in lists. Often these provide names and comments to be used separately. Navigation through such lists (alternative animation parameters, work procedures in workstations...) happens with the help of the data in the head area.



Figure 3.59: Navigation through lists

Under **Name**, the name of the entry can be specified. The entry to be processed can also be selected here. On the right of the selection box the number of the entry and the total number of all entries are indicated for information. In the **Comment** field a specification of the data can be entered. If this is not sufficient, a longer entry can be supplemented via the underlying notes.

With the help of the button **New** and **Delete** the list can be extended or reduced. The next two buttons serve to shift the entries **upward** or **downward** within the list. The third small button produces also a new entry. However these are initialized with the parameters of the current entry (copy of the current entry).



3.15 Passivation of Elements

Many elements can be switched off in the simulation model, without deleting them from the model. For this the switch **passive** is to be set in the dialog.

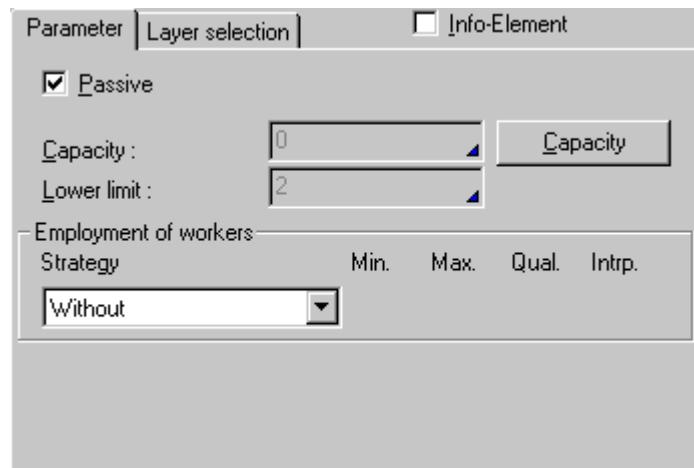


Figure 3.60: Passivation of elements

If an element is to be deactivated only for test purposes, this can be done by setting the check box to its third condition (grey-colored, see below).

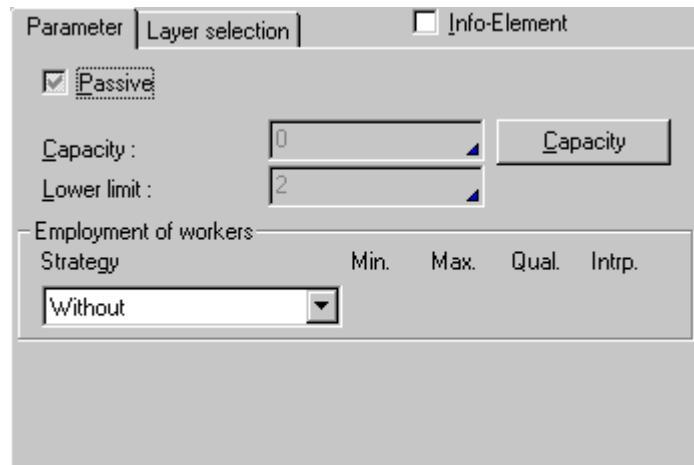


Figure 3.61: Passivation for test purposes

Passivated elements are marked in the layout with a green square in the left upper corner. If only sub-areas (e.g. groups of workers) are passivated, the information is marked with a green triangle. Additionally test mode is considered.

4 information are possible.

- The element is passive.
- The element is passive for test purpose.
- A part of the element is passive.
- A part of the element is passive for test purpose.



The indication is made when a whole group of elements is passivated in the simulation parameter.

3.16 Efficiency Factor

The efficiency factor is supported with many elements. With the help of the **efficiency factor** all processes can be accelerated or decelerated overall. The values from 0.01 to 10.00 are valid.

A screenshot of a software dialog box titled 'Efficiency factor'. It contains a single input field labeled 'Efficiency factor:' with the value '1.5' entered. There is a small arrow icon to the right of the input field.

Figure 3.62: Efficiency factor

The activities are accelerated by 50%.

3.17 Attributes

This sub-dialog of the modules makes it possible to assign attributes to the modules. These are evaluated at the beginning of the simulation. They can be used during the simulation directly in the decision tables. It is also possible to change the values of the attributes.

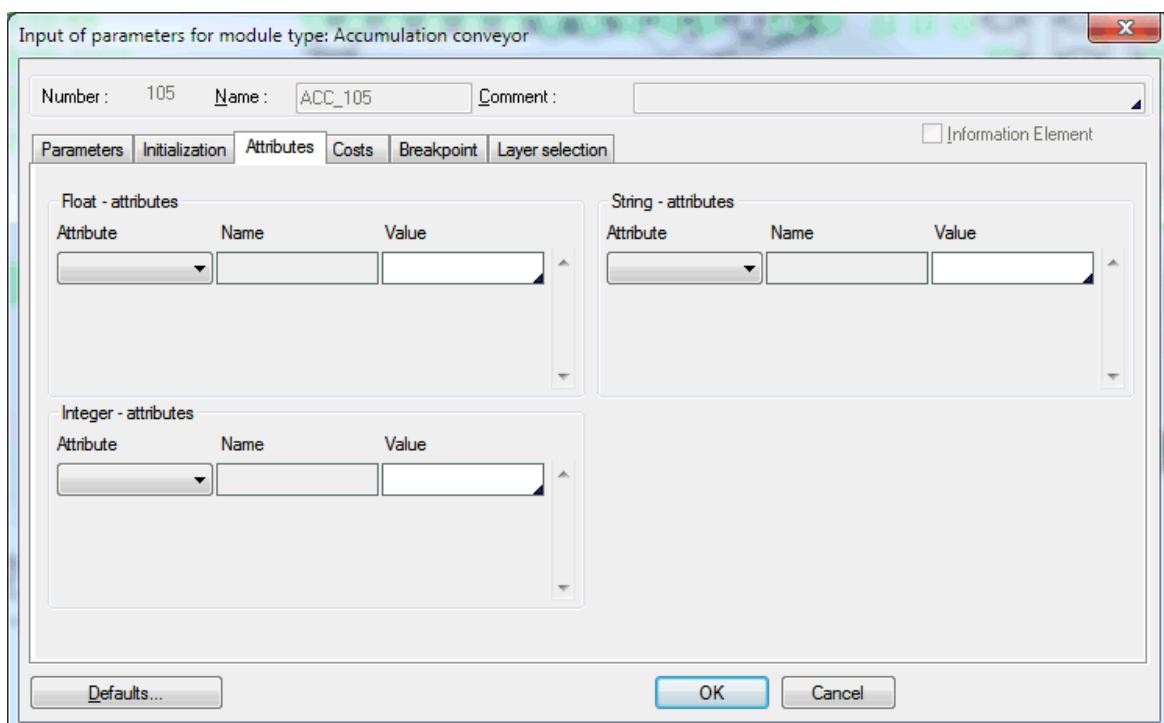


Figure 3.63: Parameters of attributes of a module



3.18 Costs

This sub-dialog of the modules makes it possible to specify the value of the objects as a function of events. On the basis of these entries, the capital commitment of the objects in the system can be seen. Objects, which are created in a source, do not possess any value. For objects, which are taken from the system (sink or assembly), the value is reset to 0.

Events entering and exiting from the module are possibly available. The process of the capital commitment can also be plotted HID_ERG_KOSTEN_ABS.

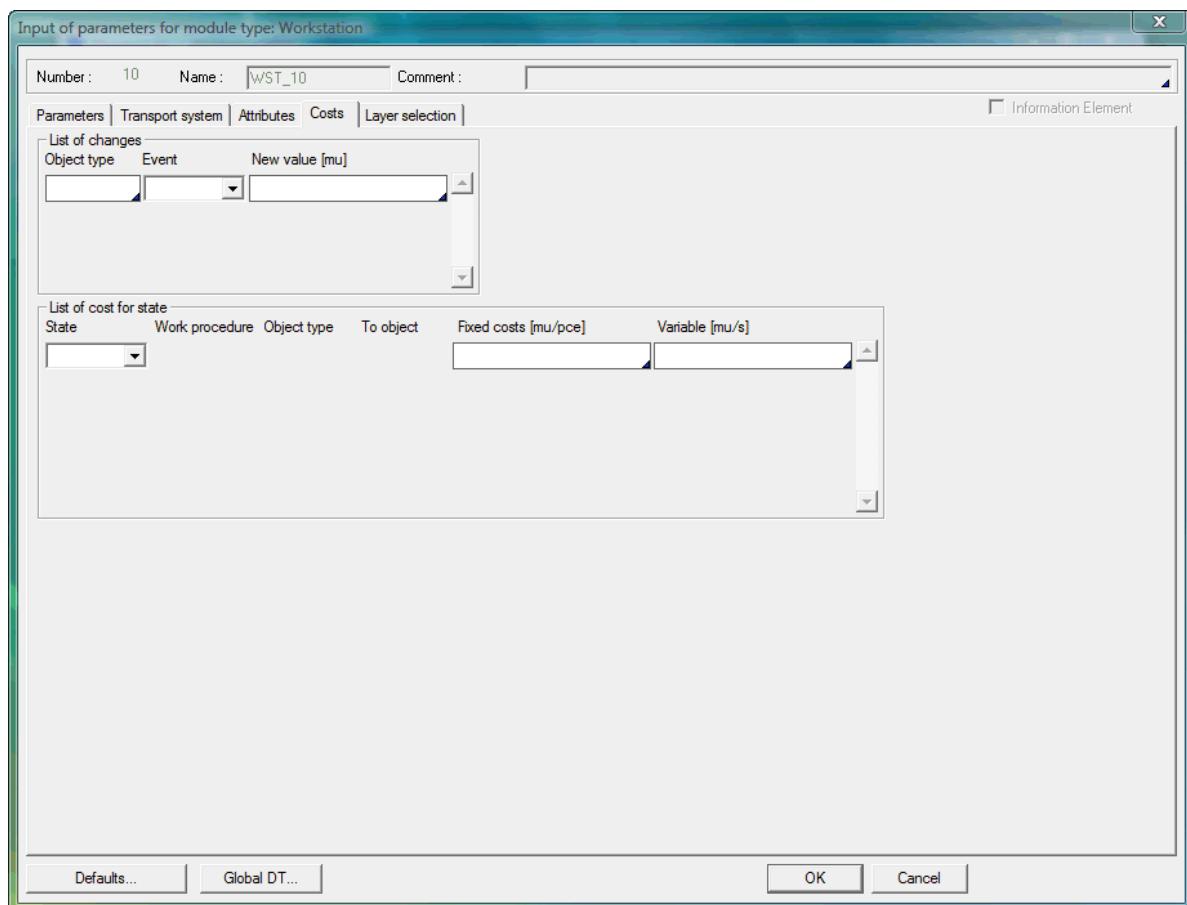


Figure 3.64: Parameters for the changing of cost of an object

In addition, each element can have state-dependent cost statistics. Therefore you have to assign a fixed and a variable amount to each state. With each change in the state these costs will be added. For stations, where processing can be defined, this can be fixed based on object type. During set-up this can be made in relationship to the previous object. These costs can be evaluated for each station and for each object type. When exiting a module, each object obtains the costs, which have been added in the module since the exit of the last object.



3.19 Initializing

It is possible to initialize modules with objects before the beginning of simulation.

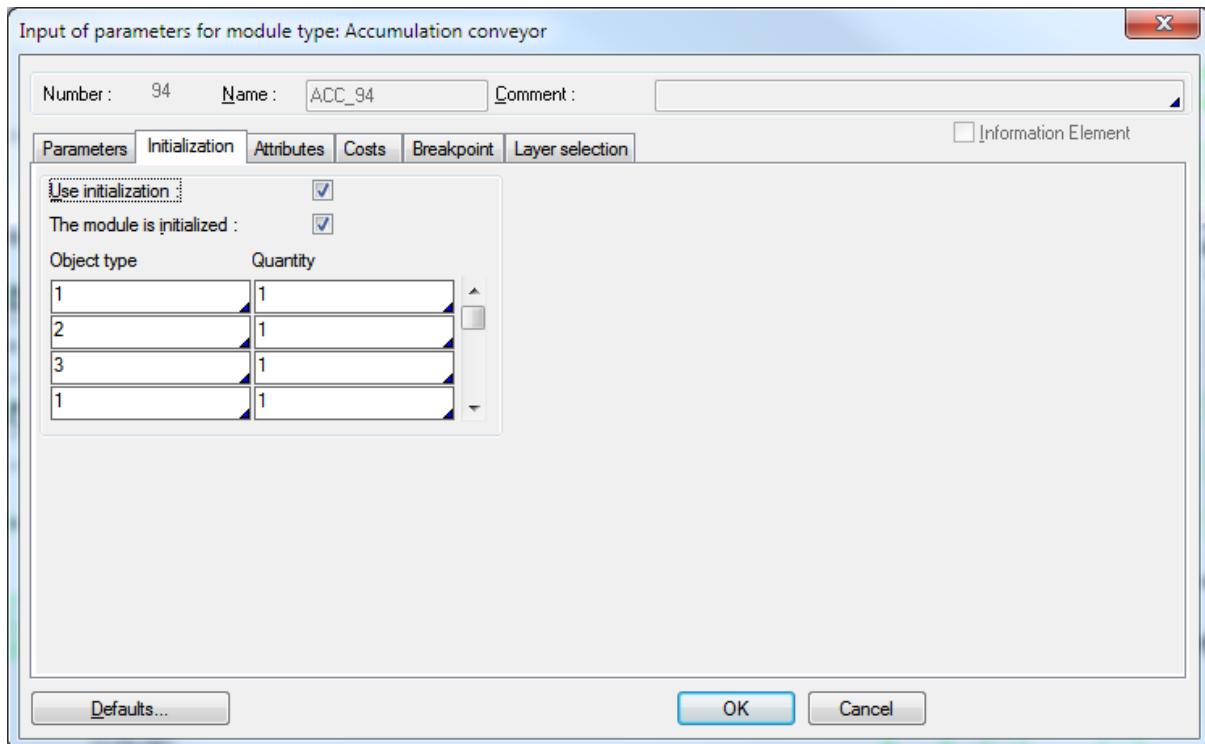


Figure 3.65: Parameter dialog of module initialization

In the sub-dialog **Initialization** the pre-setting **The module is initialized** is to be activated.

Then the list of object type and the respective quantity are to be defined. It is not necessary, to initialize the whole conveyor. The sum of quantities should not exceed the capacity of the module.

It must be ensured that the first position marks the exit of the module.

By the switch **Use initialization** one can determine, whether the defined objects have to be created at the beginning of the simulation. When the switch is deactivated, the parameters are kept, but the objects will not be generated.

Objects, which are initialized in tracks, are interpreted automatically as transport vehicles of transport systems.



3.20 Layer Selection

Each Dosimis-3 element possesses a list of layers, to which it is assigned. These can be drawn with the help of the different views.

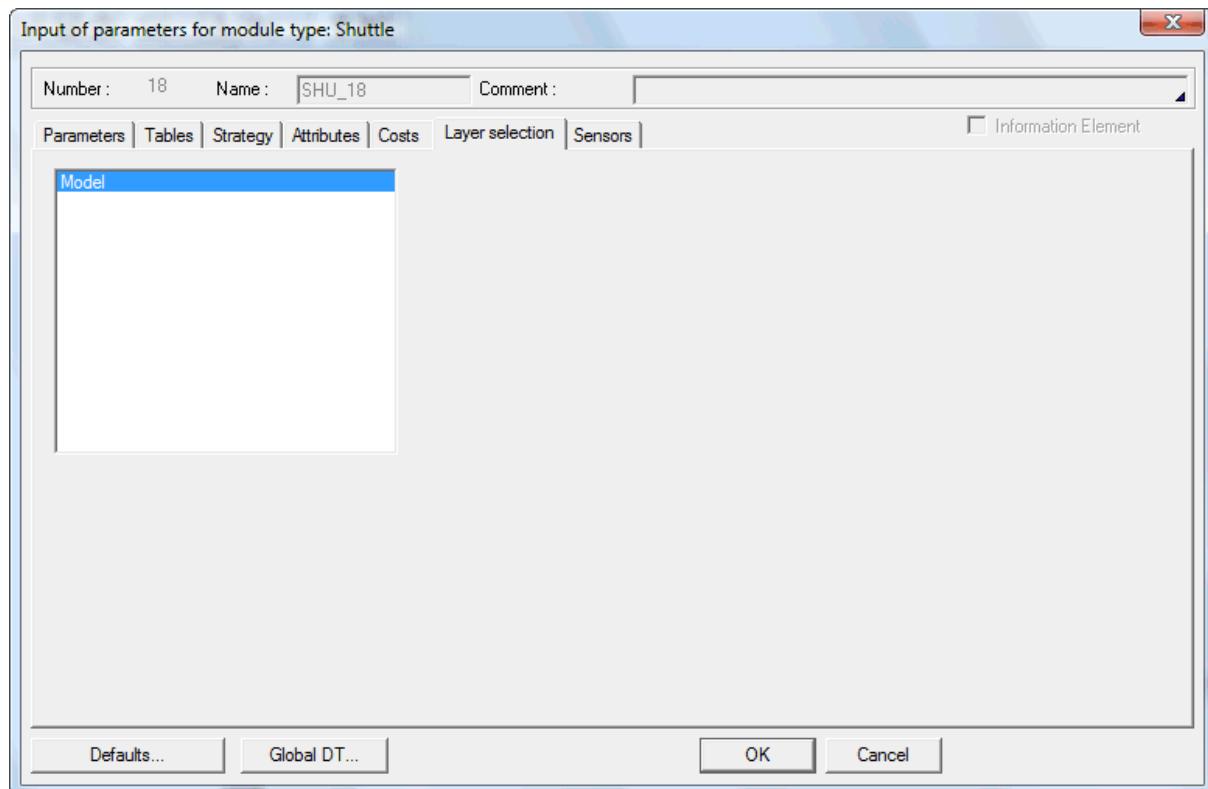


Figure 3.66: Parameter dialog of layer selection



3.21 Breakpoint

Each Dosimis-3 module can possess breakpoints. Thus the execution of the animation and/or on-line simulation can be interrupted on defined events. For online - simulation additionally still another condition can be defined, so that the execution of simulation is interrupted only then if the event is entered and the condition is true additionally. The condition is to be defined like a condition in the decision tables.

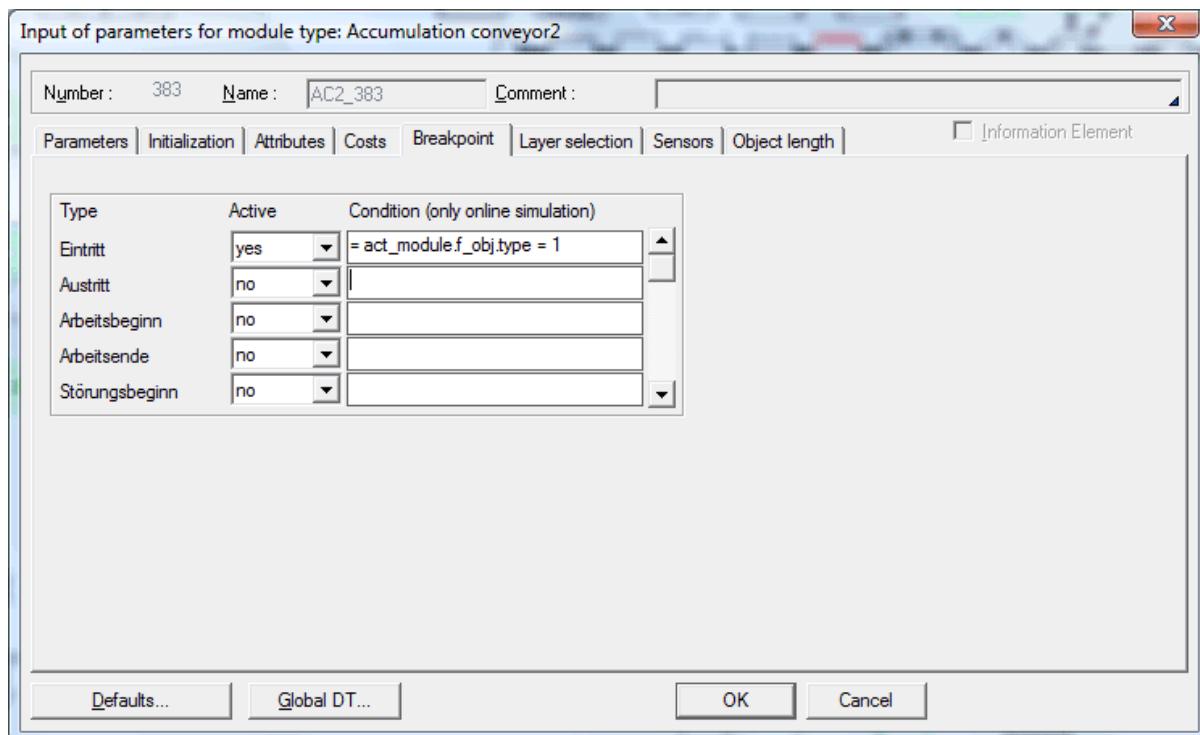


Figure 3.67: Parameter of break points



3.22 Collective Parameterization

possibility of transferring the parameter set of one element onto another element.

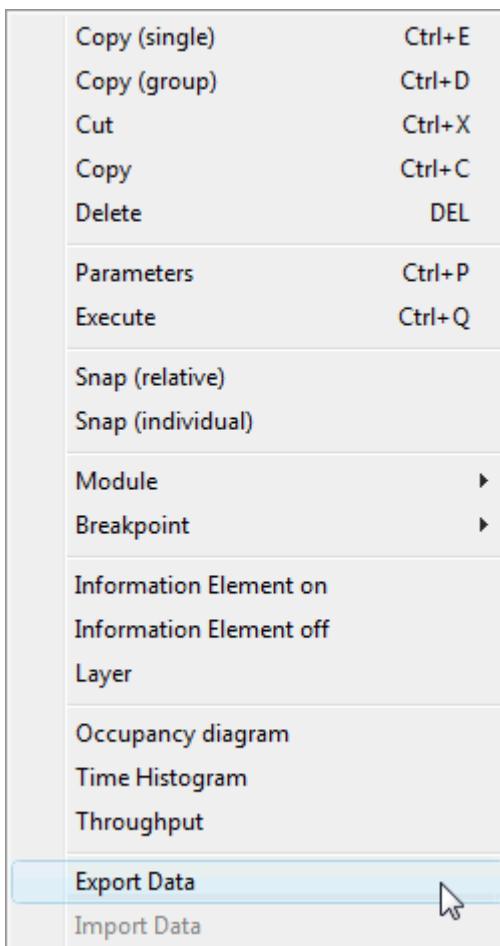


Figure 3.68: Context menu collective parameterization

For this exactly one element must be selected. With **Export Data** the parameters of this element are buffered temporary. Subsequently, the element is to be selected, to which the parameters will transfer to. This element must possess the same type as the output element. With the help of **Import Data** the parameters can be transferred. Several elements can be selected at the same time during the import of data.

3.23 Restrictions in Modeling

The special function mode of some modules necessitates the limitation of modular linking. Some module combinations lead in any case to a standstill in the system (deadlock), while others, in some cases only, represent a threat to the consistency of the system. Non-permitted module combinations lead to error messages when the system is checked later.

To avoid a standstill in the system, the assembly entrances of an assembly module may not be connected to the exit points of one of the three module types described in the previous chapter; because the answer to the question whether the way to the next module is free or not will always be negative for an assembly entrance.



A connection of two disassembly exit points of a disassembly module with two assembly entrances of an assembly module would also lead to a standstill in this system area. However, in certain circumstances this combination may work and is therefore permitted.

Linking the storage module with the assembly entrance of the assembly module is to be avoided: the assembly module always checks the modules connected to the assembly entrances to see whether the assembly lists can be fulfilled. This however is only possible for each first object in the connected module. In the storage, it does not have to be the first object which is to be stored externally. The result is that the basic object is sent on although the assembly list could have perhaps been fulfilled. This problem can be solved by simply interconnecting a buffer section between the storage and the assembly module.

Other problem modules are the conveying circuit and the paired shuttle which may not be connected to assembly entrances of an assembly unit either. Furthermore, connections with other conveying circuits or paired shuttles are not allowed. Connections to shuttles, turntables or intersections entail the danger of a deadlock, however these combinations are not generally ruled out. For all other combinations of modules there are no further restrictions.

The following table shows the restrictions previously mentioned.

	Not allowed	Dangerous
Assembly entrance	Shuttle Turntable with 2 entrances and exits Crossing (objects may not stop) Conveying circuit Paired shuttle Exit 2 of a unloading station	Disassembly exits Workstation
Paired shuttle	Paired shuttle	Turntable
Conveying circuit	Conveying circuit Assembly	Shuttle crossing
Storage	Entrances of modules with a right of way strategy which operate on properties of the object (priority of object type)	Modules with a right of way strategy

Table 3.2: Overview of restrictions

For further combinations of module types there are no restrictions.



4 Elements of Material Flow

Fundamentally the handicaps gained in the previous chapter are valid with the module parameterization. Since the default settings and particularly the module constants are very closely associated with the module parameterization, it is possible with every input dialog to change the list of the module constants or to complement by means of the button *Default*.

4.1 Source

In material flow systems, materials that might differ considerably are transported through the system. The module **SOURCE** models the entry point of the material in the system.

The parameters of a source are set in the window shown in the following picture.

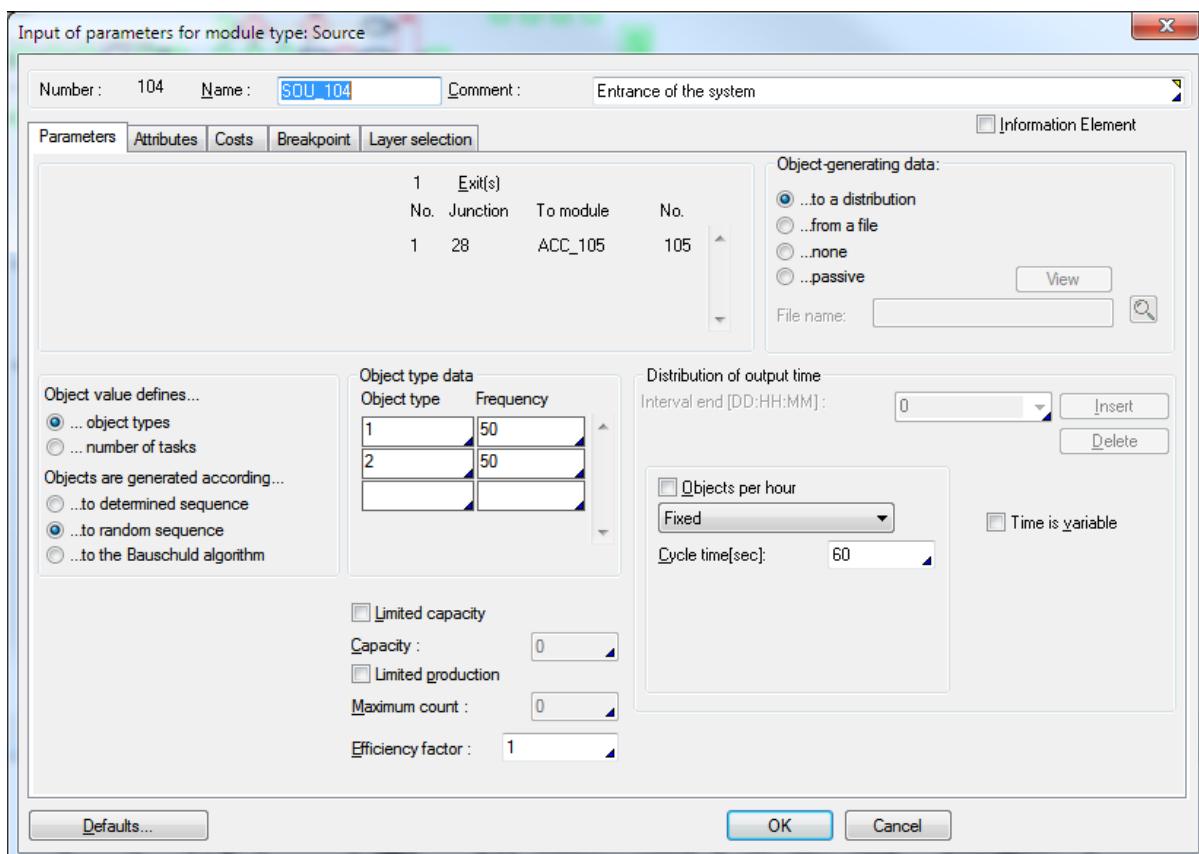


Figure 4.1: Parameters of the element type source

The element number as well as the standard name being formed by the module type and its number, e.g. SOU_1, appears.

If no link of the module has yet been established, the exit of the source is labeled **not linked**. Otherwise the name of the successive module appears. There are no entrances because the source is the entrance to the MFS.

In principle, a source can retain an arbitrary number of objects. By the switch **Limited capacity** the possibility of assigning a capacity to the source exists. Then the production



process will be switched off after reaching this value for such a time, till an object has left the source.

Additionally the generation can be **limited**. The creating process stops after the given number of objects.

4.1.1 Functionality

During simulation, two independent processes occur in a source: generating objects and supplying objects to the system. The kind of object generation is defined in the parameter dialog.

Internally, the generated objects are collected in a list, which is sorted according to the generation time points. For that, in each case first object on the list (that with the earliest generation time point), it is determined whether the succession module is receptive. If this should be the case, the object is handed over; otherwise the process stops up to the free message of the successive module.

The generation consists in turn of 2 steps. In the first, the time is determined, when the object is created and in the second the type of the object that is supposed to be created is determined.

4.1.2 Time of Generation

The times of the object generation are defined in the parameter dialog.

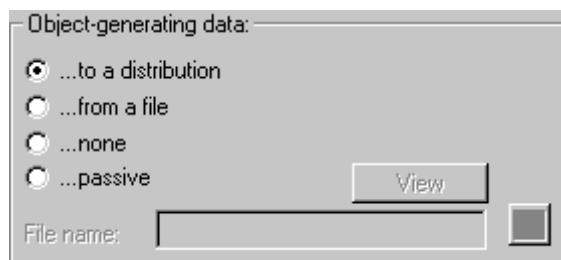


Figure 4.2: Selection object generation

There are 4 selections possible:

- Case 1)** **according to a distribution:** The chronological distance, by which the objects are supposed to be generated are to be determined in the *distribution of output time* window of the production times by means of a distribution function. The user can also access the histogram distribution in this case in addition to the other distribution functions.
- Case 2)** **from a file:** A file must be indicated, in which in a pre-set format is defined, when and which object is supposed to be generated (see generating from file). **Case 3)** **none:** If this switch is activated, no objects are given from this source. This case is to be considered if the generation of new objects on the basis of a decision table or a self-programmed global system state is supposed to occur. In the parameterization window of a source, the data do not need to be entered since the process of the generation is defined at other location.
- Case 4)** **passive:** The object generation occurs always exactly at the time when the previous object has left the source. The occupancy of the source is always 1.



4.1.3 Type of Object

The type of object generation is defined in the parameter dialog. For the generation without *multiple tasks* the variant must have **Objects values defines ... object types** activated. There are three variants of object generation.

Case 1) Objects are generated ... according to determined sequence

In this case, a lot size is assigned to each object type, determining the number of objects of this type being generated successively. The definition of these lot sizes has to be made. The user enters an object type to be defined (a natural number) and the corresponding lot size. If more than one object type was defined, the scrollbar on the right enables scrolling through the list. If, for example, object types *A*, *B* and *C* are defined with the lot sizes *d*, *e* and *f*, then at first *d* objects of the type *A*, then *e* objects *B* and finally *f* objects of the type *C* are generated. When the last entry of this list is reached, it starts again from the first entry.

Case 2) Objects are generated ... according to random sequence

In this case the amount of the objects is determined by means of a frequency for each object. The type of frequency that a generated object receives is determined with the aid of a random number, which the [distribution function](#) is given by a uniform distribution in the interval from 0 up to the sum of the frequencies.

Case 3) Objects are generated ... according to the Bauschuld algorithm.

In this case, the amount of the objects is determined by means of a frequency for each object. Due to a fixed technique (the Bauschuld technique) there is an order that guarantees that, after the generation of the amount of objects resulted from the sum of frequencies, every object was generated with exactly this frequency as well. There is exactly one Bauschuld order which can be calculated from a parameter set.

4.1.4 Objects with multiple Tasks

There is the possibility to assign not only one object type to a generated object but several. Every one of these types can be understood as a job definition which must be dealt with during the cycling through the materials flow system. The determination of this list happens in 2 steps. First the number of the object types is designated from the table Multitasking. This is distributed over all frequencies. In a second step the object types are created now, and indeed as much as determined in the first step. No object type can occur twice in this case. The maximum target number (number of tasks) is identical to the number of the defined object types. The objects generated then in the source carry this list of object types with themselves during the cycling through the materials flow system. This is differentiated as one between the type of the object (that is the number that is to be seen in the animation) and the list of the object types that is managed by these objects. This list is used for the decision at the processing and the destination location instead of the type of the object.

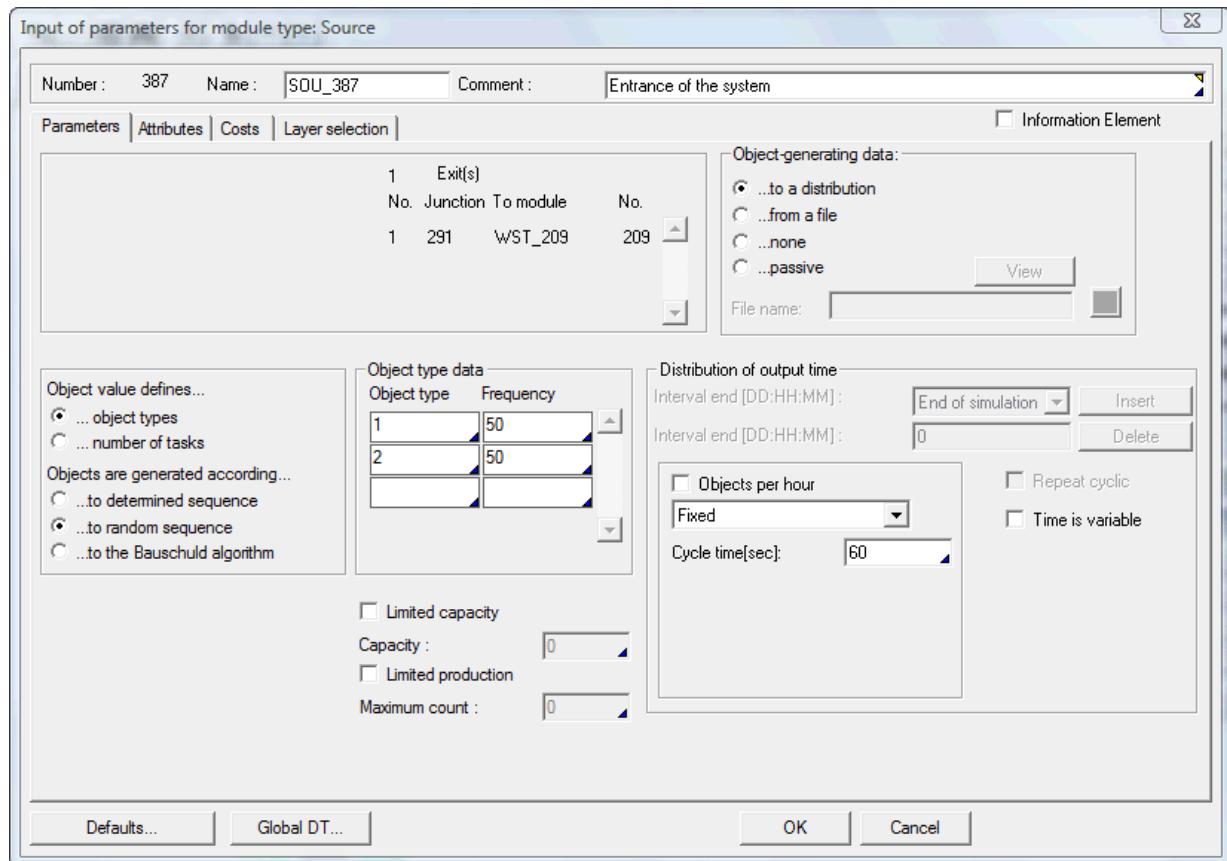


Figure 4.3: Parameters of a source with multiple tasks

The user can select between two versions, which determine in different ways, in which order the individual tasks occur. The workstation is to be understood as a task, in which one entry from the list of the object types can be processed. For that purpose, the branches of the simulation model for this station have to accordingly be parameterized.

Case 1) Objects are generated ... according to determined sequence

In this case work contents occur in accordance with ascending types of objects. The object is assigned in the source to the type of the smallest entry in the list.

At the workstation it is checked, which entries in the list are able to be processed. All object types from the list are deleted that follow in order directly after the smallest and can be processed at the station. The operating time results as the sum of the times of the found object types in the station. The new type of the object is again the smallest value of the remaining list.

The destination takes place with purposeful distribution as used according to type of the object.



Case 2) Objects are generated ... according to random sequence

In this case, the functioning contents can be worked in any sequence. The object receives the type in the source 1.

At the workstation it is checked, which entries in the whole list can be processed. At the same time all object types, which were found, are deleted from the list. The operating time arises as the sum of the single working time of the found object types in the workstation. The new type of the object is again 1.

In the distribution strategy, the types of the object are not requisitioned to the analysis, but rather all entries out of a task list of object types.

The decision results in alternatives with the secondary strategy.

In a workstation, however, all tasks are always performed and deleted as far as possible based on the sequence direction and the type of objects list of the workstation.

All list entries are run down; the object receives the type 0.

Here, it becomes clear that the strategy is only then meaningful, if at the branching points of the system, a destination-oriented distribution is used, because it can otherwise happen, that an object must be conveyed, based on other distribution rule, to a sink without having reached the workstations corresponding to its list of tasks (types).

4.1.5 Generating from File

If the generation data has to be taken **from a file**, only the name of the file has to be entered by the user. It is then determined in the file, when and which object types have to be generated.

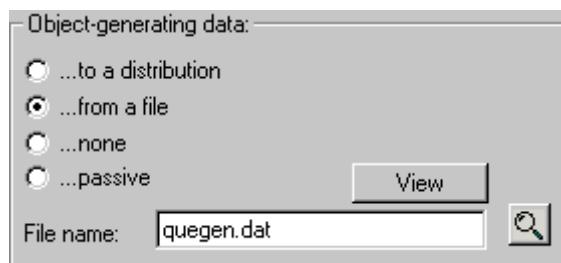


Figure 4.4: Generating from file

Via the button **View** the file will be shown. Changes could not be made.

A file which furnishes generation data has to be of the following format:

REF <time> AKT	reference time
DAT <file name>	succeeding file
COM <comment>	will not read
:	
GEN	beginning of generation data
<rel. time > <Objekt type>>	[optional: Attribute]

Figure 4.5: File including generation dates



After the label **GEN**, each line has to state the time and the corresponding object type (to be generated at that time). They are relative time values. The reference time, to which these relative time values refer, can be found in the first line of the corresponding file. After the label **REF**, either an exact reference time to which the relative time values are added, can be entered; or the relative time values that refer to the time when the file was called. In the latter case the file has to start with **AKT**. Moreover, the user can determine which file is selected to continue the object generation after the current file has been completely processed. The name of the succeeding file has to be entered after the abbreviation **DAT**. Thus, it is even possible to process the same file recursively by choosing the current file as succeeding file. This approach only makes sense if **AKT** is chosen as reference time.

The following figure shows the generation process by means of two examples.

example 1:

```
file name: data1

REF 300
DAT data2
COM This is an example file
COM for object generation with
COM a fixed reference time
GEN
 30  1
 60  2
 90  3
120  4
120  4

file name : data2

REF 600
DAT data3
COM This file is called up
COM after processing of 'data1'
GEN
 30  1
 60  2
 90  3
120  4
```

**results of example 1:**

```

time after sim. type numbers
start
330      1      1
360      2      1
390      3      1
420      4      2
630      1      1
660      2      1
690      3      1
720      4      1
:
continue with 'data3' (not defined)

```

Figure 4.6: Process of generating Example - 1

Example 1: At the time 330 seconds (reference time plus 30 seconds) an object of type 1 is generated. Objects of the types 2, 3 and 4 are generated in an interval of 30 seconds each; type 4 is produced with a quantity of 2. Thus, the file *DATA1* is completely processed. Further generation data is now taken from the file *DATA2*. At the time 630 seconds (reference time plus 30 seconds) again an object of type 1 and, every 30 seconds in succession, the types 2,3 respectively 4 are generated. Finally the file *DATA3* is called. The reference time of this file must exceed 720 seconds.

example 2

file name: data4

```

AKT
DAT data4
COM This is an example file
COM for object generation without
COM a fixed reference time
GEN
 30  1
 60  2
 90  3
120  4
120  4

```


results of example 2:

```

time after sim.type numbers
beginn
 30          1      1
 60          2      1
 90          3      1
120          4      2
150          1      1
180          2      1
210          3      1
240          4      2
270          1      1
:

```

Figure 4.7: Process of generating Example -2

Example 2: The generation times are added to the time of calling the file. Thus, at first the types 1, 2, 3 and 4 having the object numbers 1, 1, 1 and 2 are each generated after an interval of 30 seconds each. At the time 120 seconds, the file *DATA1* is called recursively, i.e. after another 30 seconds (= at the time 150 seconds), a type 1 is generated again; then types 2, 3 and 4 with the corresponding probabilities. The process continues up to the end of the simulation; thus every 30 seconds a new type is generated with type 4 having the frequency 2.

4.1.6 Attributes from File

After the key values (*time* and *type of object* with the source; *stock area*, *type of object* and *number* with the storage) an "arbitrary" number of values can follow (max 2048 letters). Here 2 cases are to be differentiated:

- The first letter is a number. Then this value becomes the integer attributes. A-J and afterwards the float attributes A-J are assigned
- The first letter is not a number. Then beside the first field ("Field1") also the following field ("Field2") will be read. The character string *act_object.Field1 := Field2* will be created and evaluated for each initialized object like an action of a decision table.

This approach is very flexible. Thus self defined attributes can be initialized. The pair of fields >*intatt.MaxValue 2*< assigns the value 2 to the Integer attribute „*MaxValue*“ of the created object. One can assign so for example also a value to the destination parameter of the object (*Destparam*).

Example:

```

AKT
COM example file - definition of attributes
GEN
0   10   0   0   0   2   3
12  20   8   9   10
40  30   real.C 10.1 destparam 1   intatt.MaxValue 2

```

Figure 4.8: Generation of data with attributes

At the time 0 seconds, an object of type 10, whose integer attributes E and F receive the values 2 and 3, is generated. At the time 12 seconds, the integer-attribute A is assigned to the



value 8, B the value 9 and C the value 10 for object 20. For object 30, the real-attribute C is set to the value 10.1, the destination parameter of the object is set to 1 and the integer attribute *MaxValue* is set to 2.

It is not necessary to indicate all twenty values, but attributes not defined are not initialized either. In order to avoid type conflicts, it is important not to enter real numbers for the 10 integer attributes.

4.2 Sink

After having passed through the system, objects leave the system at specially defined points. These points are modeled by the module type **SINK**. Every sink has an entrance linked to another module of the system. An object entering a sink immediately sets the link at the entrance to FREE.

The only parameter to be defined by the user is the time needed to release an object to the sink (cycle time; this is done in area 4) by means of the already known distribution functions.

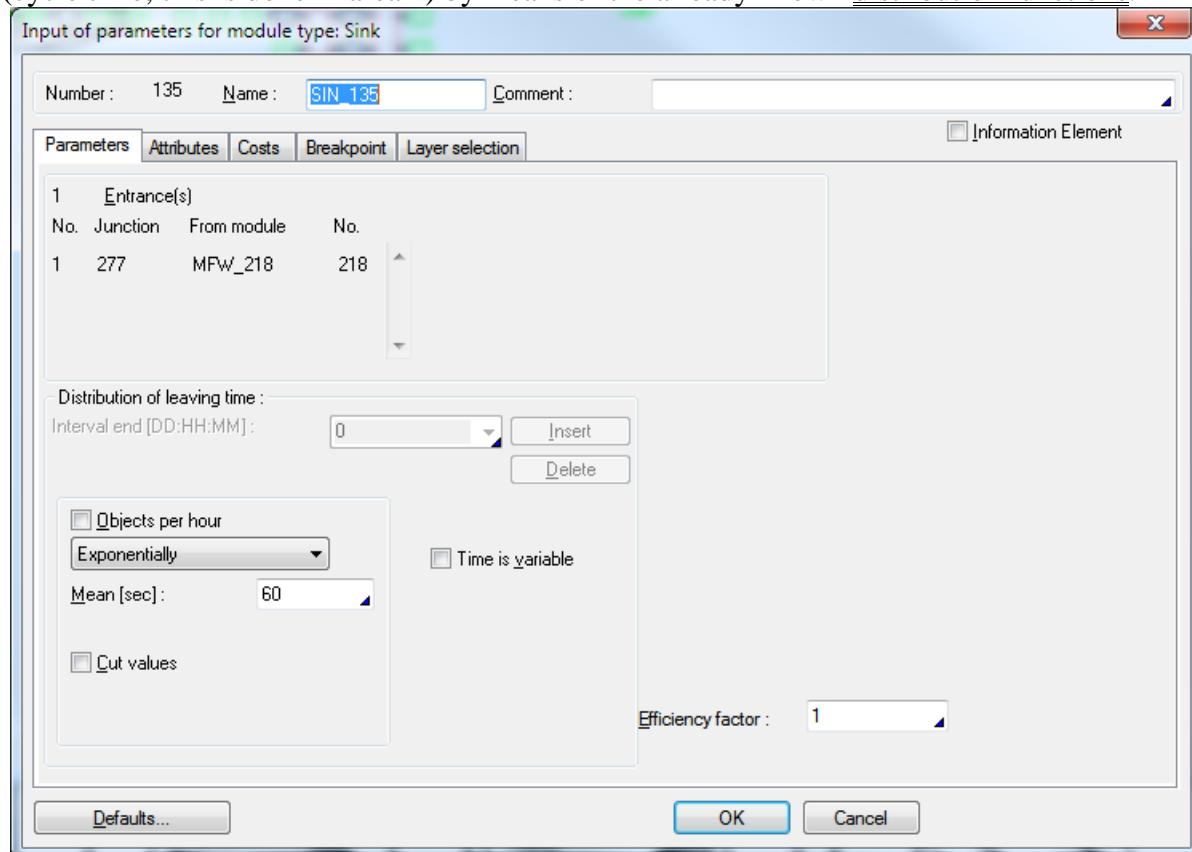


Figure 4.9: Parameters of a sink

As the capacity of a sink is only one object, another object can only enter the sink after the cycle time for the sink has elapsed.

The histogram-distributed loading of object, offered for sinks as an additional strategy, works in the same way as the source. Loading of the next object is delayed until the time defined has elapsed.



4.3 Accumulation Conveyor

Another type of conveying section is called **ACCUMULATION CONVEYOR** and it differs from the previously described section in that objects can run into one another here. In this way, continuous conveyance modules where buffer sections can be employed (e.g. multi-drive rollers) or collision-controlled ESB or transport systems are modeled.

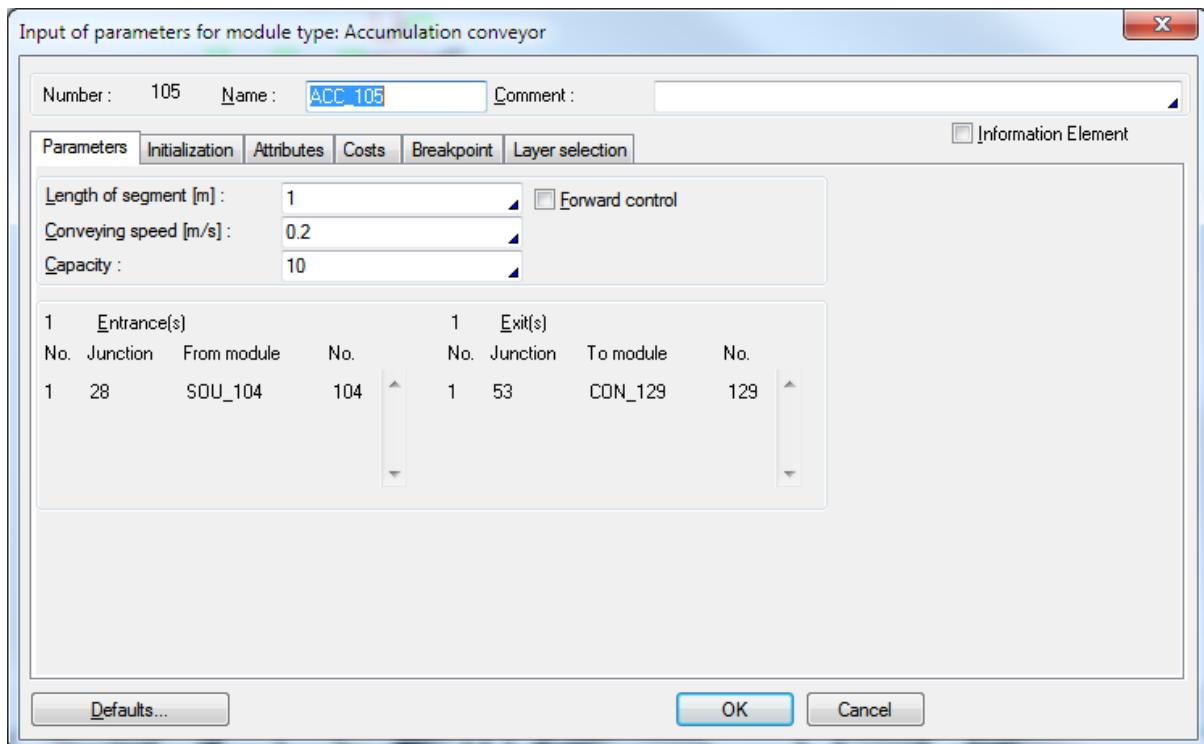


Figure 4.10: Parameters of an accumulation conveyor

An accumulation conveyor has an entrance and an exit and is divided into segments. The decisive parameters therefore are the number of segments and the length of the segments in a buffer.

There is the additional possibility that, before simulation, the buffer section can be initialized with object types in order to shorten the simulation run or to equip vehicle stations with vehicles.



This occurs in the section **Initialization**. The default setting for this option **The module is initialized** is set to disabled, but by clicking the button it can be changed. The usage of the initialization can also be disabled by the switch **Use initialization**.

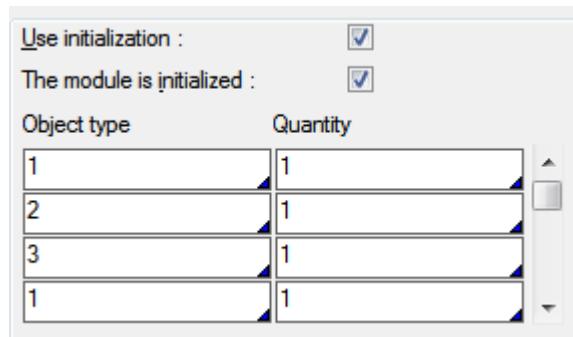


Figure 4.11: Initializing an accumulation conveyor

The following figure shows the initialization of a buffer section with five objects.

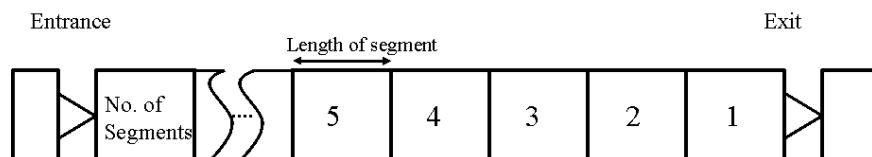


Figure 4.12: Segments

An object entering a buffer causes the junction at the entrance to open if the last buffer segment of the module is unoccupied. Then the object is transported up to the front-most unoccupied buffer segment. When the foremost object can be passed on to the succeeding module all the other objects waiting move forward by one buffer segment.



4.4 Accumulation Conveyor2

Another kind of conveyor is the **Accumulation Conveyor2**. One property is that objects can run into one another here. In this way, continuous conveyance modules where buffer sections can be employed (e.g. multi-drive rollers) or collision-controlled ESB or transport systems are modeled. In addition to the accumulation conveyor, there are two further parameters for defining accumulation sections with arbitrary length.

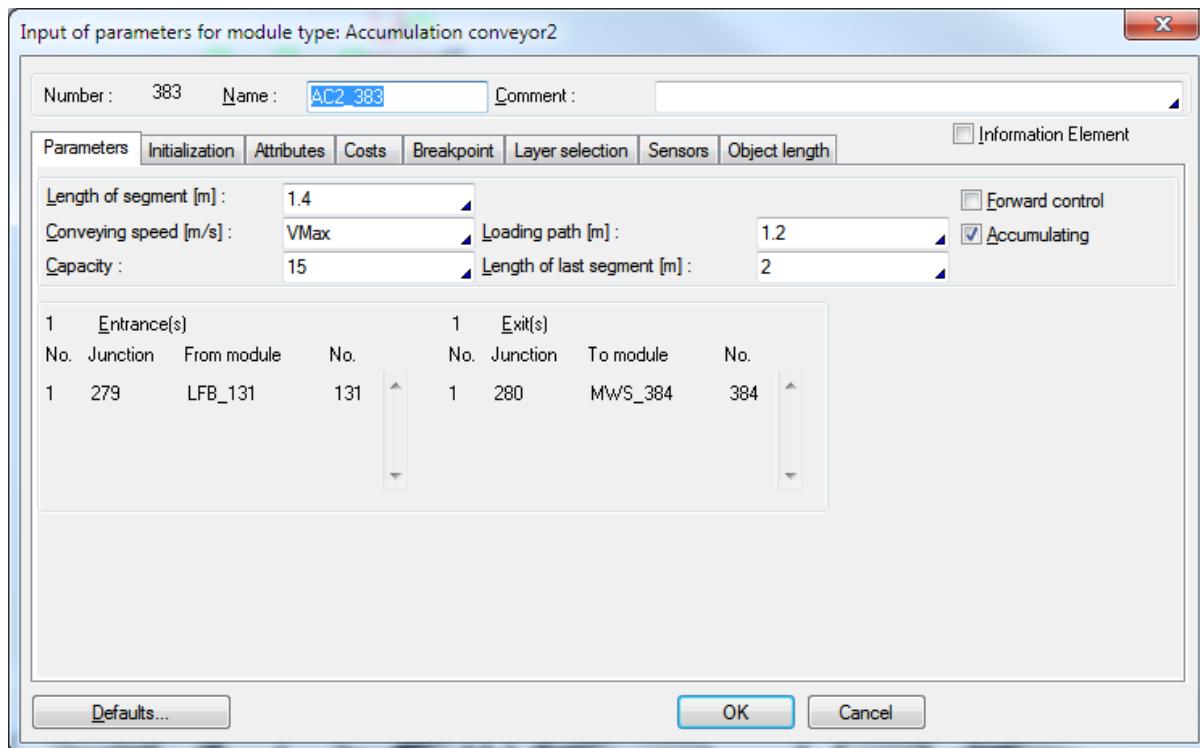


Figure 4.13: Parameters of the accumulation conveyor2

The main terms can be found in the following diagram.

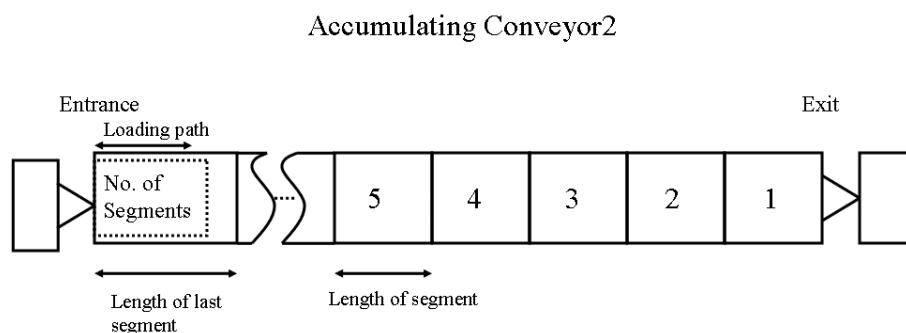


Figure 4.14: Terms of the accumulation conveyor2

An accumulation conveyor2 possesses all parameters of the accumulation conveyor. In addition, the length of the last segment can be determined separately.



Thus accumulation sections can be defined, which are no longer a multiple of the segment length. Furthermore it can be specified with the loading path, how long it takes to bring in the object. After the loading time has passed, the entrance junction is released.

Accumulation conveyors 2 can be initialized. For further information see section initialization.

Accumulation conveyors 2 can be provided with sensors. These correspond to light barriers, which activate a control process in reality. Using a sensor in the model produces the effect of a decision table, which can be reacted upon by the functions of the decision tables.

4.4.1 Sensors

In the accumulation conveyor2, sensors can be defined. The input takes place from the top to the end of the module. The input takes place at a distance from the top in meters. The highest entry corresponds to the first sensor.

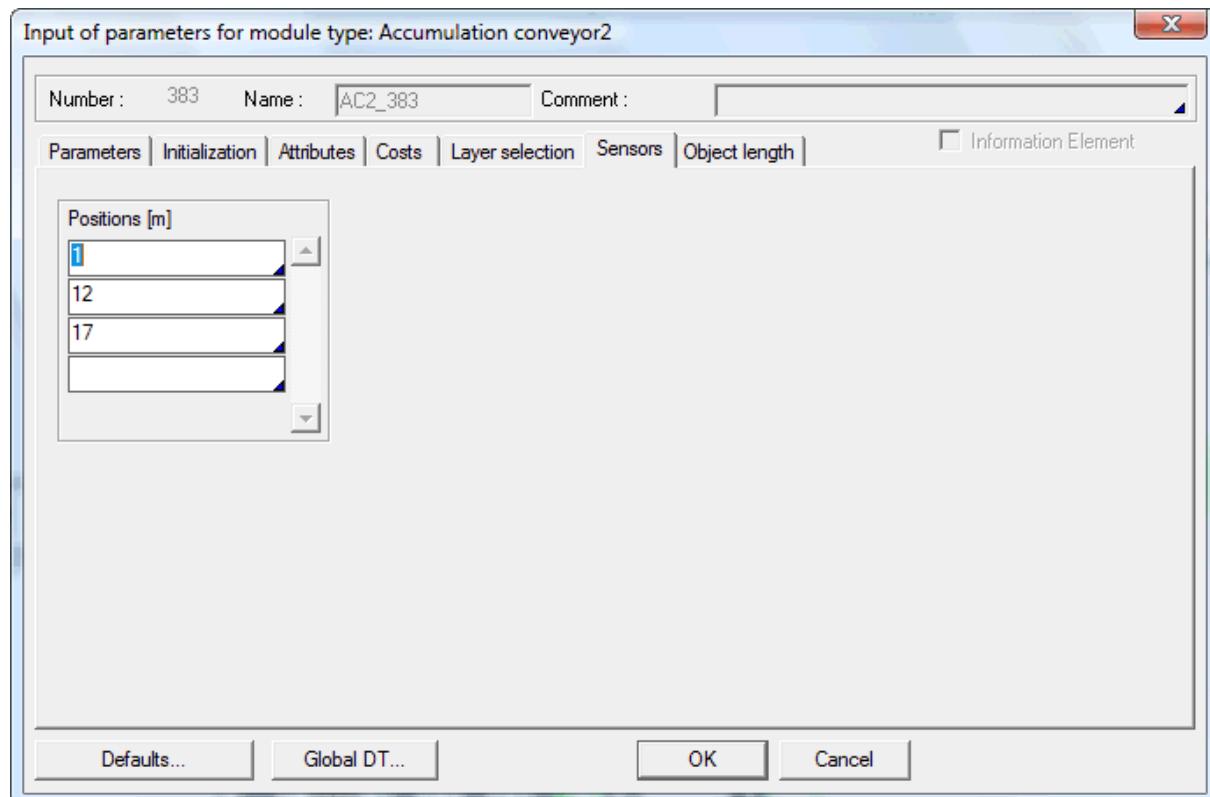


Figure 4.15: Parameters of sensors



4.4.2 Object Length

the accumulation conveyor2, objects with individual object length can be managed. For this, the appropriate length must be entered for each type of object. Wildcards "*" are possible.

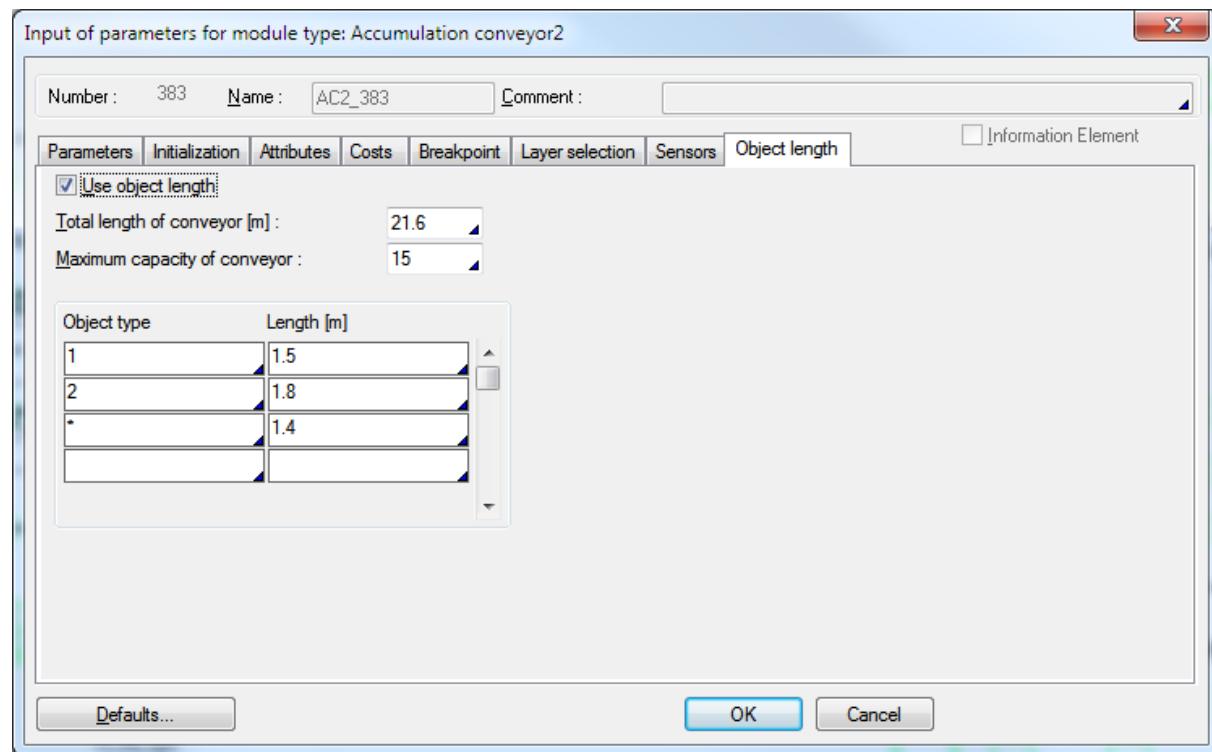


Figure 4.16: Parameters of object length

Apart from the main data parameters, a **Maximum capacity of the conveyor** is to be specified. Additionally the use of the object lengths requires input of the **Total length of the conveyor**. This may be derived from the length of the main parameters,. The use of the object lengths can be switched off by the switch **Use object lengths**.



4.5 Conveying Section

Continuous conveying systems have conveying sections, where the objects do not come into contact with each other. Examples include conveyor belts, ribbon conveyors and overhead chain conveyors. These types of sections are shown as **CONVEYING SECTIONS**.

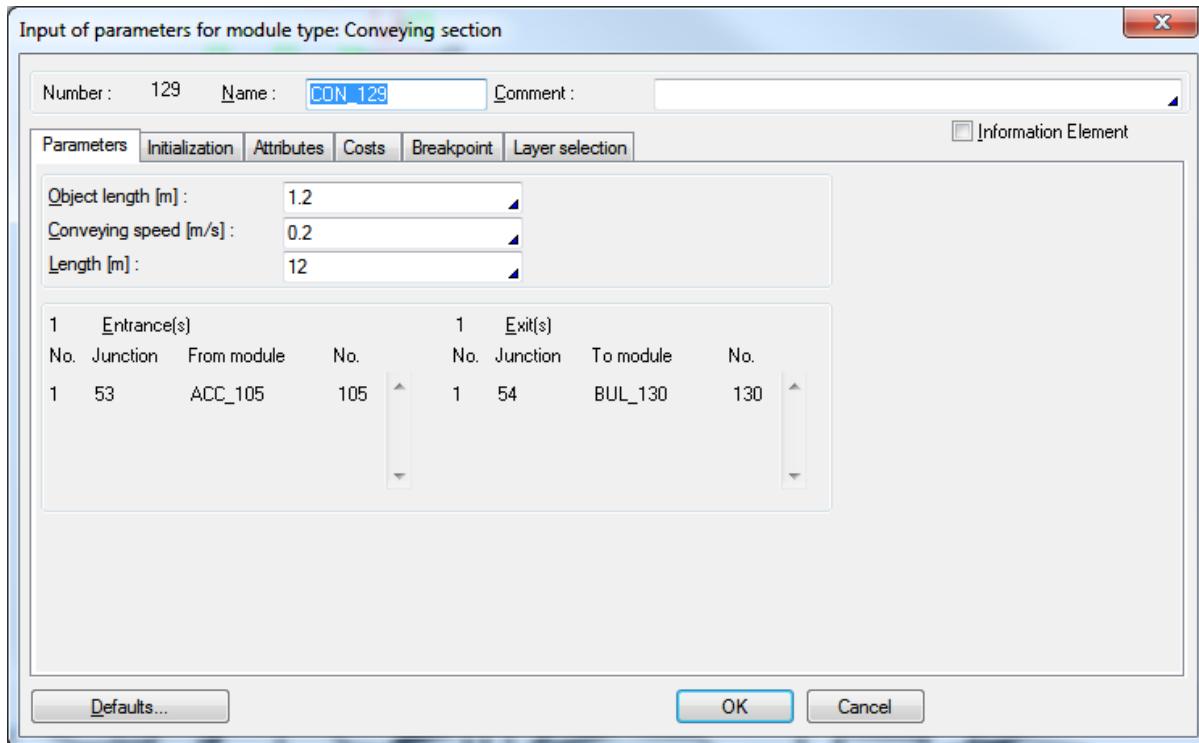


Figure 4.17: Parameters of a conveying section

A conveying section has an entrance and an exit. Its capacity is determined by **Length (of section)** and **Length of (transported) object**, where the length of the object is the same for all object types.

Conveying sections can be initialized. For further information see section [initialization](#).

An object enters the conveying section at the entrance and occupies this until the whole length has gone in. Afterwards, the link at the entrance is released and the object is transported to the exit whereby the distance to its predecessor is constant. When the object reaches the exit a check is made to see if it can proceed to the next module. If this is possible, it is sent to the next section without delay.

If movement to the next module cannot be continued, the whole of the conveying section is halted. During the waiting time, when the section is at a standstill, no object is conveyed. Even an object blocking the entrance of the conveying section (and by doing so, the exit of the preceding module) does not release the corresponding link until a complete entry has been made following a re-starting of the conveying section.



4.6 Bulk Section

A special type of buffer section is the **BULK SECTION**. Although it behaves in the same way as the buffer section concerning conveyance, pick-up and delivery of objects, it is limited by a strategy.

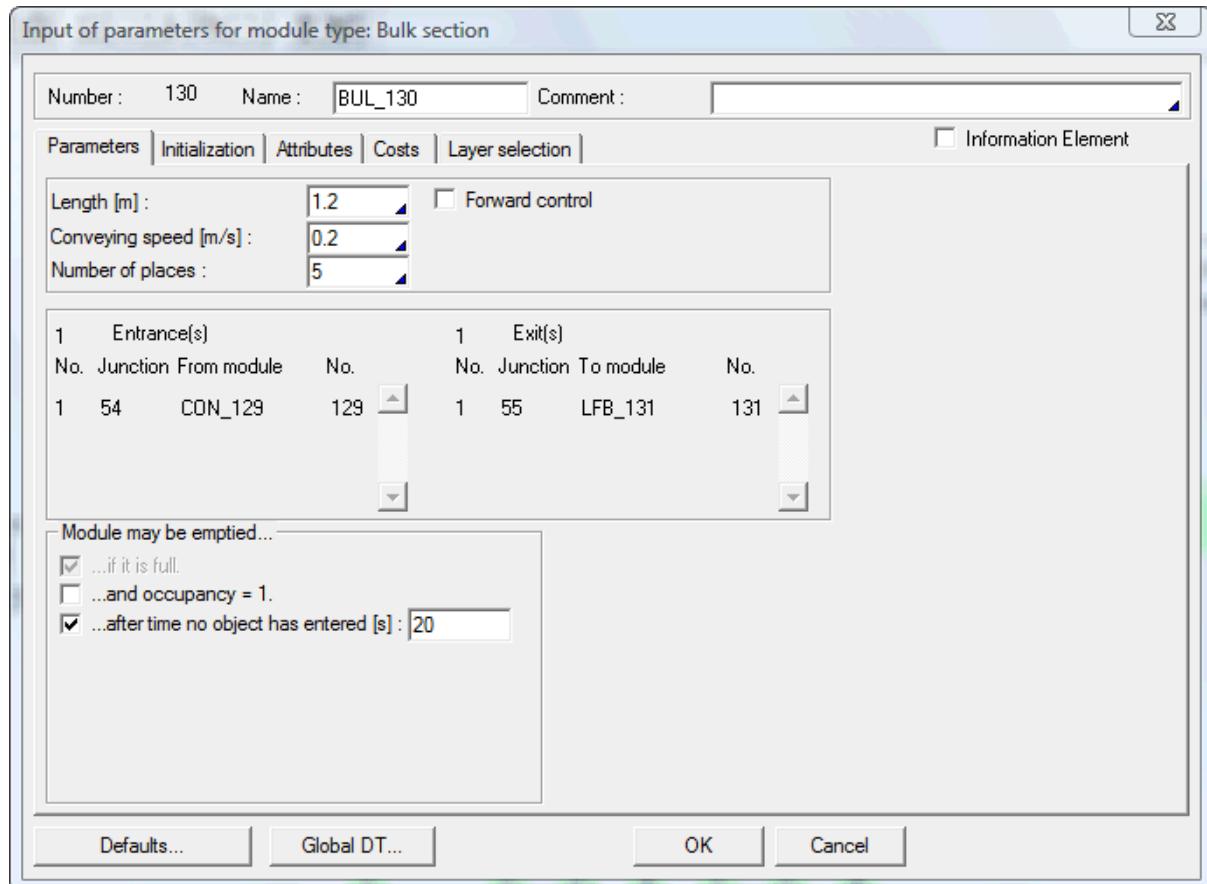


Figure 4.18: Parameters of a bulk section

Bulk sections can be initialized. For further information see section [initialization](#).

The purpose of a bulk section is to accumulate a bulk of objects. This type of bulk formation is found, for example, in a palletizing area for the accumulation of complete pallet loads or can be employed for pre-sorting at a work area where as many types of objects as possible are to be processed one after another (example: in a paint plant).



For a bulk section, both the number of the positions in the route and the total length of the route are to be indicated. Furthermore, the strategy for emptying the bulk section is selected. There are three possibilities for the user to choose from:

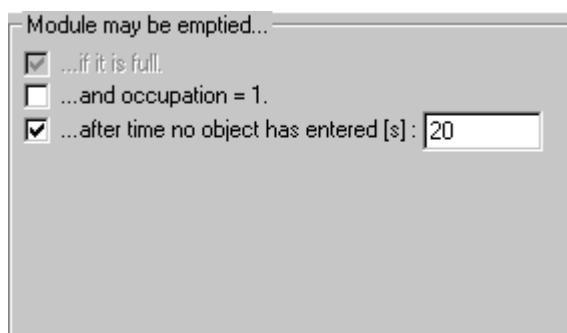


Figure 4.19: Evacuation of a bulk section

- 1.) In general, a bulk of objects is passed on to its successor only when all the buffer positions are occupied (**if it is full**). While a bulk of objects is leaving and even when the exit is blocked at that moment, if for example a basic object has not yet entered an assembly module coupled to the bulk section, no other object can enter the bulk section. Entering is possible as soon as the last object of a bulk section has left the module completely (the object has entered the successor completely).

This standard delivery strategy can be altered in two ways or can be supplemented with point 3):

- 2) single objects can leave the bulk section. This means the object can leave the sections when it has reached the end of the section and no further object has entered the section.

or

- 3) an incomplete bulk of objects can already leave if no other object has entered the bulk section within a defined period since the last entry

If point 1) and point 3) have been selected simultaneously, clearing follows either when the bulk section is full, or as described in 3).

The purpose of these additional strategies is to avoid blocking a section by unnecessarily long waiting periods in special cases.



4.7 LIFO Buffer

The abbreviation 'LIFO' means *last-in, first-out*. The object last entering the buffer is the first to leave it. This has the following consequences for transport behavior in material flow systems: The buffer has room for a certain number of objects which are determined in the parameter dialog by the object length and the conveying speed.

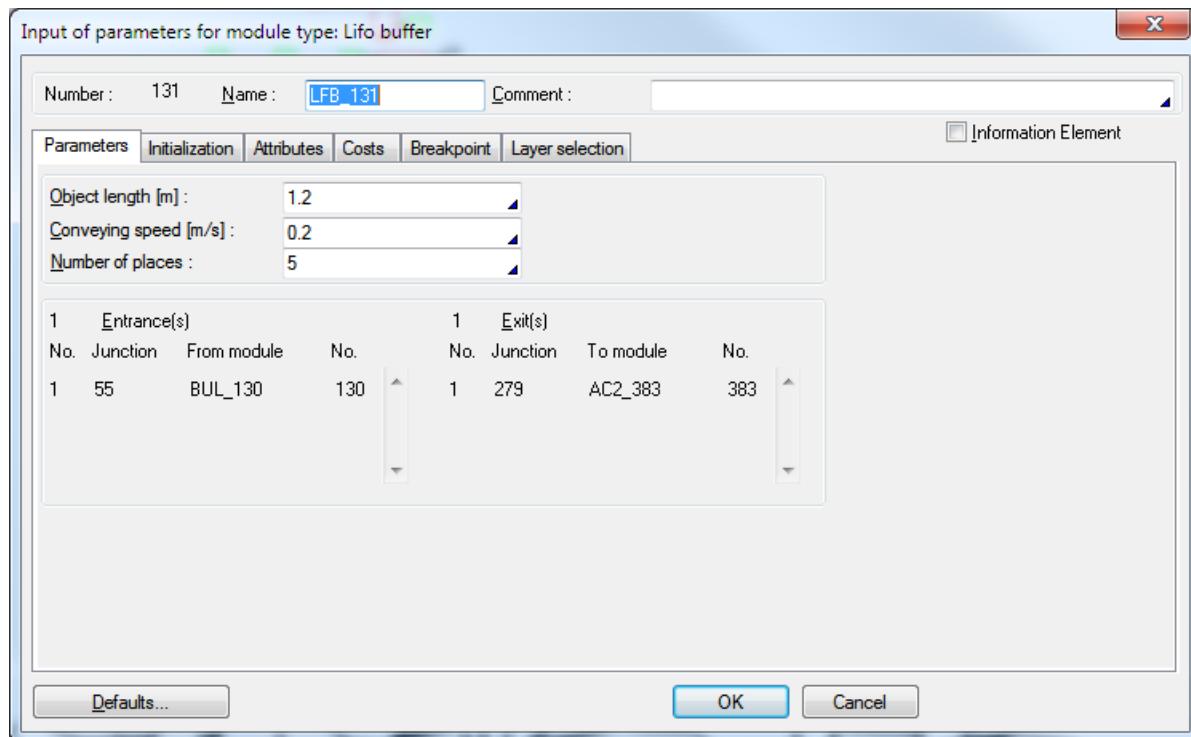


Figure 4.20: Parameters of the LIFO-Buffer

LIFO buffers can be initialized. For further information see section [initialization](#).

An object entering the **LIFO-BUFFER** first occupies the first segment (the first place). If it is possible for the journey to be continued without hindrance into the succeeding module of a LIFO buffer, the object in the first segment leaves the buffer immediately and enters the next module. If immediate further transport is not possible, the object remains on the first buffer place until another object wants to enter it. In this case, the object in the buffer enters the next place, thus making room for the new object entering. This procedure is continued until the LIFO buffer is full or the succeeding module is ready to accept objects. In the second case, the object last entering, i.e. the one on the front position enters the succeeding module first- as the name indicates.



4.8 Storage

The storage module is used for interim storing of objects. It consists of any number of storage areas working independently of one another, each of which has a separate entrance and exit. The number of entrances to be given during the positioning of the module is, at the same time the number of areas and consequently, also the number of exits. The areas are not linked with one another.

Any number of storage modules can be used in one model as is the case with all the other modules.

The following dialog appears when setting the parameters (three entrances have been selected):

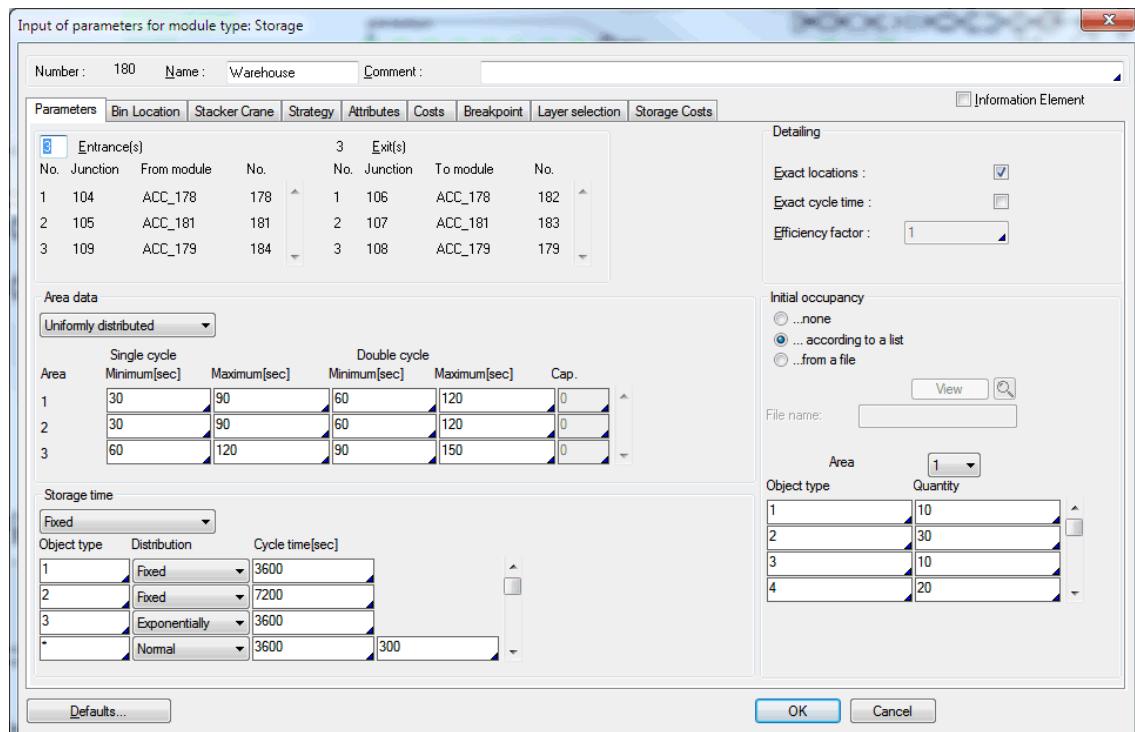


Figure 4.21 Parameters of the storage module

The number of entrances and exits can be read in the parameter dialog. This number can be altered in any way by clicking the input window. Furthermore, the junction numbers and names of the modules linked to the storage can be read.



4.8.1 Standard Area Data

The next is entering the standard area data.

Area	Single cycle		Dual cycle		Cap.
	Minimum[sec]	Maximum[sec]	Minimum[sec]	Maximum[sec]	
1	30	90	60	120	250
2	30	90	60	120	250
3	60	120	90	150	100

Figure 4.22: Standard area data

Here the user first selects one of the five distribution functions described in the chapter *Distribution Strategies*. Depending on the distribution function selected, the appearance of the input mask changes. In addition, the modeler must enter the cycle time for **single cycle** and **double cycle**. The difference between the two is as follows:

The storage is divided into independent areas, as already explained. Each of these areas has a material entrance and a material exit as links to the other modules and has internal entrances and exits to which objects for storing can be delivered or requested objects can be picked up. The mode of operation of an area is comparable to that of a shuttle.

Incoming objects arriving at the material entrance are distributed to various internal exits of the area and correspondingly objects to be picked up are transported from the internal entrances to the material exit.

A single cycle means the procedure where the shuttle carries out two movements. For example: the shuttle drives to the material entrance to fetch an object to be stored and returns afterwards with this object to the storage area. An analogous case is also conceivable for the requisition of materials: the 'shuttle' brings an object from the storage area to the material exit and returns to the storage area to fetch the next one.

In a double cycle the shuttle carries out an additional movement.

For example, an object is conveyed from the material entrance to the storage area and is stored there. The shuttle however, does not remain stationary but fetches an object waiting in the storage area and brings it to the material exit.

If a conveying order is on-hand, a double cycle is always given preference over a single one!

The distribution function entered by the user applies for all areas of the storage area. The parameters entered in this menu for the cycle duration are valid for all areas. However each one can be individually changed in the menu for each storage area:

Apart from the cycle times of the shuttles each area possesses a capacity. This is the maximum number of objects, which can be stored there. The size of the area capacity is unlimited.



4.8.2 Detailing

In the area of detail can be specified whether the warehouse space should be mapped exactly.

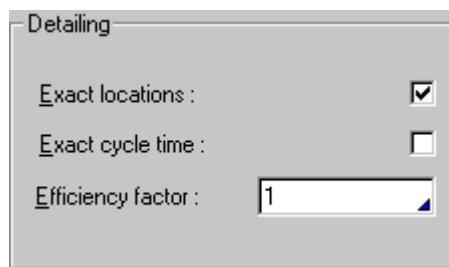


Figure 4.23: Detailing

If **Exact locations** is activated, the storage bins are individually managed. This requires the allocation of individual storage areas in [Bin Locations](#).

If **Exact cycle time** is activated, the time for imports and outsourcing is considered in detail. This requires the parameters of the stacker cranes of the storage areas at [stacker cranes](#). Condition is the activation of *Exact location*.

In field **Efficiency factor** the operating time can be increased or reduced. A factor of 2, for example, means that the operating units are working twice as fast as indicated. This performance factor relates both to the exact cycle times as well as the parameters from the standard area data.

4.8.3 Initializing

With the initial allocation the storage can be pre-allocated with objects. When **The store is initialized** is deactivated, then all areas of the storage are empty at the beginning of simulation. Otherwise the user has the possibility of determining for each area the object types and their individual quantity.

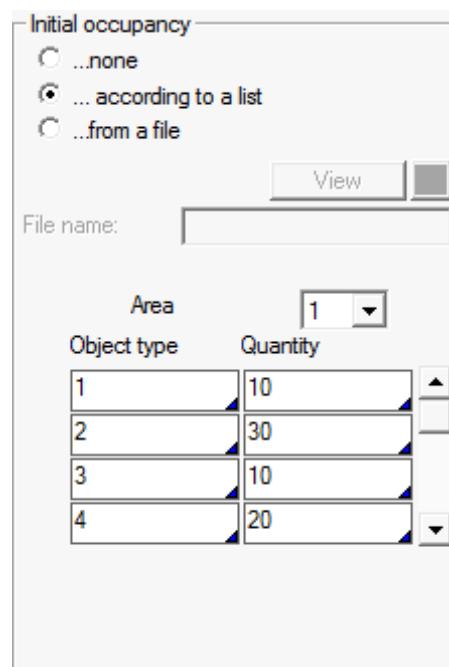


Figure 4.24: Initializing of the storage



In the case, in which the generation data is taken **from a file**, only the name of the file is to be defined. The storage area, the object types and the number of objects, which are produced, is to be specified in the file.

With the button **view** the file can be shown. A change is not possible thereby.

A file, from which generation data is to be read, must have the following format:

```
COM <Comment>           will be ignored
INI                         start of data
<area> <object type> <number> [optional: attribute]
```

Figure 4.25: generation file

The keyword COM is optional and can be used several times. The keyword INI is mandatory. The following line only contains data.

Example 1:

```
COM This is an example file
COM for initializing of objects
INI
1 1 120
1 2 50
2 1 20
2 2 30
2 3 40
```

The formats of the optional attributes can take from the chapter attributes from file of the source. **Storage Time**

The mean storage time of the objects in stocks can be indicated specific to object type. For this, a distribution and the appropriate parameters are to be indicated to the distribution. The objects of the types, which are found in the list, are then announced automatically after the appropriate time for taking out. Alternatively a wildcard (*) can be used. This means that all objects, which were not found explicitly in the list, are treated according to these parameters.

Storage time			
Object type	Distribution	Cycle time[sec]	
1	Fixed	3600	
2	Fixed	7200	
3	Exponentially dis	3600	
*	Normally distribu	3600	300

Figure 4.26: Parameters of retention time

Object types, which were not found in the list, can be brought by orders, which must be produced through decision tables, back into the system. Taking out objects by decision table



has a higher priority than the automatic retrieval. This means that an object is being outsourced by decision table in any case, even if the residence time of a later date is provided.

Initialized objects with a storage time will also be swapped out automatically. The storage time is, however, randomly between 0 and the mean storage time.

4.8.5 Delivery Strategy

The delivery takes place with the assistance of orders, which must be produced with decision tables. If several jobs are active, it is determined by means of the external storage strategy, which of the jobs is activated. If an order for a type of object is produced, which is not in storage, then this remains in the queue, until it can be fulfilled (i.e. an object of the appropriate type of storage). The **delivery strategy** is selected in the parameter dialog:

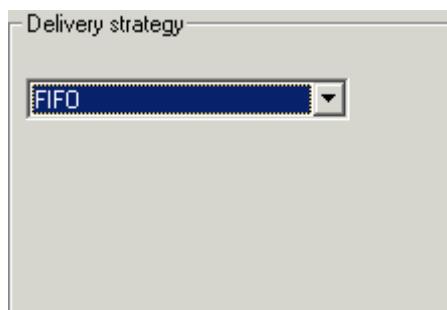


Figure 4.27: Selection of a delivery strategy

If a certain object is to be requisitioned from a storage but several areas contain this object type, this strategy selects the area to be next used for delivery. See chapter **Right-of-way Strategies** for explanations of the strategies **FIFO**, Maximum **relative occupancy** and **Maximum absolute occupancy**.

Priority of entrances selects from all areas of the storage those with the highest priority. Entrance means at least the area corresponding to this entrance. These priorities are those with positive integer values. The lower the value, the higher the priority is.

Random selects via the random generator an area of storage. If there are several of these types in an area from which a certain object is to be requisitioned, a second strategy automatically selects the object with the longest storage time.

If a delivery order is placed and there is no object of the relevant type in the storage, this order is stored for the time being. As soon as in the course of the simulation an object of the same type with a storage order reaches the storage modules entrance then this storage order is cancelled and the object is sent on to the storage exit.



4.8.6 Bin Location

In the dialog **Bin Location** the coordinates of the bin locations can be described simplified.

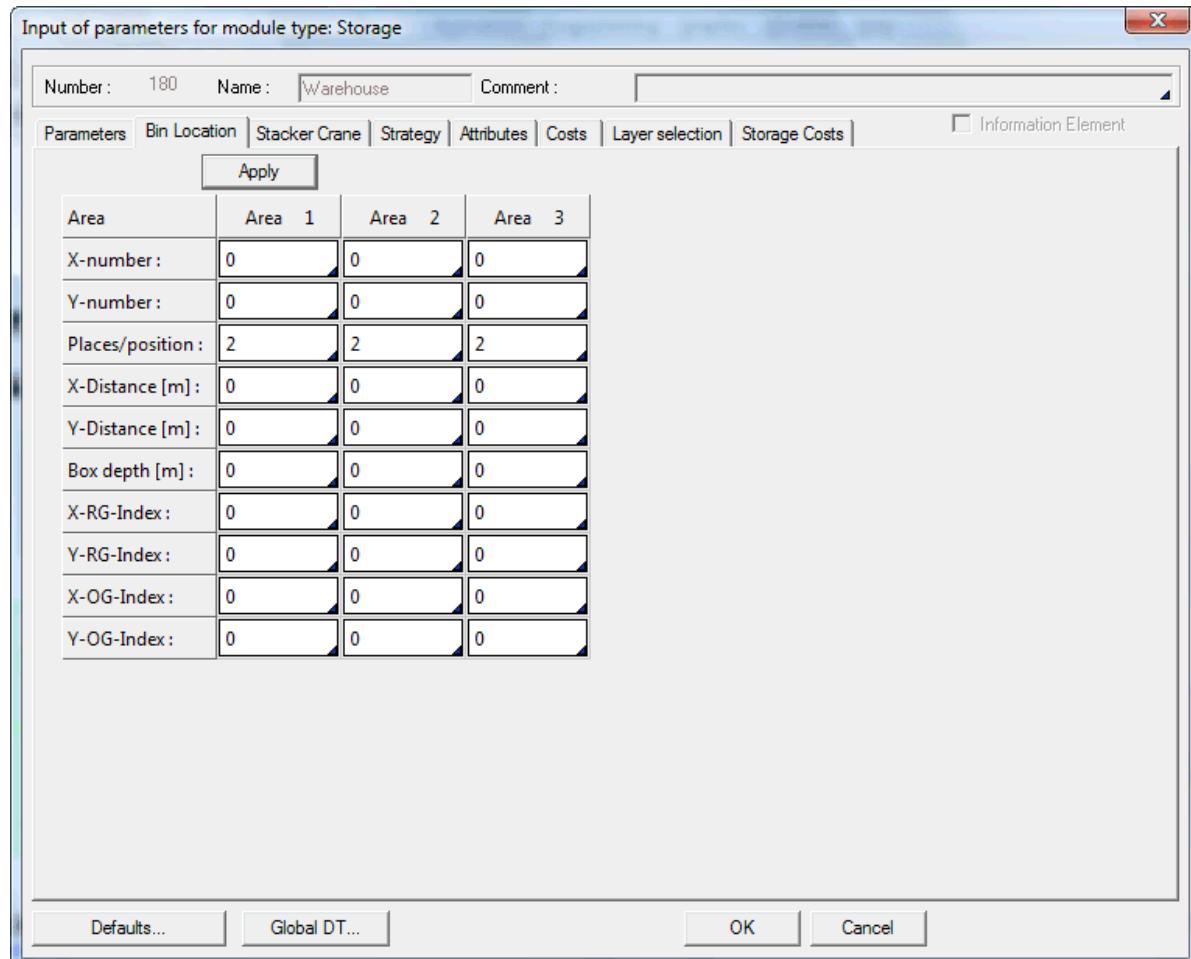


Figure 4.28: Parameter exact location

With the help of the values of **X-number** and **Y-numbers** defines, how many places exist in the direction of X-and Y-direction in a storage area. The parameter **places/position** indicates how many bin locations are at one delivery point. These are usually 2, because at every position left and right of the stacker crane is storage bin. The **X-Distance** and the **Y-Distance** defines how far apart the individual subjects away. It should be measured between the centers of the locations (including the ligament). The **Box depth** determines how long the exit way is that the stacker crane is required to deliver a load to a bin location.

The **WE/WA-indices** determine what position the entrance and exit point is located. Bottom/Left the index is 1/1, Top/Right the index is 50/20. 0/1 means that the entrance point is the storage before the first bin location is located. Each storage has exactly one entrance point and one exit point. If all areas have the same parameters, the parameters of the first division by pressing the button **Apply** to the other sectors.

In accordance with the storage strategy a place in the storage is assigned to an object, which registers at the point of storage (incoming goods).

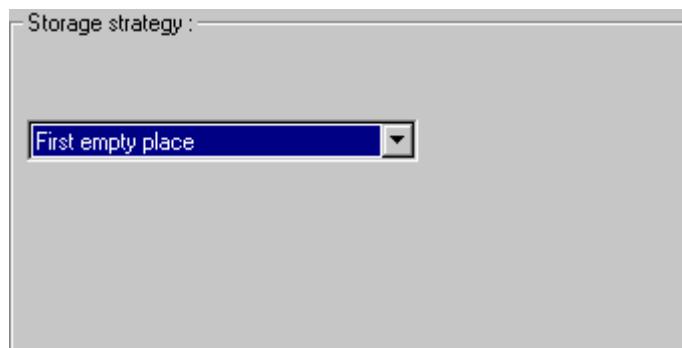


Figure 4.29: Parameter of storage strategy

At present only the strategy **first empty space** exists. This means, it is selected the free bin location, which is next to the point of storage. This strategy is evaluated, if the object registers at the stock area. For the distribution of the objects on the individual storage areas there is the stock control

4.8.7 Stacker Crane

In the sub dialog **Stacker Crane** the parameters of the stacker cranes can be entered. These correspond to those of the shuttle, so that further information is to be inferred from there.

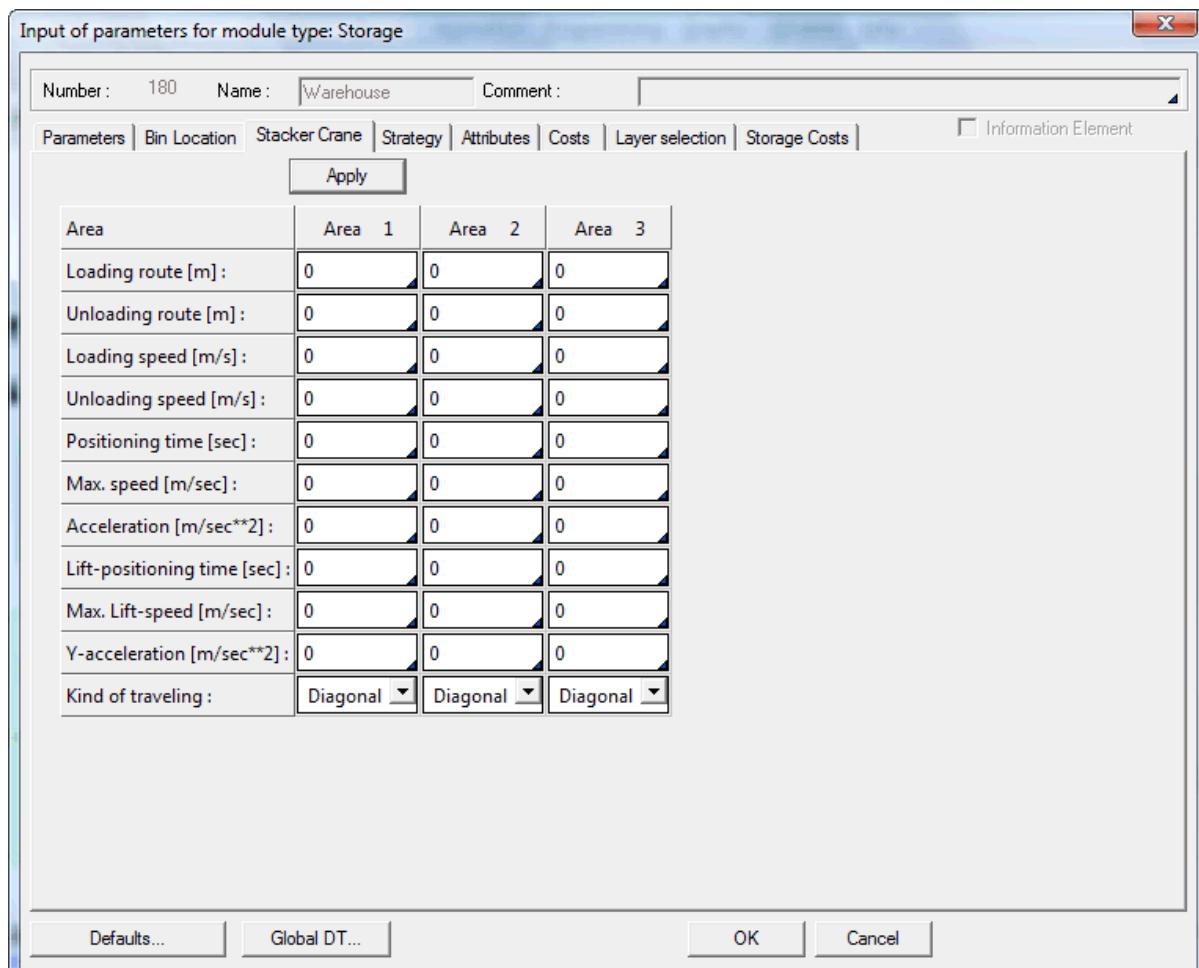


Figure 4.30: Parameter of stacker cranes



If all stacker cranes possess the same parameters, then the parameters of the first STC's can be transferred to the other control devices by the pressing of the button **Apply**. The stacker cranes can be evaluated separately statistically.

4.8.8 Storage Cost

In the storage, operating costs can be indicated. These are permanently carried out. They are distributed with each stock removal in equal parts on all objects, which are in the storage.

Object type	Fixed costs [mu/pce]	Variable [mu/s]
*	10	30

Figure 4.31: Parameters of storage costs

Additionally storage costs for each type of object can be indicated. The storage costs are a combination of a fixed portion, which results to one half during the storage and to the other half with removal. Furthermore a variable portion exists. This is calculated and assigned to the object with its removal.



4.8.9 Storage Status Report

In online - simulation the allocation of a storage can be visualized in detail. For this can over context - menu of the storage module (*menu/module/storage status report*) a three-dimensional representation of the storage to be activated. This representation is available only in the 3D-Version von Dosimis-3. Navigation takes place exactly the same as in the [3D-View](#) of the model.

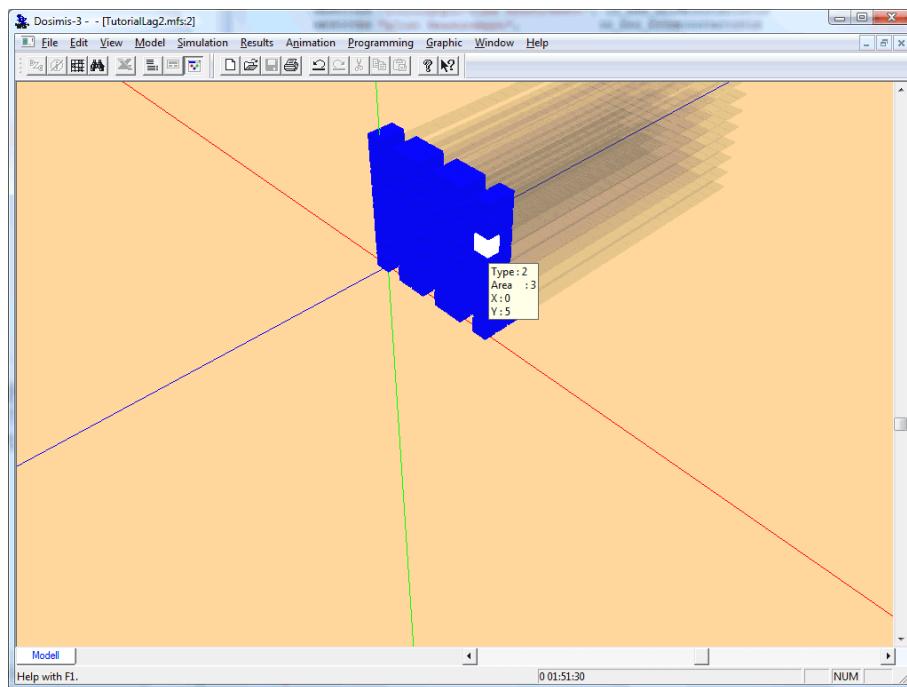


Figure 4.32: Storage status report

In this representation each individual place of the storage can be regarded. By tooltip one receives further information to the stored object. Additionally the places reserved for the storage as well as the objects planned for the paging are colored.

Over the context menu of the window a filter can be activated, over which further information can be called up.

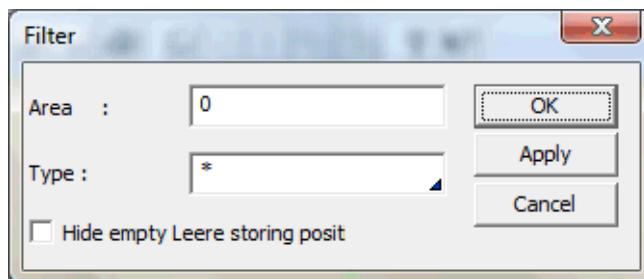


Figure 4.33: Filter of the storage status report



If in the field **area** contains a value not equal to zero, only this area is drawn. In the field **type** beside the Wildcard ('*') also explicitly a type of object can be indicated. Thus one can regard the distribution of these objects over the storage.

NOTE: The actualization of the storage is very time intensive. Therefore this should be opened only during an interruption of the online simulation and be closed again after the analysis.

4.9 Workstation

In almost all material flow systems and especially in manufacturing plants, objects are processed and changed. For example, loaded pallets are banded, storage crates requested, vehicle bodies paint-sprayed etc. A module type identified as **WORKSTATION** is required to illustrate this object processing.

A work station has an entrance and an exit and can accept one object only. Processing takes place on a buffer position where the object remains for the process duration. After the loading time, it is checked whether the old object left the station completely. Only if both conditions are satisfied can the handling can be started. This check does not take place, if a length of 0 is defined. An object entering releases the link at the entrance when it is completely on its work place. When work has finished, it is passed on to the succeeding module, if this is not occupied.

Figure 4.34 appears when setting parameters for a work station:

Input of parameters for module type: Workstation

Number : 106	Name : WST_106	Comment :	<input type="checkbox"/> Information Element																																																											
Parameters		Transport system	Attributes	Costs	Breakpoint	Layer selection																																																								
Length [m] : 1	Speed [m/sec] : 0.2	<input type="checkbox"/> Forward control	Efficiency factor : 1																																																											
<table border="1"> <tr> <td colspan="2">1 Entrance(s)</td> <td colspan="2">1 Exit(s)</td> <td colspan="3">Transport parameters :</td> </tr> <tr> <td>No.</td> <td>Junction</td> <td>From module</td> <td>No.</td> <td>No.</td> <td>Junction</td> <td>To module</td> <td>No.</td> </tr> <tr> <td>1</td> <td>281</td> <td>MWS_384</td> <td>384</td> <td>1</td> <td>282</td> <td>ASS_8</td> <td>109</td> </tr> </table>					1 Entrance(s)		1 Exit(s)		Transport parameters :			No.	Junction	From module	No.	No.	Junction	To module	No.	1	281	MWS_384	384	1	282	ASS_8	109	<input type="checkbox"/> Worker <input type="checkbox"/> Transport parameters <input type="checkbox"/> Work procedures <input type="checkbox"/> Set-up <input type="checkbox"/> New object type																																		
1 Entrance(s)		1 Exit(s)		Transport parameters :																																																										
No.	Junction	From module	No.	No.	Junction	To module	No.																																																							
1	281	MWS_384	384	1	282	ASS_8	109																																																							
Work procedures Name : ASS_1 No. : 1/1 <input type="button" value="New"/> <input type="button" value="Delete"/> Comment :					<input type="checkbox"/> Consider all work procedures																																																									
Distribution of working time <table border="1"> <tr> <th colspan="2">Object type</th> <th colspan="2">Distribution</th> <th>Mean[sec]</th> <th>Deviation[sec]</th> <th>Strategy</th> <th>Min.</th> <th>Max.</th> <th>Qual.</th> <th>Intrp.</th> </tr> <tr> <td>1</td> <td>Normal</td> <td>work</td> <td>5</td> <td>Minimal no.</td> <td>1</td> <td>3</td> <td>2</td> <td>10</td> <td></td> <td></td> </tr> <tr> <td>10</td> <td>Normal</td> <td>\$Working</td> <td>5</td> <td>Maximal no.</td> <td>2</td> <td>2</td> <td>1</td> <td>12</td> <td></td> <td></td> </tr> <tr> <td>2</td> <td>Normal</td> <td>=3600/144</td> <td>5</td> <td>Without</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>-</td> <td>Normal</td> <td>80</td> <td>5</td> <td>Maximal no.</td> <td>2</td> <td>3</td> <td>1</td> <td>5</td> <td></td> <td></td> </tr> </table>								Object type		Distribution		Mean[sec]	Deviation[sec]	Strategy	Min.	Max.	Qual.	Intrp.	1	Normal	work	5	Minimal no.	1	3	2	10			10	Normal	\$Working	5	Maximal no.	2	2	1	12			2	Normal	=3600/144	5	Without							-	Normal	80	5	Maximal no.	2	3	1	5		
Object type		Distribution		Mean[sec]	Deviation[sec]	Strategy	Min.	Max.	Qual.	Intrp.																																																				
1	Normal	work	5	Minimal no.	1	3	2	10																																																						
10	Normal	\$Working	5	Maximal no.	2	2	1	12																																																						
2	Normal	=3600/144	5	Without																																																										
-	Normal	80	5	Maximal no.	2	3	1	5																																																						
Set-up times <input type="checkbox"/> Setup only if object in main list					Employment of workers <table border="1"> <tr> <th>From object</th> <th>To object</th> <th>Distribution</th> <th>Cycle time[sec]</th> <th>Strategy</th> <th>Min.</th> <th>Max.</th> <th>Qual.</th> <th>Intrp.</th> </tr> <tr> <td>1</td> <td>10</td> <td>Fixed</td> <td>setup</td> <td>Maximal no.</td> <td>1</td> <td>3</td> <td>4</td> <td>10</td> </tr> <tr> <td>10</td> <td>1</td> <td>Fixed</td> <td>setup</td> <td>Without</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td>*</td> <td>Fixed</td> <td>60</td> <td>Maximal no.</td> <td>1</td> <td>2</td> <td>2</td> <td>5</td> </tr> <tr> <td>-</td> <td>*</td> <td>Fixed</td> <td>60</td> <td>Maximal no.</td> <td>1</td> <td>3</td> <td>4</td> <td>10</td> </tr> </table>			From object	To object	Distribution	Cycle time[sec]	Strategy	Min.	Max.	Qual.	Intrp.	1	10	Fixed	setup	Maximal no.	1	3	4	10	10	1	Fixed	setup	Without					2	*	Fixed	60	Maximal no.	1	2	2	5	-	*	Fixed	60	Maximal no.	1	3	4	10	Initial object : 1 Kind of processing : Random New object Probability 1 85 10 15									
From object	To object	Distribution	Cycle time[sec]	Strategy	Min.	Max.	Qual.	Intrp.																																																						
1	10	Fixed	setup	Maximal no.	1	3	4	10																																																						
10	1	Fixed	setup	Without																																																										
2	*	Fixed	60	Maximal no.	1	2	2	5																																																						
-	*	Fixed	60	Maximal no.	1	3	4	10																																																						
<input type="button" value="Defaults..."/>					<input type="button" value="OK"/>			<input type="button" value="Cancel"/>																																																						



Figure 4.34: Parameters of a workstation

The user must determine not only the length [m] but also the conveying speed [m/sec] of the workstation. If required, forward control can also be fixed.

The parameters of the transport system will not be discussed in any detail here as they are a part of the transport system module. **Work Procedure**

In a workstation several work procedures can take place successively. For each of these procedures, a list of parameters can be defined for operating time, set-up times and a list of the new object types. If the object has entered the station, it is checked whether for this object in the first work procedure a handling is parameterized. After the processing, it is switched to the next step and the analysis begins once more. Here however, the type of object is analyzed, which resulted out of the handling in the previous step - if this changed, for example, due to the list of the new types. If for the current object type no work time is defined in a work procedure, the handling is aborted and the object leaves the module.

Object type	Distribution	Mean[sec]	Deviation[sec]	Strategy	Min.	Max.	Qual.	Intrp.
1	Normal	work	5	Minimal no.	1	3	2	10
10	Normal	\$Working	5	Maximal no.	2	2	1	12
2	Normal	=3600/144	5	Without				
*	Normal	80	5	Maximal no.	2	3	1	5

From object	To object	Distribution	Cycle time[sec]
1	10	Fixed	setup
10	1	Fixed	setup
2	*	Fixed	60
*	*	Fixed	60

Figure 4.35: Definition of work procedures

As was already seen in the description of object generation in a source, several situations can be present after an object has been received by the work station.

Case 1) The object has a fixed object type. In this case, checks will be made to see if the object type is in the list to be defined. Therefore, a distribution function is to be selected and, for each type, the working time is to be fixed. If it is listed, then it can be processed in the station. Otherwise, it is immediately transported further.

In the parameter dialog, the user selects an object type from the object list by means of the two upper arrows. It is now possible to define one or several new types which can result from the processing of the object. In order to do this, the user must allocate to each object type in the list a probability number. If only one new object type has been defined, this value is irrelevant and the old type is changed after processing into this new type. On the other hand, if there is a choice of several objects in the list **new object**, a new object type is chosen for the simulation via the random number generator taking into account the defined probability.

Case 2) If a multitask object enters a work station, the list of destinations and/or object types is searched for those object types which can be processed in the module observing the rules of



sequence. The processing times for the objects found are added together and these types are then deleted from the list.

4.9.2 Work Times

In the list of the work times, the object types are specified, for which the processing is permitted in this work procedure. If a **wildcard** (*) is used, all objects, which were not explicitly indicated, are processed with these parameters.

Distribution of working time									Employment of workers		
Object type	Distribution	Mean[sec]	Deviation[sec]	Strategy	Min.	Max.	Qual.	Intrp.			
1	Normal	work	5	Minimal no.	1	3	2	10			
10	Normal	\$Working	5	Maximal no.	2	2	1	12			
2	Normal	=3600/144	5	Without							
*	Normal	80	5	Maximal no.	2	3	1	5			

Figure 4.36: Input of working time

Additionally, for each object type, it must be determined whether or not a worker is required for handling this object. This definition is given in the parameter dialog. In the first two fields the minimal (min) and maximal (max) number of workers required is given. Activity does not begin until the minimal number is available. In the third column the necessary qualification level of the workers is given and in the last column it can be determined whether or not the activity can be interrupted by a higher-order activity. If no workers are necessary to carry out an activity no entries need to be made in the corresponding entry fields (See Chapter Work area for explanation of terminology).

4.9.3 Change of Object Type

According to the procedure, the type of the object can again be defined. For this, a list can indicate probabilities assigned by object types as to each input object. According to the handling, the new object type is randomly created according to the probability.

Initial object :	1
Kind of processing :	Random
New object	Probability
1	85
10	15

Figure 4.37: Change of Object Type

If the list contains only one entry, the input object is changed exactly into this type. If no list has been defined for an object type, it remains the same after processing. Based on the kind of work a selection can be made between *by random*, *bauschuld* and *fixed sequence*. With *fixed sequence* the objects are produced as indicated. The value under probability means that this quantity of objects is produced before the type of object changes. Bauschuld is a deterministic sequence, which is defined by the Bauschuld algorithm (see also source). **Set-up Times**



In case in the real system certain changes - necessary for the processing of this object type - must be made before the processing of a new object type, a set-up matrix can be created to take into account the time required to do this. In the parameter dialog, the time needed to set-up a machine for the processing of a new object type can be defined in a list. The program notes the type which has been processed and has left the module. When another object type reaches the processing station the entries in the list are checked to see if a new set-up is necessary.

The screenshot shows the 'Set-up times' dialog. On the left, there is a table titled 'Set-up times' with columns: 'From object', 'To object', 'Distribution', and 'Cycle time[sec]'. The table contains four rows of data:

From object	To object	Distribution	Cycle time[sec]
1	10	Fixed	setup
10	1	Fixed	setup
2	*	Fixed	60
*	*	Fixed	60

On the right, there is a section titled 'Employment of workers' with a table showing employment strategies for workers. The table has columns: 'Strategy', 'Min.', 'Max.', 'Qual.', and 'Intrp.'.

Strategy	Min.	Max.	Qual.	Intrp.
Maximal no.	1	3	4	10
Without				
Maximal no.	1	2	2	5
Maximal no.	1	3	4	10

Figure 4.38: Input of Set-up times

The input can be done by wildcards. It is first checked whether the combination without wildcard, which can be analyzed, can be found, then with the help of one wildcard and in the last case with two wildcards.

4.9.5 Configuration

Within the area **Configure** optional parameters can be hidden. Thus the dialog has a better structure. The switches **Worker** and **Transport parameters** correspond to those the properties of the user interface.

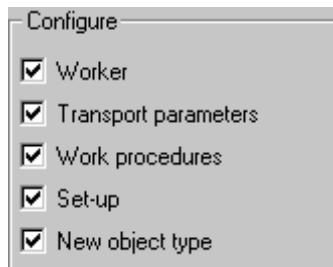


Figure 4.39: Configuration of Parameters

The remaining three alternatives are concerned with special parameters of the station. All parameters can be adjusted individually for each component.



4.10 Multiple Workstation

A special case of the workstation is the **MULTIPLE WORKSTATION**. There several objects can be processed simultaneously. The parameterization of a multiple workstation essentially corresponds to that of the workstation. Additionally however, a capacity is to be indicated, where it can be specified how many objects are in the station for simultaneous processing:

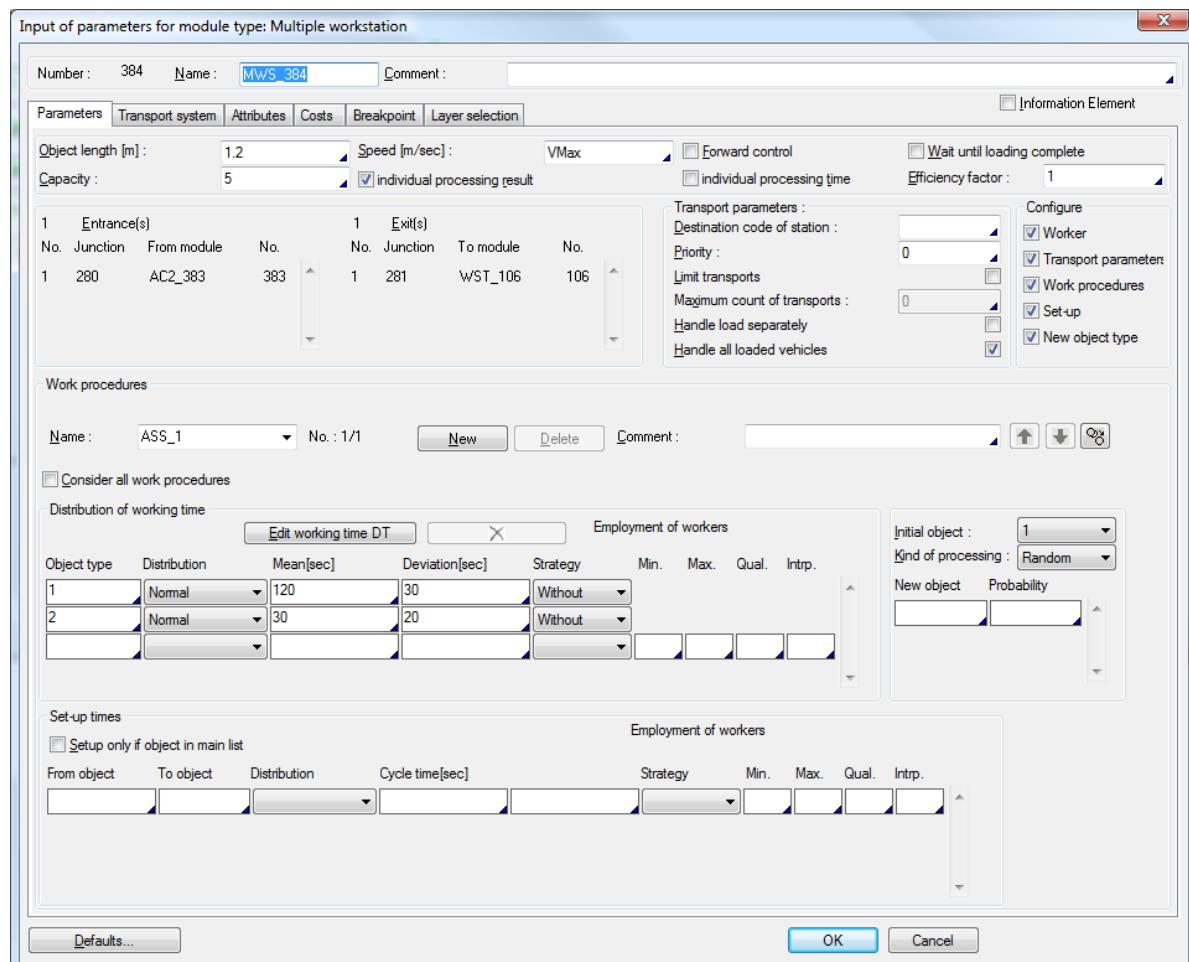


Figure 4.40: Parameters of a multiple workstation

Although an object length of 0 is possible, it makes no sense to do so due to the functional mode of the multiple work station. The duration of the loading process determines whether yet another object can be simultaneously processed. The duration of entering the module for an object is the time, which is needed to cover the object length. The first object must travel, however, the overall length of the station, before processing of this object can start. The overall length of the station is calculated by the product of capacity and object length. If the switch **Individual** is activated, the new object types are calculated individually for each object in the station. Otherwise all objects receive the process result of the first object. When the switch **individual processing result** is activated, for each object in the station its own result is determined. Otherwise all objects receive the result of the first object. When **individual processing time** is active, each object is assigned a different processing time. The duration of the process is the longest time of all objects.



4.10.1 Functionality

Multiple workstations possess as many working places as indicated by the capacity. If now an object travels to the first place, after reaching this position it is examined whether further objects are also arriving. If all objects are now in position, the work will start. Here, the first object in the station determines the operating time. The working starts either when the station is full or if the last object, which enters the station, is in position and no further object is announced. In the last case, it means that some working places are empty. After the operating time is completed, all objects leave the station. Parallel to this, the new objects can enter the station.

4.11 Assembly Station

The **ASSEMBLY STATION** is used to illustrate object mergers which can occur for example in a manufacturing process or from loading of an electro-suspension track. The parameters of an assembly station are set by using the following window. In this example, there are three entries.

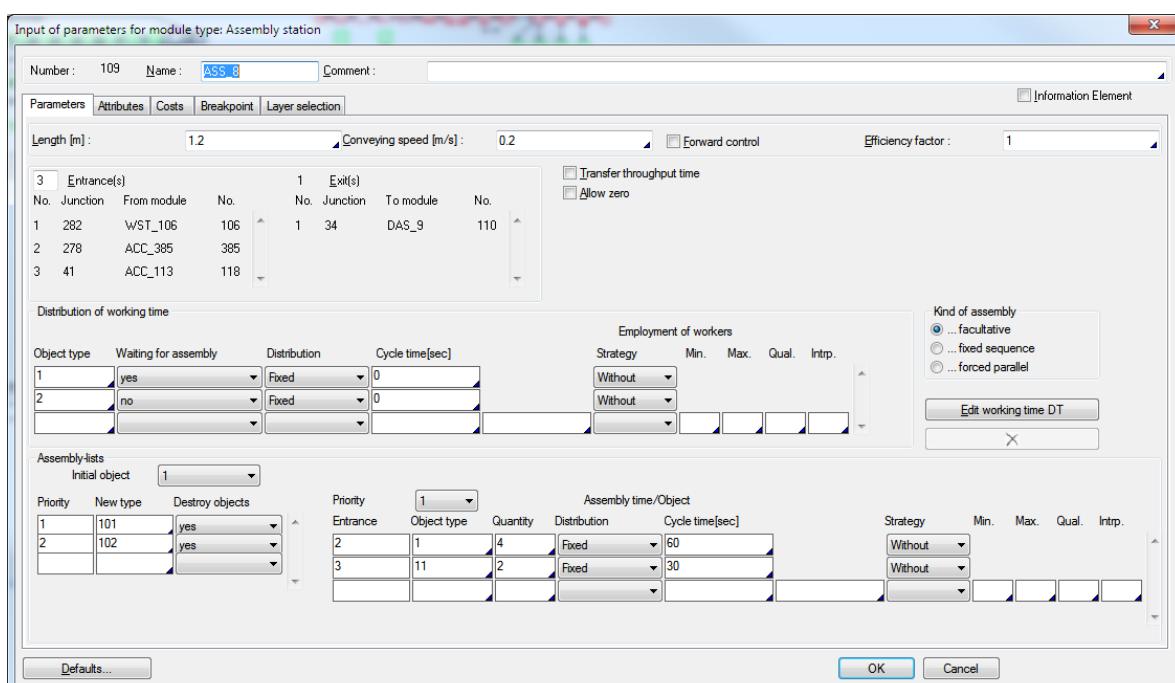


Figure 4.41: Parameters of an assembly station

First, the user can set the forward control. Additionally, the length of the assembly station and the conveying speed must be set.

The parameter dialog has a list of the entrances and exits of the assembly station. As the number of entrances is variable it is presented in detail and it is possible to search through the list and/or change the number.

In an assembly station, several objects can be merged into one. First, one object is received through the transport entrance (entrance 1) on which objects received through the assembly entrances (entrance number > 1) are assembled.



First the user must define a **list of acceptable object types**. These are objects which are received through the transport entrance. For each of these so-called basic object types a total assembly time can be defined. This will take place after assembling of all objects is finished. Furthermore an input must be made as to whether the assembly is manual or not. This can be defined for the assembling of the single objects as well as for the total assembly time. The parameters for the workers are described in the section of work area.

For the input of the object type, wildcards (*) are valid. They have a different meaning.

- When a wildcard is used in the distribution of work time in the field *Object type*, each work on an object not explicitly mentioned is processed by the values of this entry.
- When entering a wildcard in the field *New type*, 2 alternatives have to be considered. When a wildcard is used in the work time, the base object keeps his type after the assembly process. When the base object is referenced explicit in the work time, the object receives the type of the assembled object. If several objects have been assembled, the object receives the type of the first assembled object.
- When entering a wildcard in the assembly list, an object with the same type as the base object has to be assembled.

If the assembled objects are necessary for the determination of the throughput-time of the objects, this can be activated by the switch **Transfer throughput time**. Normally only the basic object would determine the throughput time. If the switch is activated, the earliest object of all objects (basic object and assembly object(s)), which occurred first in the system, determines the throughput time up to this station.

In principle the number of objects which can be assembled should be positive. If the switch **Allow zero** is set, the zero values are also valid. This is helpful with the use of attributes in combination with decision tables.

If there is no list defined for an object entering the assembly station via the transport entrance, it will be forwarded at the exit according to the transport time.

4.11.1 Waiting for Assembly

For each base object it has to be determined, whether it has to wait for assembly or not.

If *Waiting for assembly* is set to *Yes*, waiting occurs independently of the assembly condition and the situation at the entrances until the assembly of an object can be accomplished. The waiting procedure is broken off only if the most high priority and still grantable assembly list is processed or a deadlock has occurred (see below).

If *Waiting for assembly* is set to *No*, two cases are to be considered. If an assembly had not started yet and, at the assembly entrances, neither the suitable nor object types are waiting, then the object will leave the module unassembled. If however an object has been installed via the object entrances, then waiting occurs until an assembly list is fulfilled. If several assembly lists are available, it is attempted to fulfill that with the highest priority. The assembly procedure is terminated, however, despite everything if any assembly list is fulfilled and no further or the wrong objects are waiting at the entrances. While assembling multiple objects through one entrance, it is to be noted that the objects which can be assembled need a certain time, until these are moved up to the entrances.



4.11.2 Alternative Assembly Lists

Next, for every acceptable basic object an assembly list must be defined. After entering the basic object type, several variants of the assembly can be fixed depending on the object types waiting at the assembly entrance. Each of these variants is given a priority identification number to ensure an unambiguous decision in favor of one variant at any time. Also, in this area, the object type resulting from the fusion of several objects to a new object must be determined. For each assembly alternative, it must be specified how many objects of a type from which entrance are to be installed, so that the selection of this variant can be made. The number of the assembly entrance (each entrance is only to be used once), the object type waiting at this entrance and the number of objects needed for the assembly are to be entered.

If only one assembly list is defined to the basic object, the assembly is broken off, if a wrong object is waiting at an entrance where not all the indicated objects were assembled. This case is logged as deadlock in the error file.

Normally, the information of the objects that have been assembled, are no longer available after the assembly process. When the switch **destroy objects** is set to no, this information will remain. The objects can then be brought back into the system in a disassembly station.

If several assembly lists are defined, the goal is to fulfill the highly prioritized assembly list for as long as possible. The objects are assembled in arbitrary order, as soon as these wait at the assembly entrance. If at an entrance, which is not yet completely worked on and where wrong object waits, this list is marked as not grantable and the next assembly list is analyzed. Therefore, if the correct object waits at an entrance, it always waits, although a list with lower priority is possibly already fulfilled. On the other hand objects are not installed, as long as these are not used in the most highly prioritized assembly list and that list is still grantable. Only if *Waiting for assembly* is set to *No*, the work is broken off after a completed assembly, as soon as no further object is waiting at the assembly entrances after the assembling time of the last object. If after the waiting procedures finally no more assembly list can be fulfilled, this is logged as deadlock in the error file. The basic object leaves then the component possibly in a partly assembled condition.

The assembly time is determined by one of the known distribution functions. The entries change depending on the choice of distribution function. This assembly time only applies to one object waiting at the respective entrance and not to the total number of waiting objects.

4.11.3 Kind of Assembly

By the kind of assembly the sequence of assembling objects will be defined. The meanings are:

- **facultative :** The objects are assembled as soon as they wait at the entrance.
- **fixed sequence:** The objects are assembled in the sequence as defined in the assembly list. First the objects of the first entry are assembled, then those of the second one and so on. When using *fixed sequence*, no alternative assembly lists can be used.
- **forced parallel:** The assembly starts, when objects are waiting at all entrances. If there is a different number of objects according to the different entrances that should be assembled during the progress of work, only those entrances are considered, where objects have to be assembled. When using *forced parallel*, no alternative assembly lists can be used.



In all the cases, assembly of an object starts only if the previous process is completed.

4.11.4 Functionality

The steps taken during the assembly can be summarized thus:

An object enters the module via the transport entrance and covers the corresponding conveying distance. The transport entrance is then released. If an assembly is required, then the objects to be assembled are taken one after the other from the assembly entrances whereby the corresponding entrance node is released immediately. If no assembly was required, or the assembly has been completed, the object is passed on to the successor module which is ready to receive it. The next object can then enter behind the object leaving. When the loading path is greater than zero, assembly cannot begin until the old object has left completely so that the exit link is again released. When the loading path is zero, this verification does not take place.



4.12 Disassembly Station

The complementary module to the assembly station is the **DISASSEMBLY STATION**. It serves to separate objects into parts, as for example what happens when an electro-suspension train (EST) is unloaded or when depalletizing takes place.

Analogue to the assembly station, the setting of parameters for a disassembly module with three exits is described. A significant difference to the assembly station is that the disassembly station has no disassembly alternatives thus making it unnecessary to enter a variant list for each object type.

The linking up of the connections of a disassembly station to the rest of the system is done as follows. The link node at the entrance is released when the object has completely reached the place of disassembly. After the loading time it is additionally checked whether the old object left the station completely. Only if both conditions apply can the handling start. This check does not take place, if a module length of 0 is defined. Here the disassembly takes place according to a disassembly list, where objects which each leave the module via one of the disassembly exits are created. Only one object can leave through one exit. The objects leave when disassembly is completed, i.e. after disassembly time.

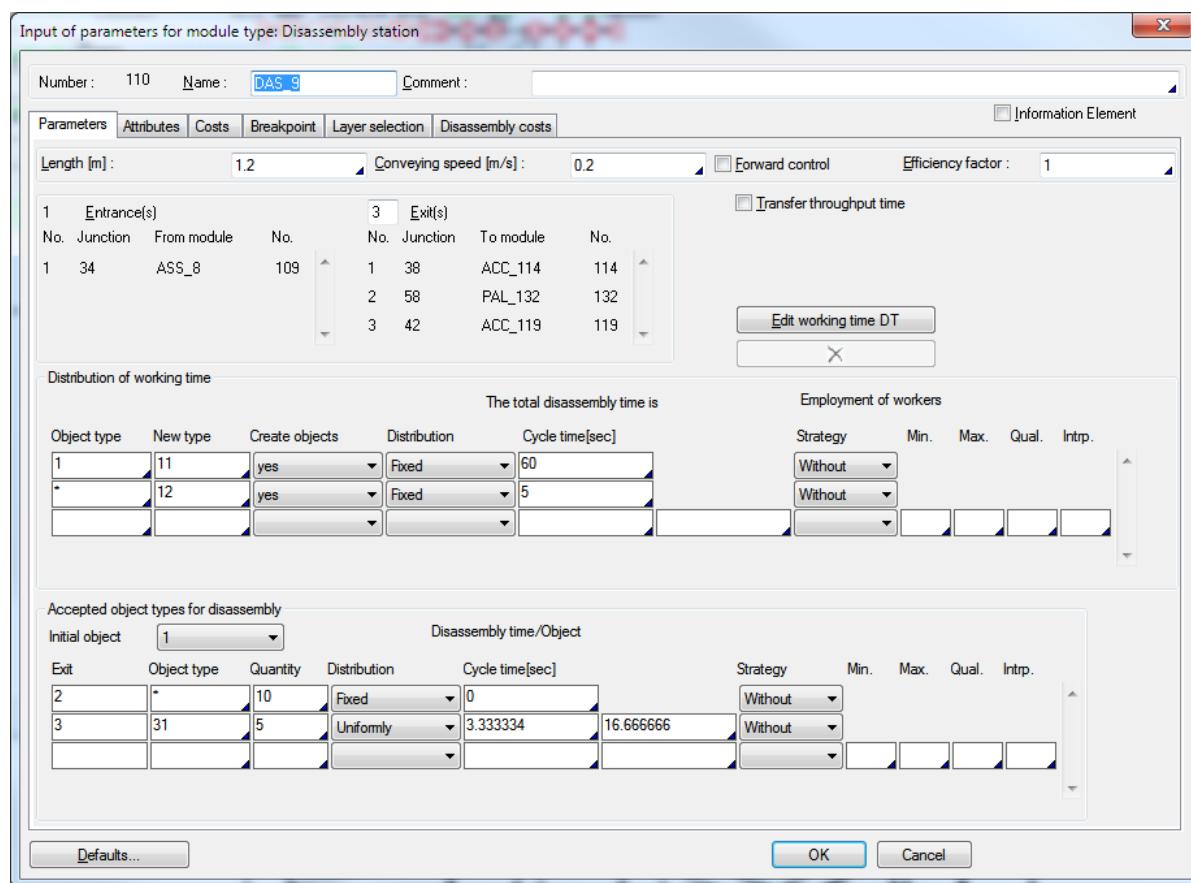


Figure 4.42: Parameters of a disassembly station

The entries to be made are identical to those of the assembly station. There is a list of the module entrances and exits, where, in this case, entrance 1 is identified with **Transport exit** and the other exits with **Disassembly exits**. The basic object type of the object to be



disassembled always leaves the module via exit 1, the transport exit; the rest of the objects resulting from the disassembly leave through the disassembly exits.

Objects which are not to be processed are passed on at the given speed. The list of objects to be disassembled is to enter. For objects of this type the object types resulting from the disassembly and the new object type of the basic object type have to be defined.

The latter are fixed in the list „distribution of working time“, in which the type of disassembly (manual or not) is defined.

If the throughput time for the basic object accrued till the current point of time is to be referred to in order to determine the throughput time of the disassembled objects by the system, this can be activated with the switch **Transfer throughput time**. In principle, the disassembly time for the disassembled objects is stored as the time of system entry. If, however, the switch is activated, the system entrance time of the basic object is transferred to the disassembled objects.

The disassembly list is completed with the new object types resulting from the disassembly. Their respective numbers and the number of disassembly exits through which they leave the module have to be determined.

The disassembly time is determined in the known way by selecting one of the distribution functions. This period applies for the total duration of the disassembly and therefore does not depend on the type of objects to be disassembled.

Additionally, work time, can be specified to each entry in the disassembly list, which results then per object. If one of these times possesses a value not equal to zero, each object is transferred individually after the indicated time to the successor. If the object has left to the successor, the next object is brought in accordance with disassembly list into the system.

If all individual times however are zero, then all objects are created and sent in parallel by the different outputs into the system. However for each output, a waiting time occurs, until the object is brought in completely into the successor, before the next object can leave the disassembly by this output.

Normally the disassembled objects are created new every time. When the switch **Create objects** is set to no, the process is trying to bring back those objects, which have been assembled previously in an assembly station, back into the system. When the base object is not carrying such objects, the objects will still create new and there is an error message.

During the input of the object types wildcards (*) are permitted. These have different meanings.

- If a wildcard is used in the list of basic objects, these parameters take place for the processing of all objects, which were not explicitly inputted.
- If the *new type* is a wild card in the list of basic objects, the type of the object remains the same after the disassembly.
- If the *object type* is a wildcard in the disassembly list, an object is produced, which possesses the same type as the basic object.



Only object types defined in the above-described list are disassembled. For other object types the module acts as a buffer, i.e. they leave the module (if that is possible) immediately after entry.

Basic objects which are of a different type after the disassembly can only leave the disassembly station when the module is completely free of all the newly-created objects. While one is leaving, the next object can take its place in the module, but disassembly takes place only after the preceding object has left the module completely.

4.12.1 Functionality

The steps taken during the disassembly can be summarized thus:

An object enters the module via the transport entrance and covers the corresponding conveying distance. The transport entrance is then released. If a disassembly is required, first the total disassembly time will be processed. Then the objects are brought into the system through the disassembly exits. This is done simultaneously, when no process time is defined. If no disassembly was required, or if the disassembly has been completed, the object is passed on to the successor module which is ready to receive it. The next object can then enter behind the object leaving. When the loading path is larger than zero, disassembly cannot begin until the old object has left completely so that the exit link is again released. When the loading path is zero, this verification does not take place.

4.12.2 Disassembly Cost

In the disassembly station it can be defined, how the cost of the basic object is to be distributed over the disassembled objects. For this, a percentage portion is to be indicated for each exit, which contains the key for the distribution. If several objects are disassembled for one exit, the portion is accordingly reduced.

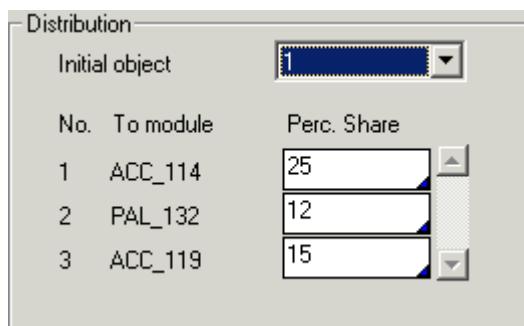


Figure 4.43: Disassembly costs

When no distribution of costs is defined, the basic object retains its value while exiting the station.



4.13 Multifunctional Workstation

In a material flow system, it sometimes happens that work, assembly and disassembly processes take place in the same station. For this the **Multifunctional workstation** offers the possibility of specifying different work processes just like in the work station. However, each work step is allotted a mode of processing (work, set-up, assembly or disassembly).

A multifunctional workstation possesses all characteristics of the workstation (refer appropriate chapter).

For the parameterization of a multifunctional workstation, a parameter mask appears which is shown here:

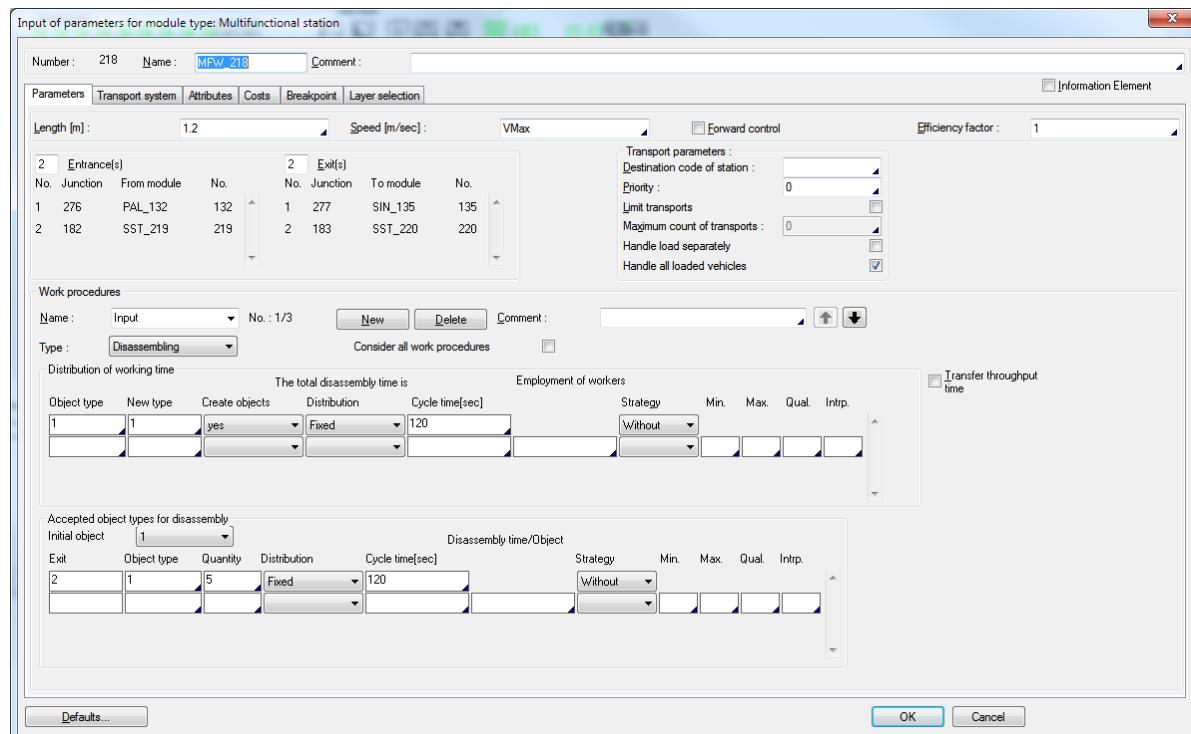


Figure 4.44: Parameters of a multifunctional workstation

A valid entry in the field **Destination code of station** is either a positive number or an empty field. Further parameters of the sub-dialog for the transport system will not be dealt with here in more detail. Further information can be found in the chapter [transport systems](#). **Work Procedures**

In a multi-functional workstation, several work procedures can take place successively. For this, individual parameter sets can be defined for each step. As a function of its type, the necessary data correspond to those of the workstation in the case of the type work or set-up or those of the assembly or disassembly. Beside these well-known operational the multi-functional work station sequence it possesses a further procedure: Cleaning.

When the object enters the workstation, it is checked whether a process has been parameterized for this object in the first work step. After the processing, the next step is changed to and the analysis starts again. However, the object type that has resulted from the processing in the previous step is analyzed here, in case it has changed because of the list of



the new types. If no working time is defined in a work procedure for the current object type, the processing is aborted and the object leaves the module.

This abort can be prevented by the option **Consider all work procedures**. If this switch is activated, all work steps are analyzed, even if no processing is intended for the first object in the current work procedure.

Object type	Distribution	Cycle time[sec]	Strategy	Min.	Max.	Qual.	Intrp.
1	Fixed	120	Without				

Accepted object types for disassembly	Initial object	Object type	Quantity	Distribution	Cycle time[sec]	Strategy	Min.	Max.	Qual.	Intrp.
2	1	5		Fixed	120	Without				

Figure 4.45: Definition of work procedures

4.13.2 Cleaning

In multifunctional work station also cleaning processes can be defined. For this the work procedure of the type cleaning is available. The parameterization takes place in same way as with procedures of type *Working*. This procedure is indicated separately in the statistics.

Object type	Distribution	Cycle time[sec]	Strategy	Employment of workers

Figure 4.46: Definition of Cleaning Process

With cleaning three cases are to be differentiated.

- If the type of the first work procedure is cleaning, this begins in the moment, if an object wants to enter the module. The object waits in front of the station, until the process time has passed. The duration is determined as a function of the type of this object.
- If cleaning takes place within a sequence of work procedures, then the process time passes exactly the same as when working.
- If the cleaning process is the last work procedure, then the object can leave the module before this work procedure begins. The module however cannot take up a new object during this work procedure. This happens only then if the cleaning procedure is final.



The process duration is determined as a function of the type of the object, which left the module.

If workers are needed for cleaning then in the corresponding work area activities of the type *cleaning* have to be defined.

4.14 Distributor

In a material flow system there are often transport alternatives for one object, i.e. the object is distributed to certain positions in the system. The most common module with which the illustration of such distribution positions is possible in the system is called the **DISTRIBUTOR**.

The distribution module has one entrance and a variable number of exits, but at least two. The capacity of a distribution module is one object so that at any one time only one object can be transported simultaneously from the entrance to one of the exits.

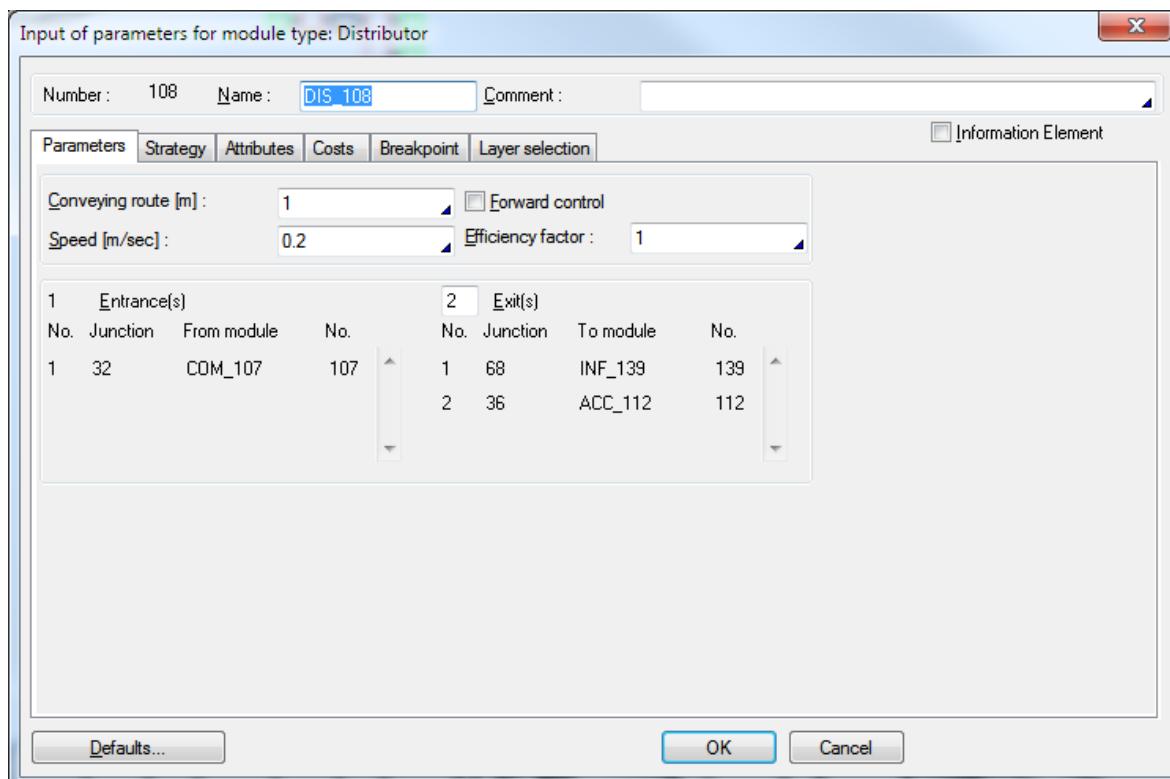


Figure 4.47: Parameters of a distributor

Setting the parameters for the distribution module is limited to the input of the **length** and the **conveying speed**. In this regard, length is understood to mean the conveying distance between the entrance and one of the exits whereby this distance is the same length for each exit. Forward control can be set as required. A distribution strategy can be defined, and the parameter dialog contains as usual a list of the module connections.



4.15 Discharger

A special form of distribution module frequently used in continuous conveying systems is the **DISCHARGER**. There are many technical devices for a discharger module, for example pushers, slewing arms and tilting pans are used to divert an object from the main conveying direction.

A discharger has one entrance and at least two exits. The link to the first exit is called the *main conveying direction* and generally means an object is transported in a straight line. The other links are called *discharging direction* and are always a diversion from the main conveying direction. The discharger has a capacity of one object.

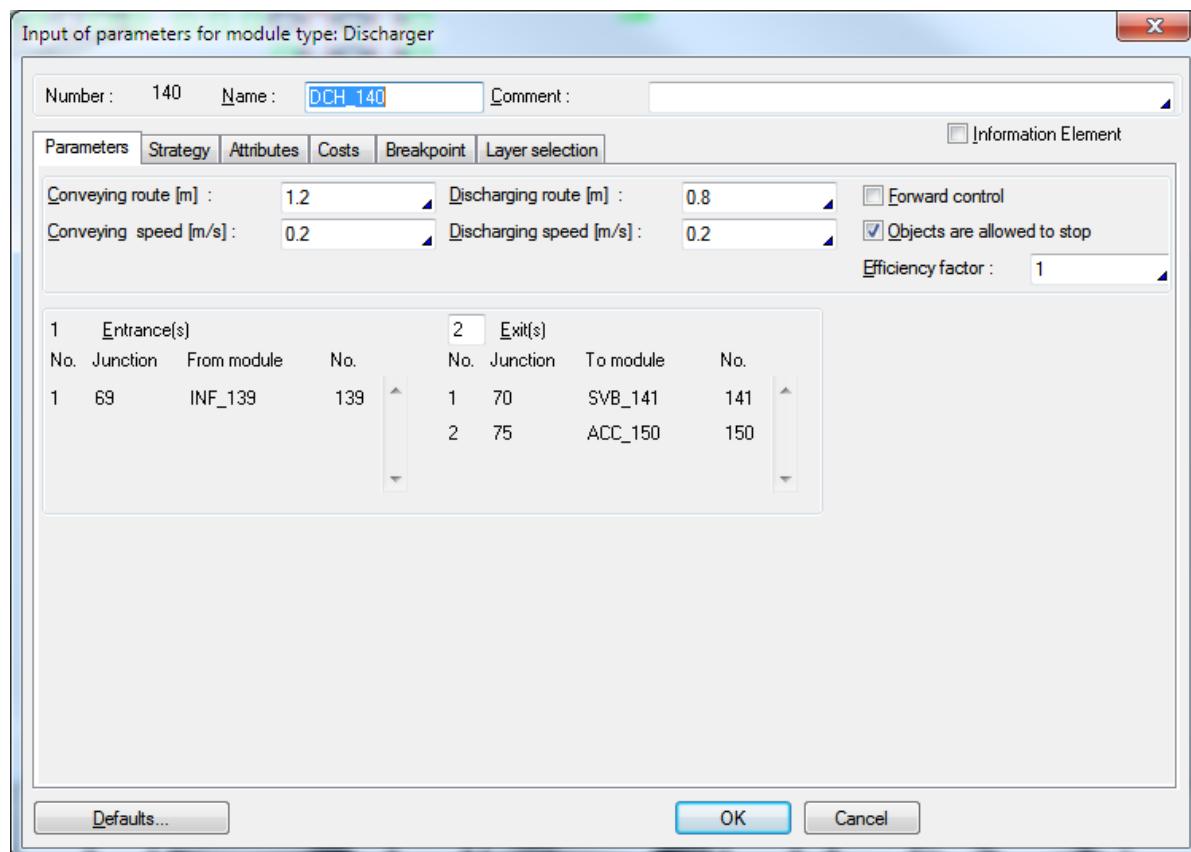


Figure 4.48: Parameters of a discharger

The parameters for a discharger are similar to that of an infeed element. The two exits of the module are called **main conveying exit** and **discharging exit**. In addition to setting the free-space position control for the main conveying route, the length of the discharging route, the length of the main conveying route and the conveying speed for each route can be determined.

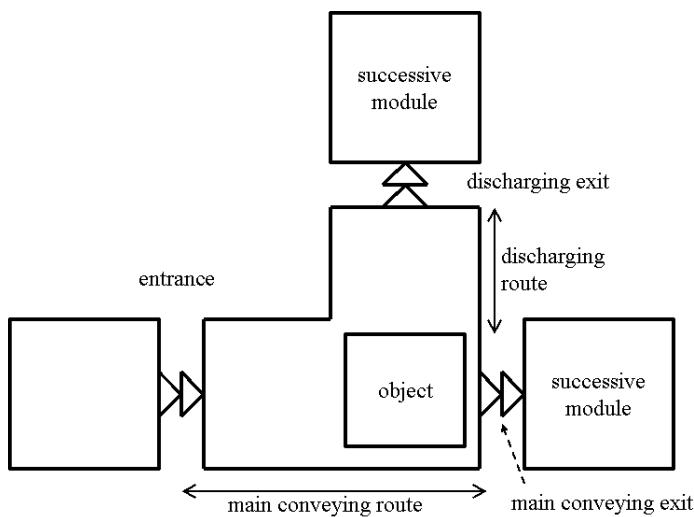


Figure 4.49: Parts of a discharger

An object entering the discharger module releases the link at the entrance when it has passed the whole main conveying route. Then a decision is made according to one of the distribution strategies, to be defined in the mask, as to whether the object is to be discharged.

If this is the case, then the object travels the discharging route before being able to enter the relevant succeeding module. If the main conveying direction has been selected, the object can enter the other succeeding module immediately. Prerequisite for a continued journey into one of the succeeding modules is, of course, that this is vacant.

Furthermore, it is asked whether objects to be discharged are permitted to be stopped at a blocked discharging exit. If the succeeding module is not ready to accept the object there follows either further transport through the main conveying exit or the object remains stationary until the successor is ready to accept it.



4.16 Swiveling Belt

A further special distribution module for continuous conveying systems is provided by a **SWIVELING BELT**. This is the technical realization of a movable conveyor belt suspended at the entrance which can be slewed horizontally or vertically to one of two possible exits.

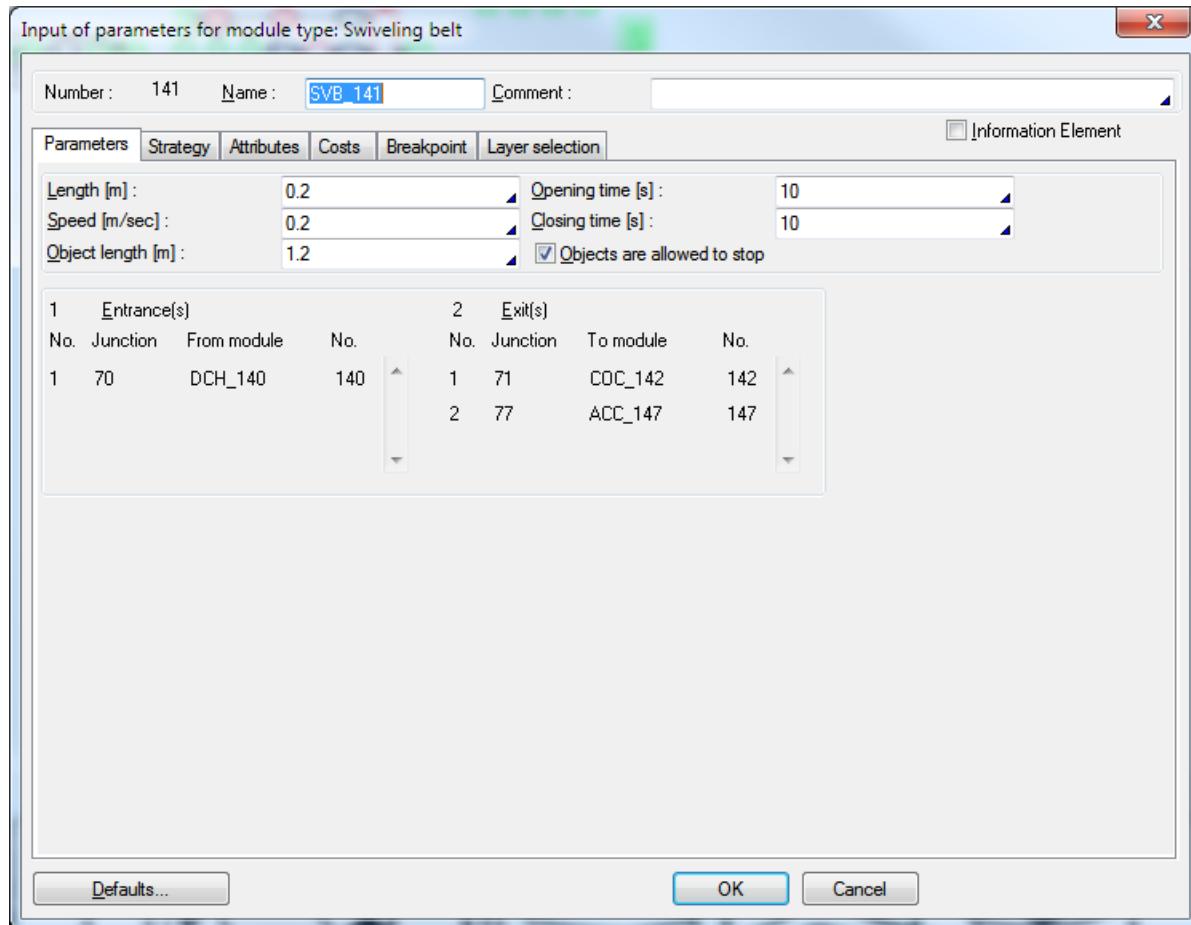


Figure 4.50: Parameters of a swiveling belt

In principle, the swiveling belt is a conveying section which can head for one of two exits. Three of the parameters to be entered correspond to those of the conveying section. These are the length of the section, the object length, which is the same for all types, and the conveying speed; plus the opening and closing times. The times characterize the duration of the swiveling procedure and are to be given in seconds. Swiveling from exit 1 to exit 2 is called opening, vice-versa closing.

By means of the one of the distribution strategies to be defined, the exit for the front-most object on the swiveling belt is selected. As the belt is continuously moving, it could occur that the transport time to the end of the belt is shorter than the required swiveling time. The user is able to influence the behavior in such a special case by selecting a special strategy. The parameter dialog asks whether the belt is to be stopped. If this is the case, and swiveling cannot be carried out before the exit is reached, the belt is stopped as soon as the relevant object reaches the end of the belt and then swiveled to the other exit. If the belt cannot be stopped in the above-described special case, swiveling is dispensed with and the object conveyed further through the exit set.



4.17 Combining Station

The analogue to the distributor is the **COMBINING STATION**. It serves to illustrate the contrary case, namely that of linking object flows. The number of entrances in a combining station is variable, however there are at least two and there is just one exit. Technically it is like the distributor, i.e. it can accept one object and the conveying routes between all entrances and the exit are the same.

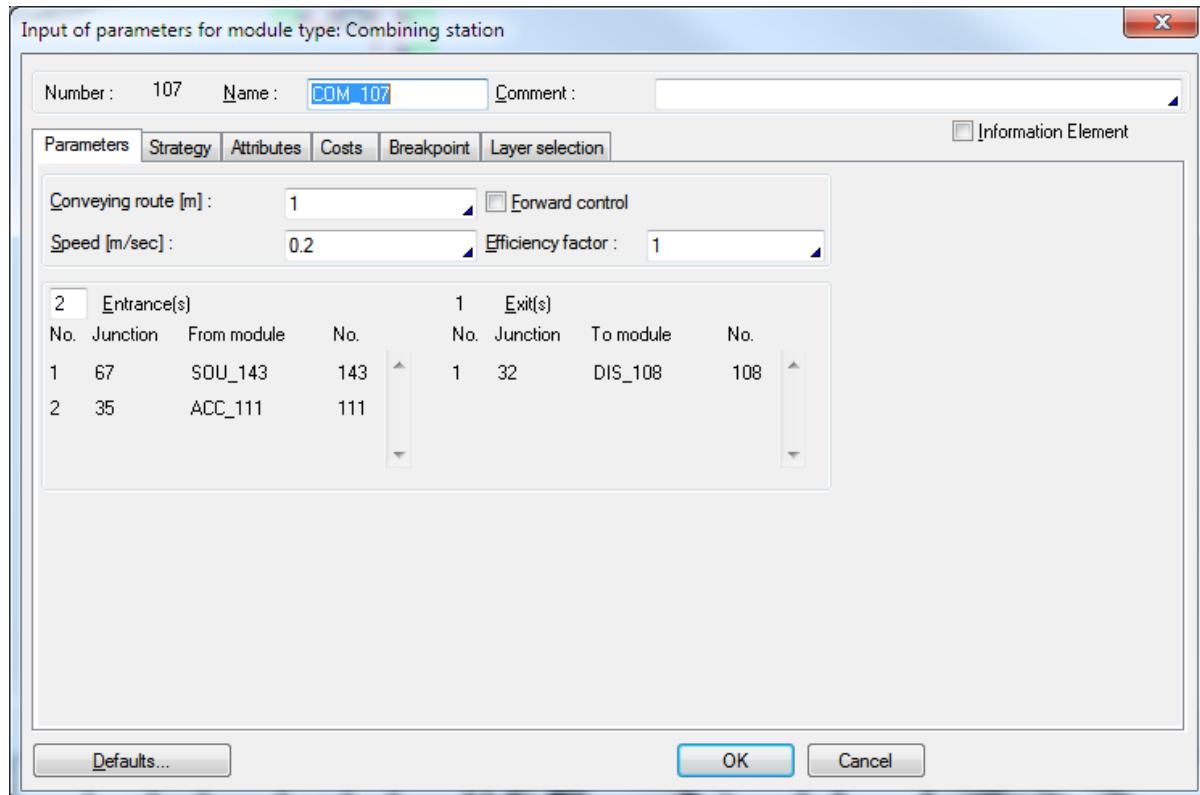


Figure 4.51: Parameters of a combining station

Setting the parameters of a combining station is analogue to that of the distributor. A strategy, here a right-of-way strategy of course, is to be selected; a **length** and the **conveying speed** of the module are to be entered, whereby length means the same as in the distributor, i.e. the conveying route between one of the entrances and the exit.



4.18 Infeed Element

Frequently in continuous conveying systems, a special merging module is used called an **INFEED ELEMENT**. Technically it resembles the discharger where pushers or slewing arms are used for the slip-in procedure.

The infeed element has at least two entrances and one exit where the connection between the first entrance and the exit is the main conveying direction; the other connection is the slip-in direction. One object can be accepted.

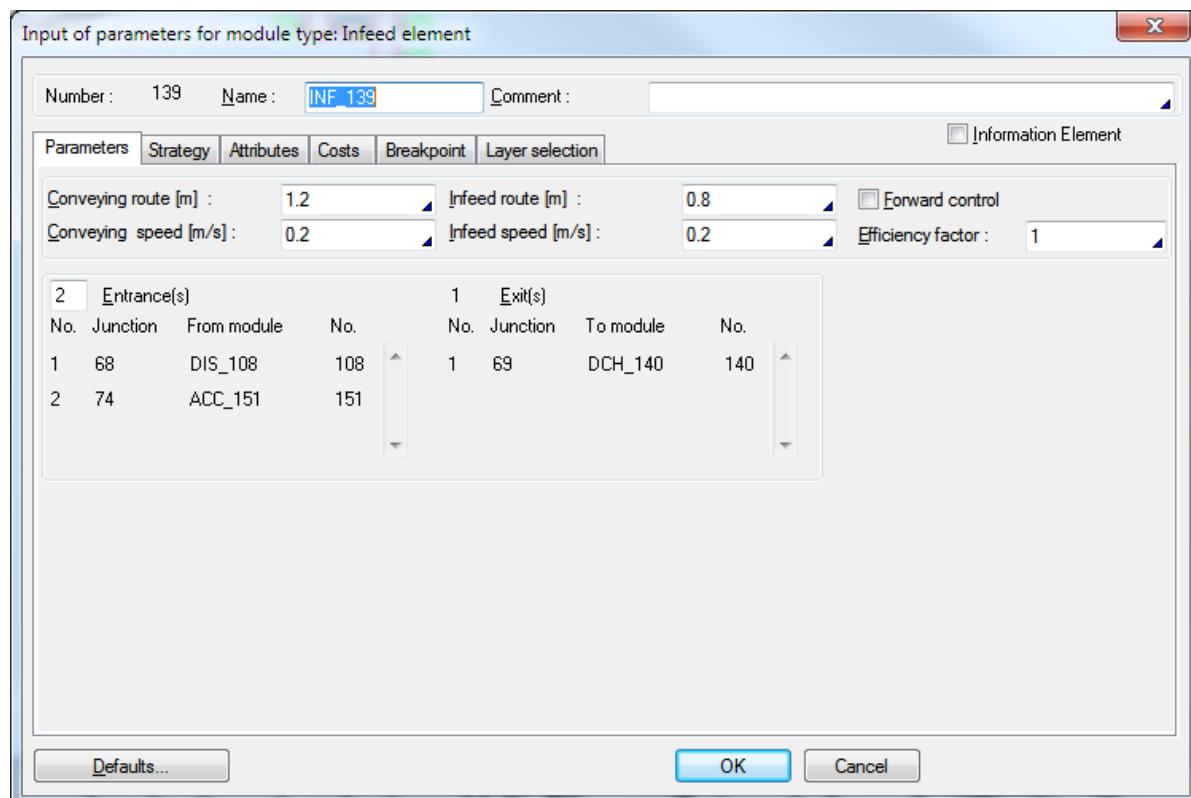


Figure 4.52: Parameters of an infeed element

Although the infeed element is really a merging module with two entrances, the two modules differ in their parameters because a difference is made between conveying entrance and infeed entrance at the two entrances and certain parameters have to be set separately for each entrance. These are **length of conveying route**, **length of infeed route**, (both values given in meters), **conveying speed** and **infeed speed** (in meters per second). This is illustrated in Figure 4.53.

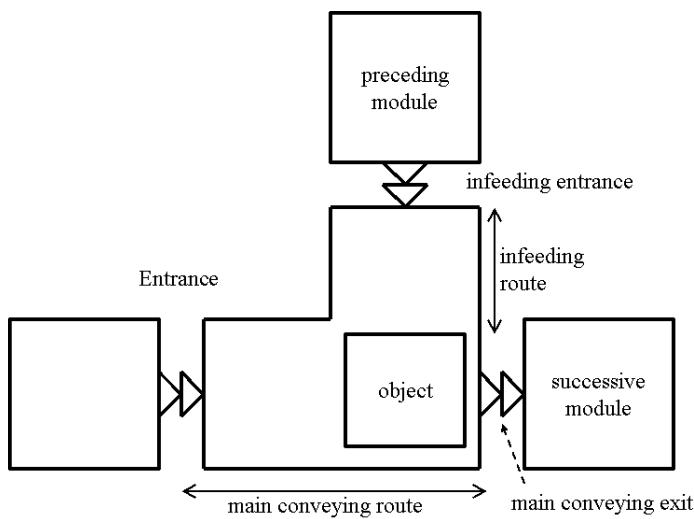


Figure 4.53: Parts of an infeeding element

Additionally, the forward control can be selected for the main conveying route of an infeed element. The infeed route is principally forward controlled.

The entrance out of which the next object can enter the infeed element is determined by a certain right-of-way strategy determined by the user. Then the object is either transported via the conveying route or via the slip-in route to the exit. On reaching the exit, the link to the relevant entrance is released. A new infeed procedure cannot take place after the object has been passed on to the succeeding module until the object has completely entered the succeeding module and the link at the exit has been released.

As described in previous sections, there is a special module-specific right-of-way strategy for the infeed element which is called **priority of the conveying route**: an object is only accepted through the slip-in entrance, when, during the infeed procedure, there is no other object in the preceding module of the conveying entrance (see right-of-way strategy) otherwise the slip-in procedure is postponed.



4.19 Crossing

The module type **CROSSING** is used for modeling of intersections and complex distribution and linking modules. Examples of conveying modules which can be illustrated with crossings are junctions in transport system plants, buffer sections which can be crossed in both directions, distribution and/or linking modules which have varying conveying routes between entrance and exit.

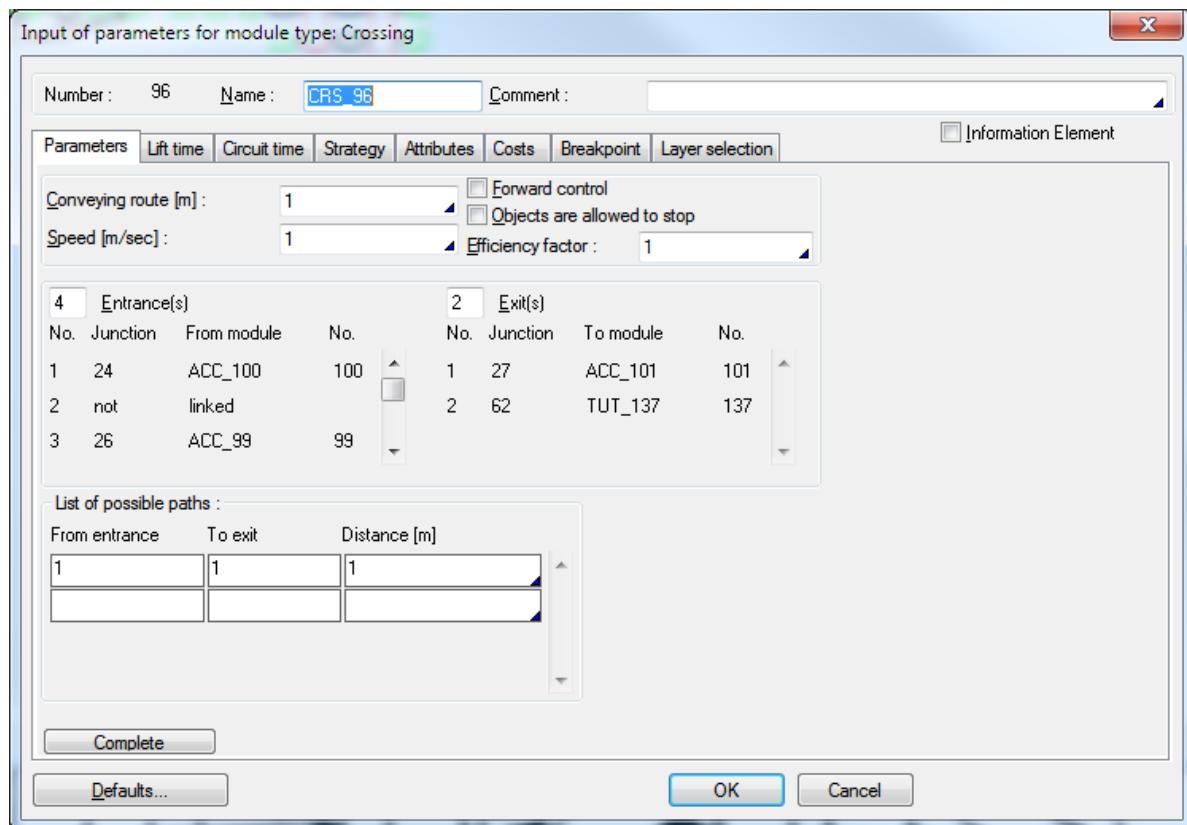


Figure 4.54: Parameters of a crossing

The crossing is a module with a variable number of entrances and exits. The capacity of a crossing is one object so that it can be seen as a simple buffer section with one entrance and one exit. At least in theory, each entrance can be linked to each exit but in practice it is often sensible to limit the number of these theoretically possible routes, where, however, there must be at least one route for each entrance and exit. An important item of crossing parameters is therefore the definition of a linking route between entrances and exits.

All possible routes between entrances and exits have a common conveying route whose length must be defined under **Conveying route [m]**.

In case of more than one entrance to the crossing a right-of-way strategy is to be defined.

In case of more than one exits of the crossing a distribution strategy is to be defined.



4.19.1 Possible Paths

In the **List of possible paths** it can be specified, which entrances and exits are supposed to be connected. Relations between an entrance and an exit, which are not contained in this list, cannot be carried out.

In the parameter dialog, the entrances and exits to be linked together can be determined in the corresponding list.

List of possible paths:		
From entrance	To exit	Distance [m]
1	1	1

Complete

Figure 4.55: List of possible paths

The lengths to be entered at this point are the path-dependent conveying routes which are additional to the mutual route of all paths.

Each linking route between an entrance and an exit therefore is composed of two parts, on the one hand the route length, which all routes have in common, and, on the other hand the route dependent lengths defined from the difference between the total length of the respective linking route and the common conveying route. With the button **Complete** all start/destination-relations will be generated. The distance of these will be set to 0 [m].

The conveying speed is the same for all routes in an element of the type crossing.

4.19.2 Functionality

Basically, the transport of an object through a crossing is as follows:

At first, the entrance from which the next object enters the crossing is determined by a right-of-way strategy. Then the object enters the module, upon which the conveying time that is valid for all routes firstly elapses. After this time, the entrance node is released, i.e. the object is considered to have completely entered. Finally, the exit, through which the object will leave the module, is determined using one of the distribution strategies. The object is then transported to the exit via the additional route-dependent conveying route.

Additionally, the user can determine whether the object can be stopped or not. If the successor for the exit, which has been determined by the distribution strategy of the node, is not ready to accept the object, it waits in the module until the modeler permits further transport. If the halting of an object has not been permitted, it is checked before entrance into the node whether the relevant successor is ready for acceptance so that the module can definitely be passed without hindrance. The latter only occurs when the crossing has at least two entrances and two exits. If there is only one entrance and one exit the situation is the same as the one described for the shuttle so that acceptance takes place as an avoidable blocking of the node is impossible.



4.19.3 Lift Time

Lift times can be assigned to the crossing. These correspond, for example, to an elevating platform when removing a roller conveyor. As many as desired lift positions can be assigned to the module. For this the value is to be entered in the field **Lift positions**. Subsequently, one of these positions must be assigned to each entrance and exit. Additionally, the duration of the lift time between the positions is to be specified. If up and down rates are different, this can be activated via the appropriate switch.

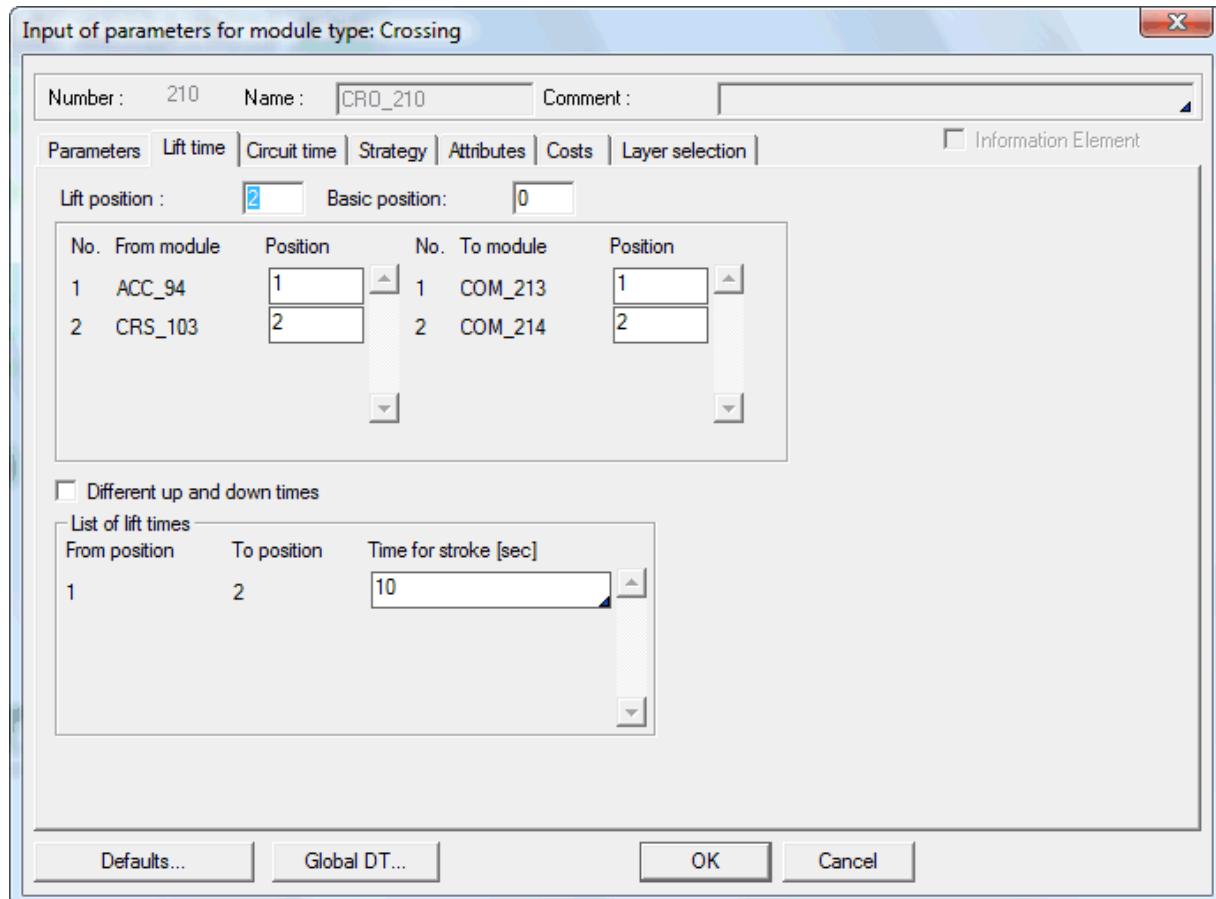


Figure 4.56: Parameters of lift time

4.19.4 Functionality of Lift Time

Before an entry, it is examined whether the elevating platform has the correct position. Otherwise, the object waits in front of the crossing, until the elevating platform is in the correct position. After the entry of the object the stroke takes place into the correct position to the exit, through which the object is to leave the module.

If a basic position is defined, putting the elevating platform back takes place automatically into this after the delivery, if no further object is to be transported.



4.19.5 Circuit Time

Circuit times can be assigned to the crossing. These correspond, for example, to the switch of an electrical overhead conveyor (EHC). First the number of possible positions must be entered in the field **Cycle position**. Then a cycle position can be assigned to each path, which was defined in the parameters. Different relations can possess the same position. Additionally, the cycle time is to be defined, which results during the transition from a position to the other one.

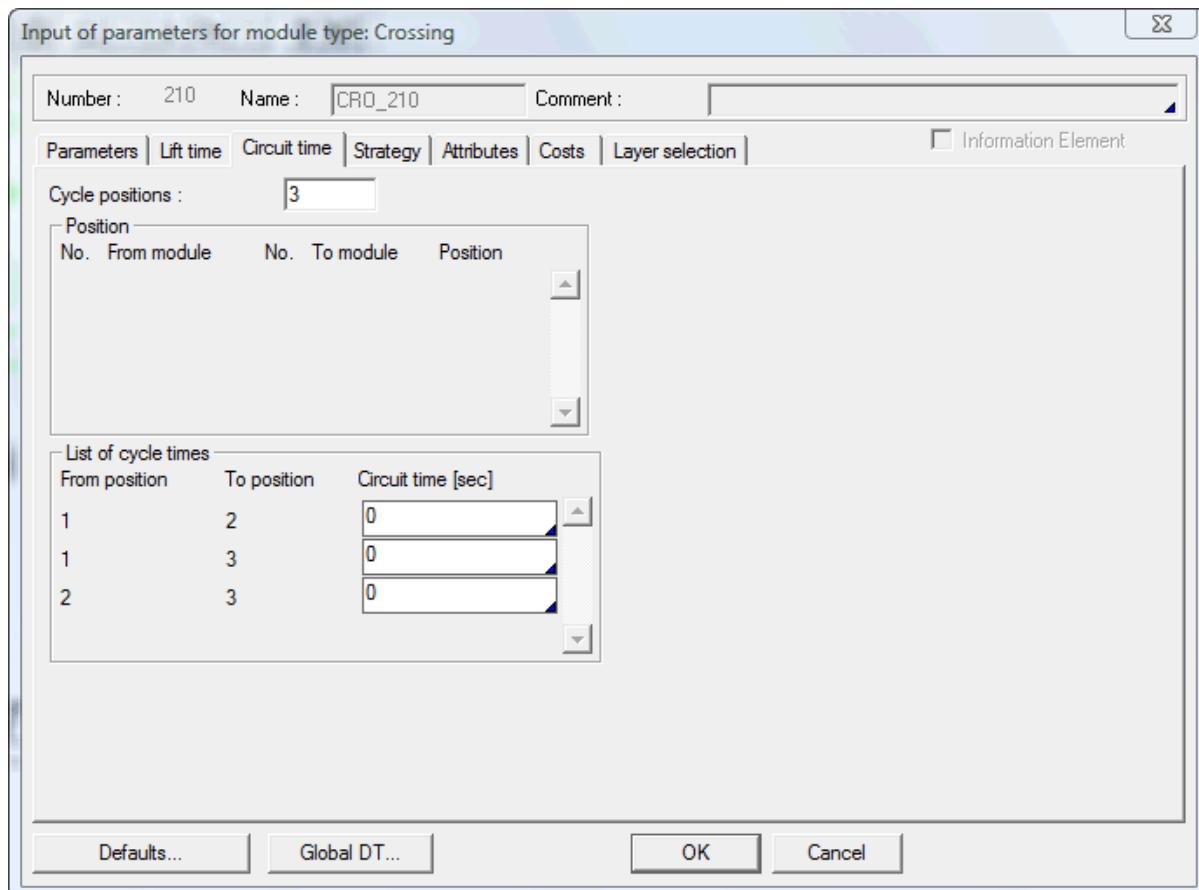


Figure 4.57: Parameters of circuit time

4.19.6 Functionality of Circuit time

Before the entry, it is examined whether the switch has the correct position. For this, the destination of the object must be entered, since the position corresponds to the switch of a starting/destination relationship. Otherwise, the object waits in front of the crossing, until the switch is in the correct switching position. After driving through the crossing it can be switched again only if the object left the crossing completely.

Note: Circuit time and cycle times are mutually exclusive.



4.20 Turning Table

The **TURNING TABLE** is a module type which is to be found in continuous conveying systems, in electro-suspension rail systems and in transport systems. Its implementation can be varied, for example, there are turning tables with one or two conveying sections on the revolving table; conveying sections which convey in one or both directions and turning tables which can be turned in one or both directions. What these implementations all have in common is that basically only the revolving time is influenced.

A turning table can have any number of entrances and exits, even the combination *entrance 1 - exit 1* is possible in theory. The list in the parameter dialog shows which of these possibilities has been selected.

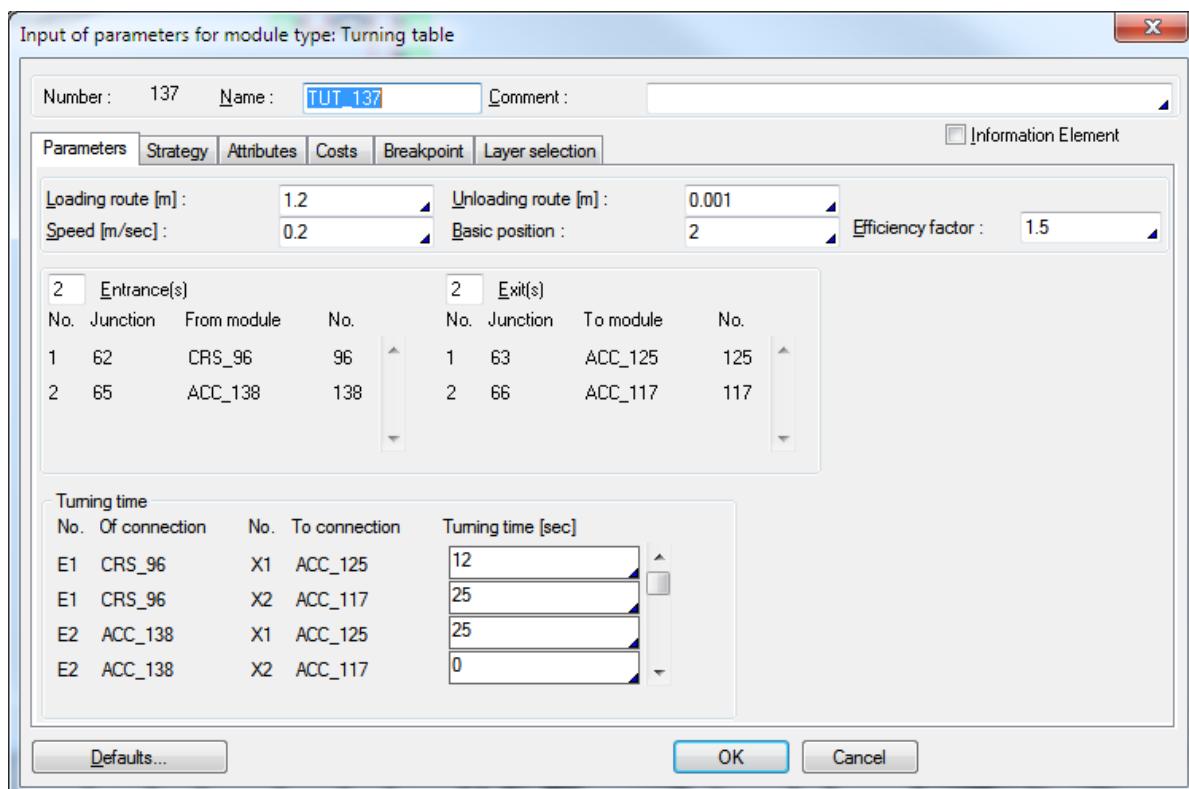


Figure 4.58: Parameters of a turning table

The parameters of a turning table are completed by entering the **loading path**, the **unloading path** an object must travel in order to report to the successor module and the **speed** with which these two ways can be traveled. The following sketch shows the importance of these ways.

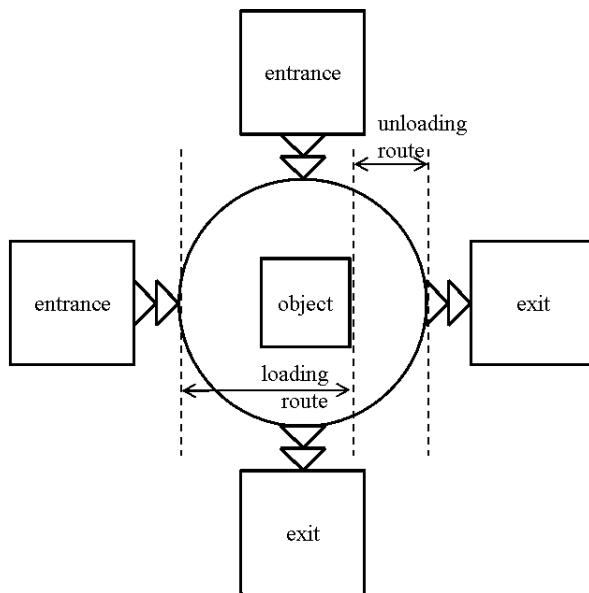


Figure 4.59: Parts of a turning table

The turning table also allows an additional right-of-way strategy to be selected which prefers the entrance, to which, at that moment, the shortest revolving time is needed. Accordingly with more than 2 exits, one determines by the distribution strategy, through which exit the object leaves the module.

4.20.1 Turning Time

The time taken between the connections during turning is an important parameter. These times can be seen in the area **Turning time** in a list containing all the possible connection combinations. Two time values must be entered for each connection combination - one for the revolution from connection 1 to connection 2 and a further one for the opposite direction.

Turning time				
No.	Of connection	No.	To connection	Turning time [sec]
E1	JUN_96	X1	ACC_125	12
E1	JUN_96	X2	ACC_7	25
E2	ACC_138	X1	ACC_125	25
E2	ACC_138	X2	ACC_7	0

Figure 4.60: Turning times of a turning table

A **basic position** can be defined for the turning table as it is done similarly for the shuttle by specially marking a relevant position. The turning table then turns back to this entrance when it is not occupied. A turning table with only one entrance has a basic position of 1. The standard set „0“ means that the turning table remains in the current position if there is no object waiting at another entrance. That is the reason why in the list of turning times, combinations from entrance to entrance are taken into account when defining the basic position, whereas revolutions between two exits can in no case occur.



4.20.2 Functionality

The turning table transports an object as follows:

When an object approaches the turning table and is permitted to enter it in accordance with the right-of-way strategy, the turning table is turned to the required position. After this revolving period has ended, the object travels on to the turning table via the loading path and then releases the entrance node. Then a turn is made to the exit chosen in accordance with the distribution strategy. When the exit is reached, the unloading route has to be covered before the object reports to the exit to proceed to the successor module. A new object can only enter the module when the old object has released the exit. This means that there can only be one object on the turning table at one time.

If the turning table has two entrances and two exits as in the previous case an object will not be accepted until it is certain that the successor module is ready to do so (cf. shuttle and module bunch).



4.21 Pallet Changer

The module type **PALLET CHANGER** is used for modeling a special palletizing process which is really better described by the term **RE-PALLETIZER** for it is the task of a pallet changer to place objects from one pallet to another. There are therefore two entrances and two exits, one for the pallets (pallet entrance and/or pallet exit) and one for the objects (object entrance/object exit). These four connections define two routes through the module, the pallet route and the object route. The transport mode of a palletizer is therefore described by four parameters, by the lengths of the object route and the pallet route and a transport speed for each way. This information is to be entered and it is also possible to set a forward control.

The duration of the palletizing process can be defined by means of one of the available distribution functions.

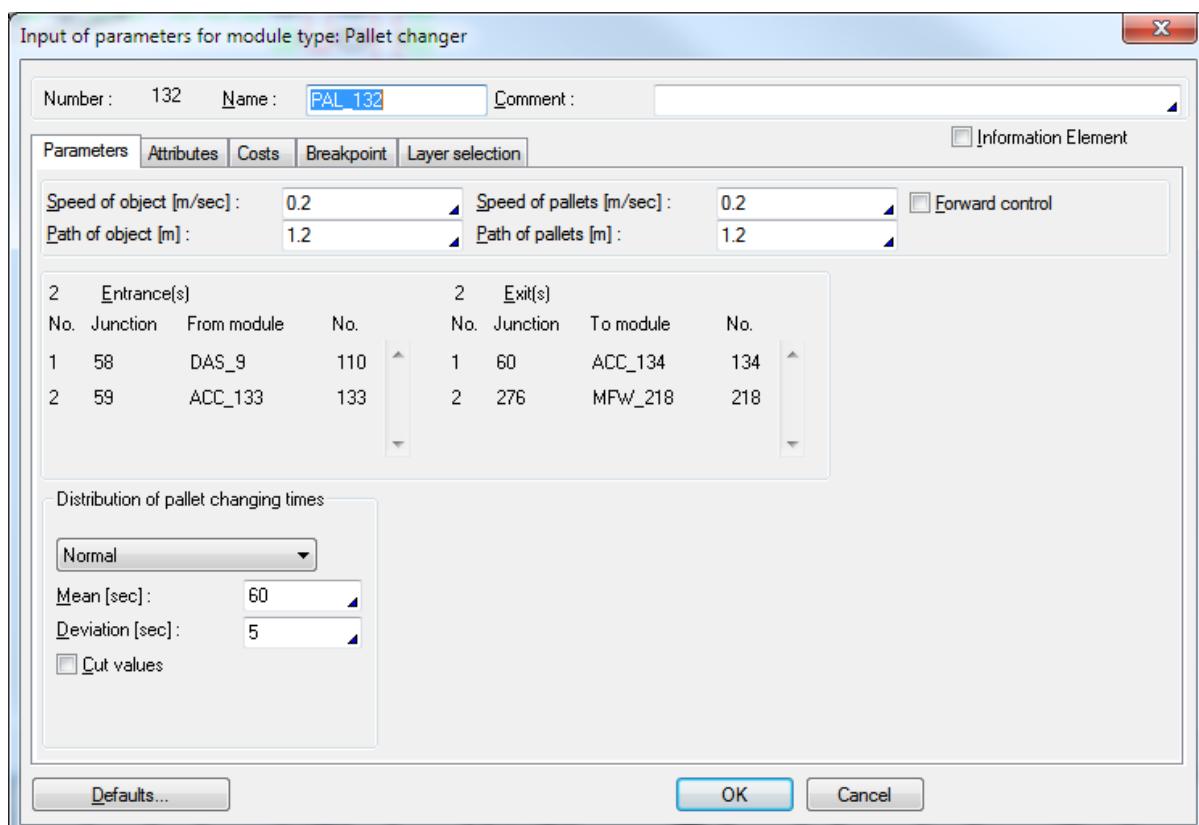


Figure 4.61: Parameters of a pallet changer



The following sketch shows the exact significance of the parameters.

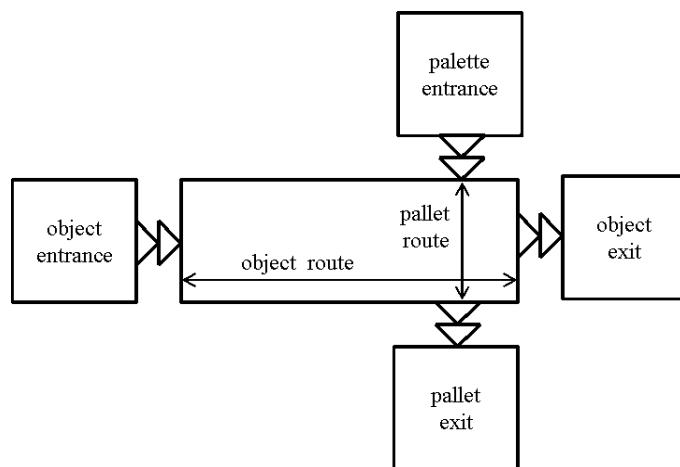


Figure 4.62: Parts of a pallet changer

4.21.1 Functionality

The module behavior which is important for the modeler can be described as follows:

An object enters the module covering the respective transport route at object speed. The link at the object input is then released. Then the new pallet which has covered the transport route at pallet speed enters. The link node at the pallet input is released so that the palletizing/ pallet change can begin.

After palletizing has ended, the old pallet leaves the module. It must leave the module completely before the object which is now on a new pallet can leave the module.

By means of model constructions, other palletizing processes can be displayed. For example, placing on to a new pallet was modeled with a sink for the old pallet leaving. Palletizing several objects on one pallet can be reproduced with an up-stream assembly unit.



4.22 Conveying Circuit

A standard structure in a continuous conveying system is the **CONVEYING CIRCUIT**. It serves to represent sorting circuits, re-positionable tables etc. which transport objects to exactly defined positions on a constant route to one another. Objects in a conveying circuit are always moved in such a way that two objects do not change their route to one another.

The conveying circuit is divided into segments. A conveying circuit is characterized by entering the number of segments, the length of a segment in meters and the transport speed.

In order to define the positions of the connections of a conveying circuit unambiguously, a zero position is determined, from which the position of a connection is given in meters in the transport direction. The parameter dialog shows the relevant list of connections. The values of the positions must be less than the total length of the circuit, which is calculated by multiplying the length of one segment with the capacity.

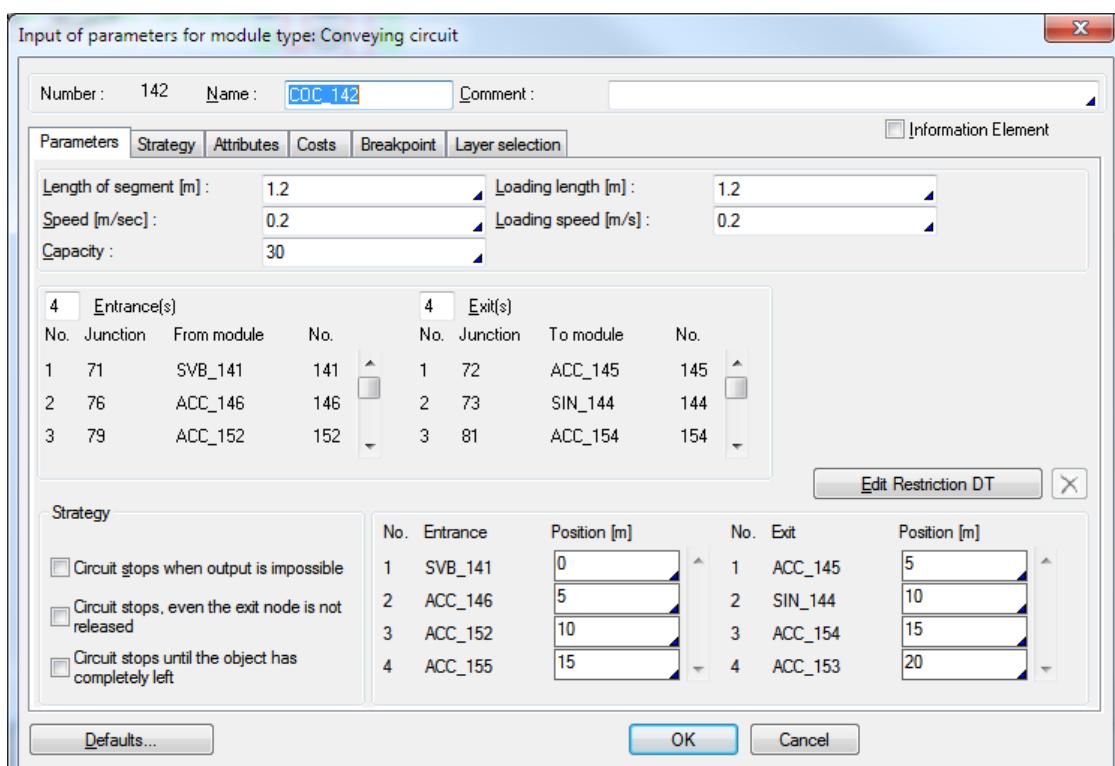


Figure 4.63: Parameters of a conveying circuit

As one of the most frequent applications of a conveying circuit consists of selecting objects, a list of permissible object types must be defined for each exit. This is the only distribution strategy supported by the conveying circuit.

The conveying circuit does not stop during the slip-in process. When a loading length of 0 is set, the entrance junctions are released immediately. In the other case this is done after the time for entering the module. If the circuit has stopped (by strategy or in case of a disturbance), the entering process is correspondingly completed.

The alternative **Circuit stops when output is impossible** refers only to the case that an object can be delivered, but delivery is not possible because the successor module is not ready to



accept it for some reason. Then the circuit stops if the user so wishes by selecting the relevant checkbox. In this case, it is neither possible to feed in an object which is waiting at an entry into a vacant position. . On the other side the conveying circle can be steered in such a way that the **Circuit stops, even the exit node is not released**, i.e. that the previous object did not completely drive out yet. In addition, it can be determined, whether the **Circuit stops until the object has completely left**. Then the next cycle takes place, when the exit junction is not occupied, that means the object has entered the succeeding module completely.

The conveying circuit has a restriction decision table. This is called before any delivery of an object. This can be used to determine whether the object is now to be actually delivered. This corresponds to a second strategy. If no decision table is defined the delivery is carried out strictly in accordance with the distribution strategy of the support group.

4.22.1 Functionality

In the following drawing, care must be taken that when determining the connection positions, measurements are taken in the middle of the conveying circuit. The zero position is, for example, in the middle of segment 1, thus measurements are made along the dashed line for the positioning of connection 2. This means that all connections, whose position is less than the length of a segment, are assigned to the first segment. Similarly, the assignment of the further segments takes place. The determination of the process time takes place without consideration of the positions. The process time between two segments results from the number of cycles that are needed in order to transport an object from the start segment to the target segment.

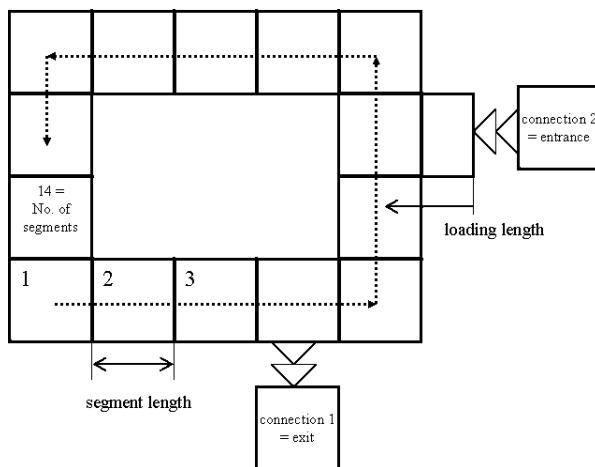


Figure 4.64: Parts of a conveying circuit

The characteristic for the behavior of junctions of a conveying circuit is the fact that the delivery or overtaking of objects is possible only between two cycles, where, with **cycle** in this case, the progress of one segment is meant. First it is checked whether objects wait at an input that has just passed a free segment. These objects are transferred. Subsequently each object, which is at an output, which is admissible for this type of object, is transferred to the successor module, if that is ready for input. The input junctions are released immediately or after the duration of the entering time. The release of the output junctions only have influence on the flows of the conveying circuit, if the switch *Circuit stops until the object has completely left* is activated.



When modeling a system, care must be taken to see that only those objects are slipped-in into the conveying circuit for which there is actually an exit; because contrary to a shuttle, for example, there is no enquiry to check the delivery of the object before it is accepted and no strategy which automatically slips-out the object after a certain time. That means an object without an defined exit can no longer leave the conveying circuit, which in unfavorable circumstances is blocked with these object types causing a deadlock.

4.23 Shuttle

A **SHUTTLE** module displays construction modules which use a transport vehicle firmly linked to a construction module for transporting objects; e.g. lifting devices, (elevators) or shunting instruments.

The number of entrances and exits of a shuttle is variable; it has capacity for one object. For a vehicle with two entrances and two exits, a parameter window appears as shown in Figure 4.65:

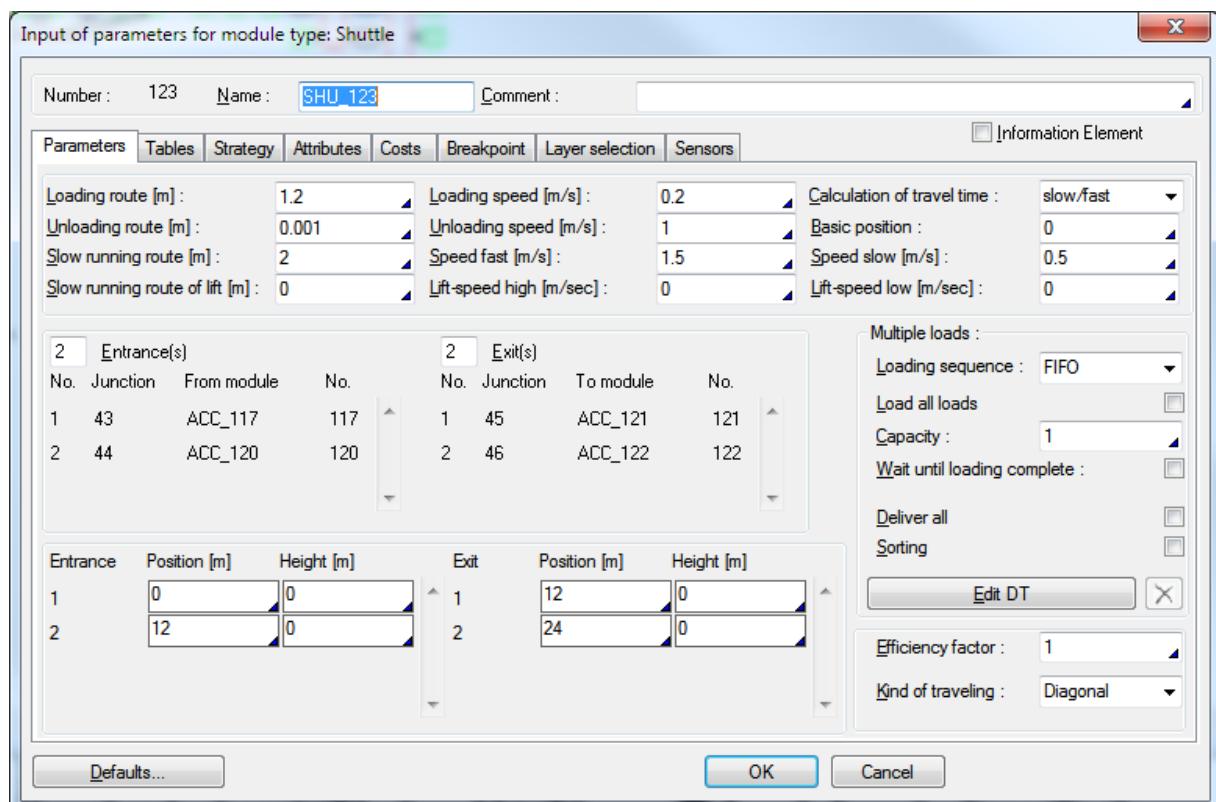


Figure 4.65: Parameters of a shuttle

The parameters in the upper area describe the travel behavior of the shuttle. Each trip can be divided into four phases. First the vehicle is accelerated to its maximum speed (acceleration phase). In the second phase, the vehicle travels for as long as possible at maximum speed and then it is slowed down in the delay phase to its positioning speed before in the positioning phase being linked to the desired connection in the positioning phase. In the model, however, positioning, acceleration and delay phases are only described by the required running route and an average speed. These results in the travel behavior of a shuttle being described by three parameters: **Slow speed** (average speed from the three above-mentioned phases), **Speed fast** (max. speed of the vehicle) and **Slow running route** (running route necessary for phases 1, 3 and 4). These values are to be entered for the vehicle in the x and y-direction. Here, the total



travel time is the larger of the two travel times. This means that the shuttle drives diagonally from the starting position to the destination.

The following figure shows these four phases:

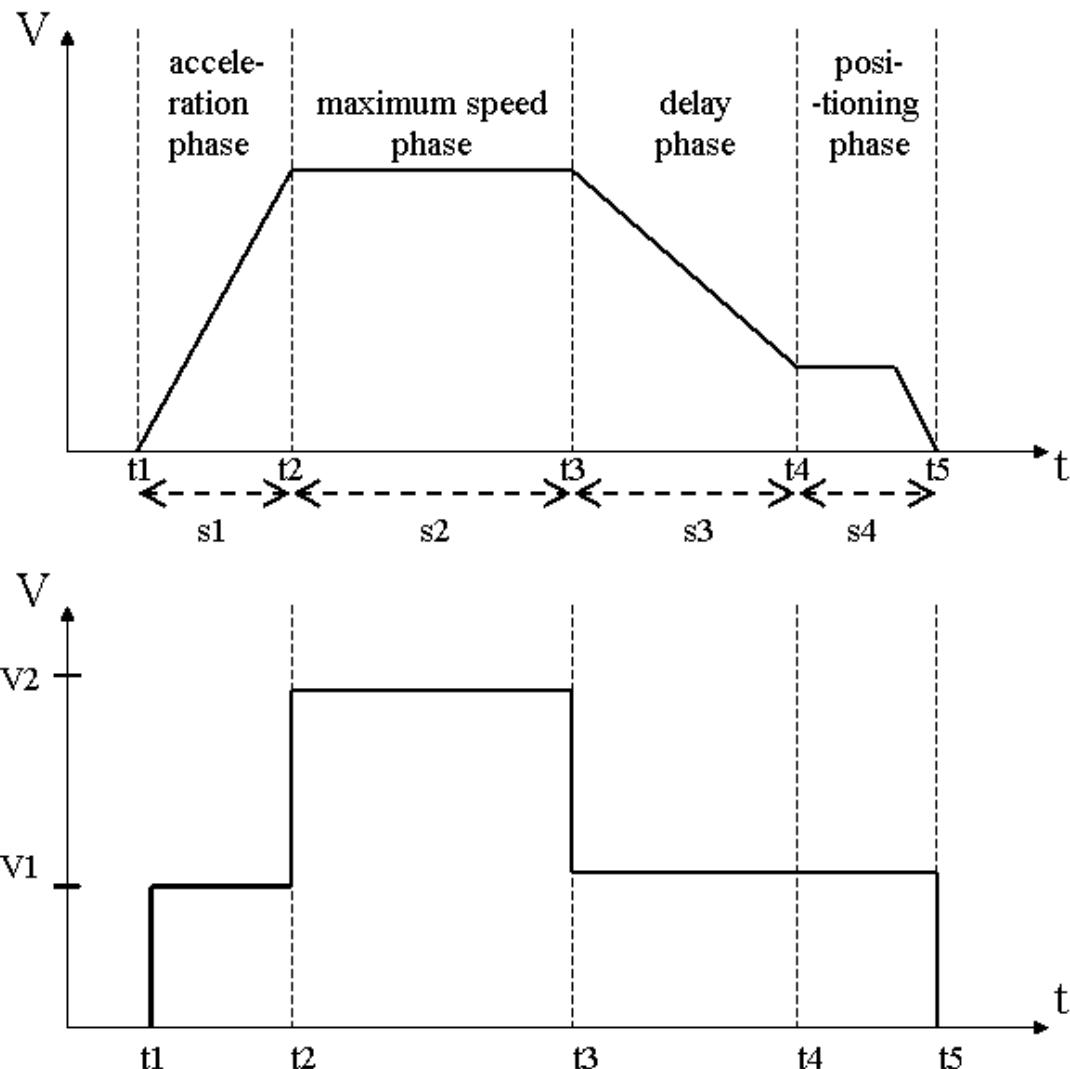


Figure 4.66: Distribution of velocity

Finally, the modeler can define a **basic position** for the shuttle by entering the number of an entrance. Consequently, if no object is standing by any other entrance, the vehicle can return to this basic position (at this entrance). *Zero* as an entrance means that none of the entrances is marked in any particular way and therefore the vehicle remains stationary in its current position, as long as no object is standing by to be picked up. A shuttle with only one entrance has a basic position of 1.

One area usually contains a list with the entrances and exits of the component. With the help of the **efficiency factor**, all processes can be accelerated or retarded. Here values from 0.01 to 10.00 are permissible.

With the help of the **kind of traveling**, it can be specified, how the travel time between 2 positions is calculated. The *diagonal travel* is the longest of the two travel times



(horizontal/vertical). The *rectangle travel* is calculated by the sum of horizontal travel and vertical travel. With the third alternative (Y0-travel) the table drives only vertically into the 0-Position, proceeds then horizontally on the x-position of the destination and afterwards vertically up to the delivery position. The kinds of procedures are independent of whether the table is loaded or not.

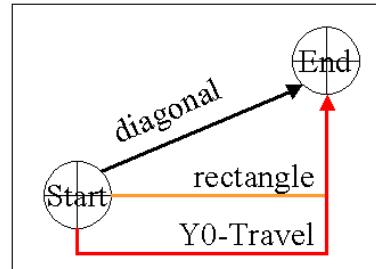


Figure 4.67: Alternative kinds of traveling of the shuttle

One area contains, as usual, a list of the entrances and exits of the module. Also the user can select a right-of-way strategy here with the addition of a **shortest running route** strategy. This strategy means that if there are objects standing by, the vehicle moves to the entrance from which it must travel the shortest running route from its present position after delivering the object.

In another part of the parameter dialog where the distribution strategy is defined, it is similar. Here **shortest running route** means the exit the closest to running route from the present position is targeted. A second strategy is also possible.



4.23.1 Positions

Furthermore, each connection (entrance and/or exit) is allotted a position by determining an imaginary zero point on the distribution vehicle, from which every other connection in direction x and in direction y has a positive distance (e.g. the bottom left corner of the vehicle) as it is not possible to enter negative values. The relevant list with connections and their positions are also in the parameter dialog. In the system, they serve to determine the running route between two connections, which is important for the special strategies described. **Position [m]** refers to the running route of the connection from the nil point in direction x and **Height [m]** refers to the running route in the y direction. Typical entries essential for the shuttle are also made. Loading and unloading are characterized by the four parameters length of the loading route, length of the unloading route, loading speed and unloading speed, as shown in Figure 4.68.

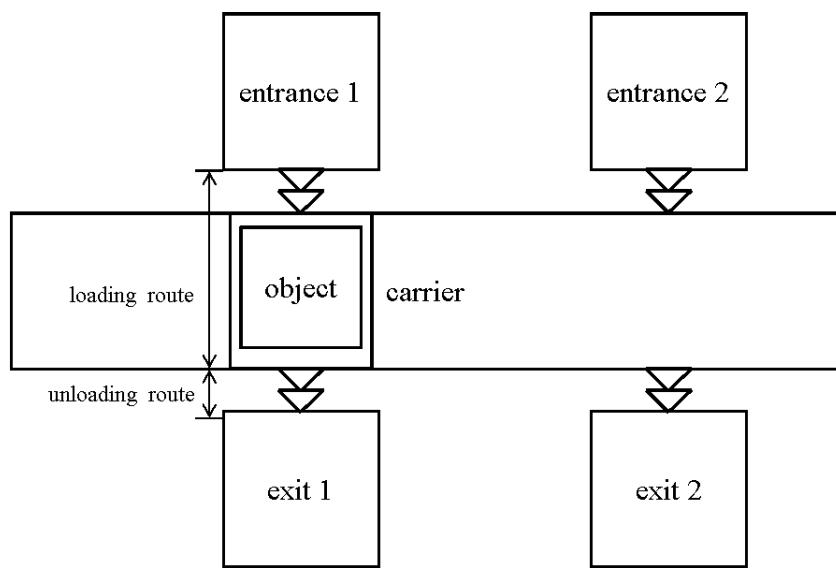


Figure 4.68: Functional principle of a shuttle

The figure also explains special features of the transport strategy of the vehicle. As previously explained, the shuttle checks that a successor module is ready to accept a waiting object if the number of entrances and exits is greater than one. This is to avoid unnecessary delays. If, for example, an object is to be conveyed from entrance 1 to exit 1, a check is first made to see if exit 1 is ready for acceptance. When the object would be accepted in any case and exit 1 is disturbed, the shuttle would be blocked for the duration of the failure. By enquiring beforehand, there is always the option of transporting the object from entrance 2 to exit 2, although exit 1 is disturbed.

If there is only one entrance or one exit, an object will be accepted in any case even when one or several exits are disturbed as delays cannot be avoided as in the above-described case. If the shuttle has only one entrance and one exit is disturbed, the object is accepted because transport is not possible from any other entrance and subsequently the object must wait, whether it is in front of or in the vehicle. If there is only one exit and this is disturbed, the object is nevertheless accepted because all objects regardless from which entrance they come are not transported further until the exit has been released.

4.23.2 Functionality

The transport mode of a shuttle can be summarized as follows:



First, the entrance, from where the next object is to be fetched, is determined using a right-of-way strategy. The transport vehicle travels to this entrance. The object travels the loading route and is then completely on the transport vehicle so that the link at the entrance can be released. The transport vehicle then brings the object to the exit which has been allocated by the distribution strategy. There, the object reports after the conveying time for unloading to the successor module. In the above-described way, a check is made before transport whether the successor module is ready for acceptance. That is why shuttles are considered a part of the module bunch.

The next transport cannot take place until the object has completely entered the successive module, which means the exit link has been released.

4.23.3 Multiple Load

The shuttle can possess a **capacity** larger than one. Then so many objects are loaded, as it is indicated in the capacity.

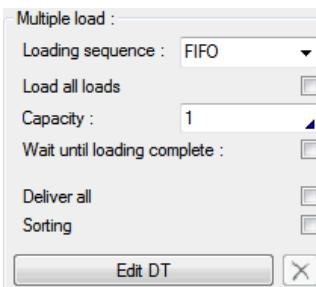


Figure 4.69: Parameter of Multiple Loads

If the switch **Wait until loading complete** is active, the shuttle waits, until it is completely loaded. Otherwise the load procedure is interrupted, if after loading of the current object no further object waits at the same entrance. Furthermore the load procedure is broken off, if the type of object changes or the entrance junction is blocked or the module at the entrance is disturbed.

The objects can be recorded in different ways. This will determine the order in which the tax burden and thus the trips take place. The selection is made in the field **Loading Sequence**.

If **Load all loads** is enabled, all waiting objects will be taken. This happens independent of a change of the object type.

At the output usually the front objects will be issued. When the switch **Deliver all** is active, all objects are given, regardless of whether the object is specified for the exit.

When the switch **sorting** is active, all objects are given, which are provided for this output. This is done regardless of the location on the shuttle (random access).

With a decision table it can be determined whether a partially discharged shuttle should overtake an object. If the decision table exists, it is called after each transfer or discharge. If the return value is > 0, the entrance is approached.

ATTENTION! The function of multiple loads is restricted with shuttles with several tables. Only *Capacity*, *Load all Loads* and *Wait until loading complete* is considered.



4.23.4 Table

A shuttle can be defined with several tables. These can proceed independently on one rail. For each table an individual **Basic position** can be defined. Furthermore the tables can be **passivated** individually. Then the shuttle behaves in such a way, as if this table does not exist.

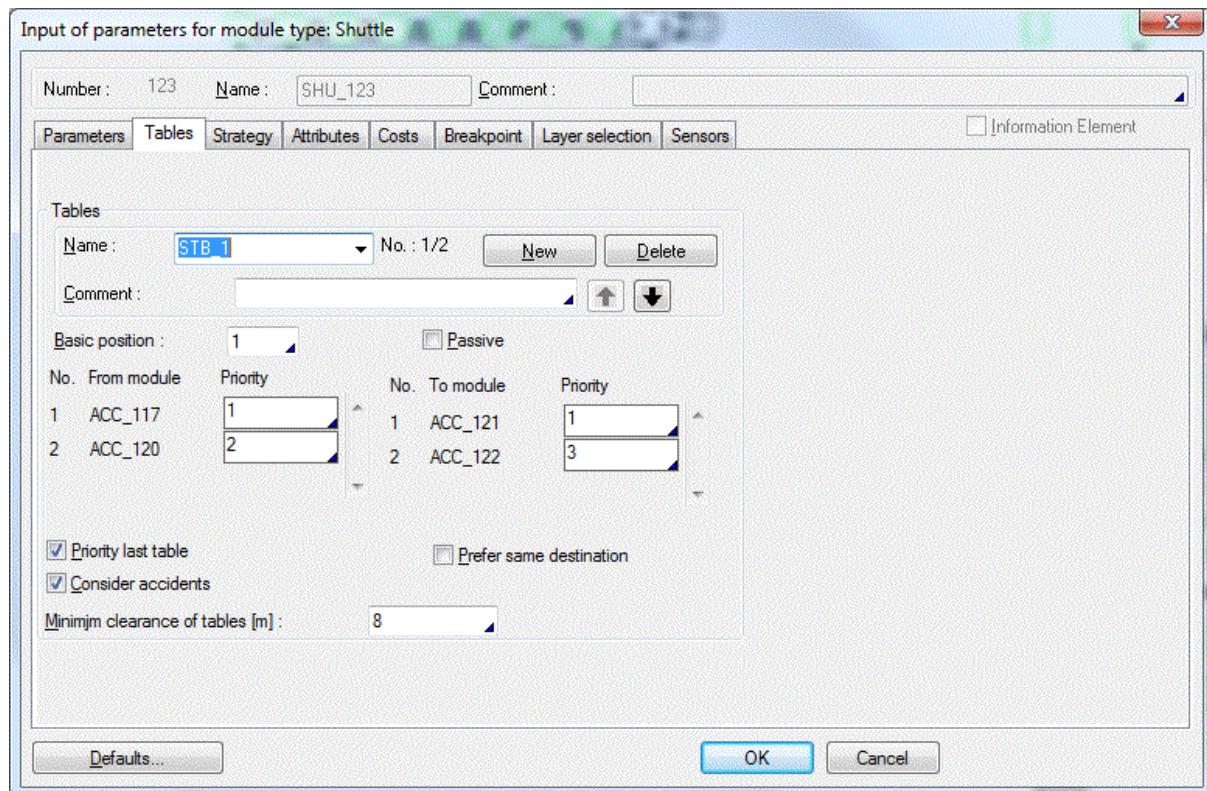


Figure 4.70: Parameters of the table

4.23.5 Functionality of the Table

For each object, which announces itself at the shuttle, a driving order is produced, which specifies, from which entrance to which exit this object will be transported. For the determination of the table, which is to realize the order, the priority list is consulted. The characteristic value of an order concerning a table is calculated by the sum of the assigned entrance/exit priorities. Afterwards the free table that possesses the smallest characteristic value is selected.

Thereby the priority 0 takes a special case. Entrances or exits, which possess the priority 0, will not be used by the table.

An order is executed only if it can be realized without any accident. For this, all entrances and exits of the orders of the different tables must be arranged in ascending order. For tables that have already taken an order, the comparison takes place in both cases with the position of the exit of the order. Tables, which do not have an order, are not regarded. If the value of **Minimum clearance of tables** is positive, the entrances and exits must lie apart at least by this value. This accident examination takes place only if the switch **Consider accidents** are activated.



The tables with the smaller numbers should therefore use the inputs and outputs to the minor positions (X-Coordinates). The height of the inputs and outputs will not be considered.

In each case, it is tried to execute an order with no accidents. If a possible accident is determined, the order is not executed. The order with the next lower priority is regarded. The other order is realized, if this is possible without an accident.

When the switch **Priority last table** is set, the currently release table will be favored if several tables are available for the next transport

When the switch **Prefer same destination** is set, the shuttle transports all objects to the same exit in the case that the shuttle carries multiple objects.

4.24 Paired Shuttle

A **PAIRED SHUTTLE** is in principle a shuttle which has two tables for improved performance. Objects can be loaded and unloaded simultaneously, thus avoiding empty runs. A paired shuttle can have any number of entrances and exits although maximum pick-up capacity is two objects. When the vehicle is actually loaded with two objects, there are three transport options, known as double cycles.

Double cycle type 1: One object is loaded and one object is unloaded;

Double cycle type 2: Two objects are loaded;

Double cycle type 3: Two objects are unloaded.

When setting parameters for a paired shuttle, a few parameters are to be entered for the slow running route, fast speed and slow speed, i.e. those values characteristic for the travel behavior of the vehicle and which are in accordance with those of a simple shuttle.

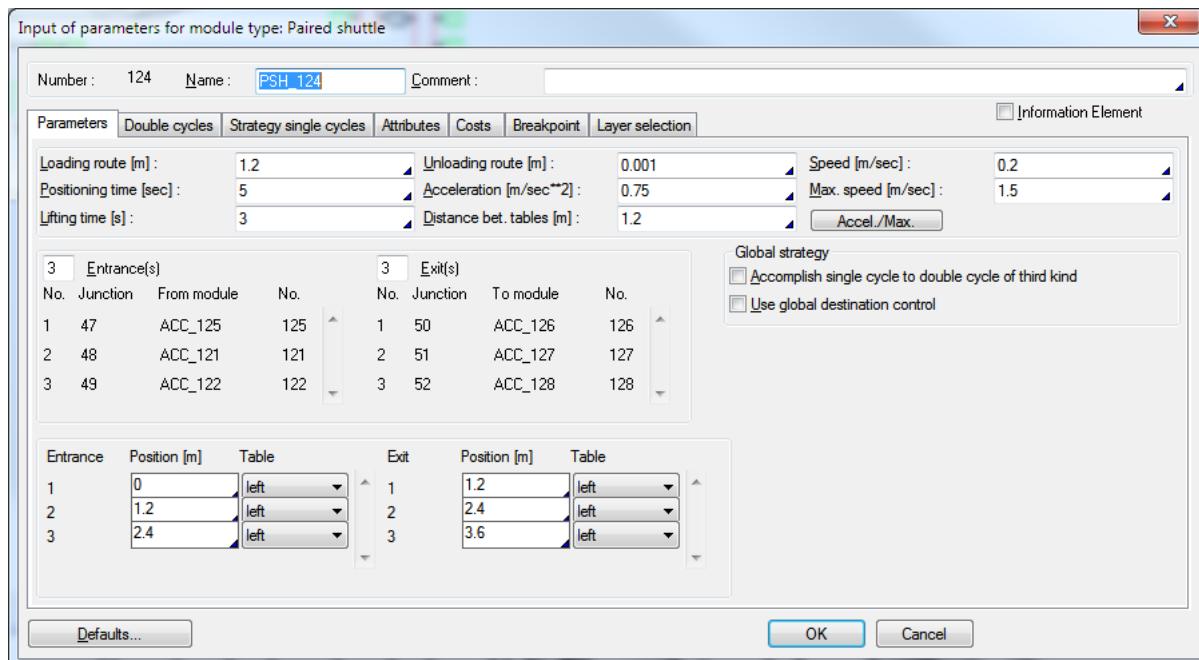


Figure 4.71: Parameters of a paired shuttle



Loading and unloading are more similar to the turning table, since, apart from the length of the loading and unloading path (see following diagram), only one transport speed is to be entered, unlike the shuttle that has a separate speed for each path. However, there is one more parameter for the lifting time which is necessary when coupling and de-coupling of vehicle tables before each loading and after each unloading, and the value for the distance between both vehicle tables.

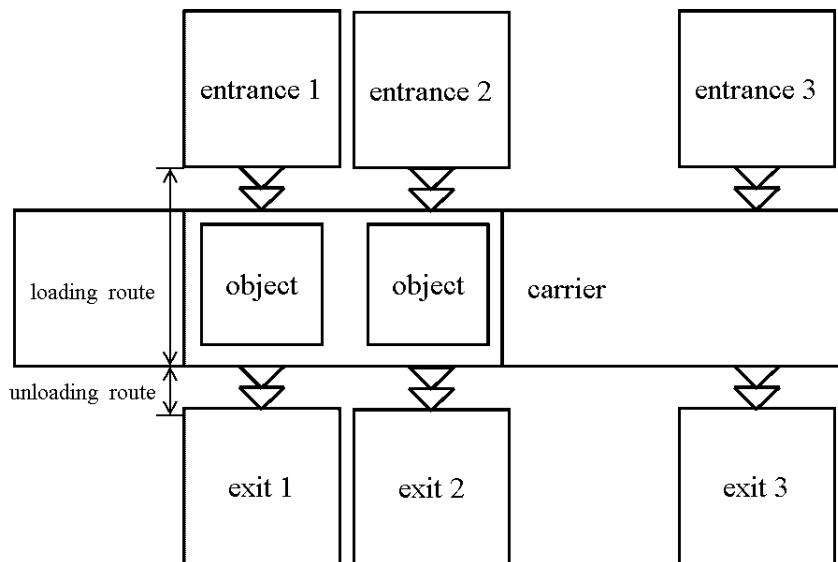


Figure 4.72: Functional principles of a paired shuttle

Here too, each connection must be allocated a position analogue to the simple shuttles. The user must take the utmost care that each connection is only served by one of the two vehicle tables. Correspondingly for each connection under **Table** the kind of table (left or right) must be selected, where **left** means the same as „nearer to the determined zero point“.

4.24.1 Global Strategy

The principle strategy when steering a paired shuttle is, as fast as possible, to carry out a double cycle not only for right-of-way- but also for distribution strategies, otherwise a single cycle is possible. An exception is the **global destination control** which corresponds to a destination targeted distribution with other modules and is valid for all distribution rules, for double cycles 1 and 3 and for single cycles.

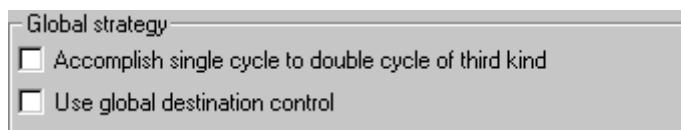


Figure 4.73: Global strategy

If the user selects this strategy, the permissible object types for each exit can be defined. When setting the destinations of objects, global destination control has priority over the basic strategy of carrying out a double cycle, i.e. global destination control must not be interrupted because of a possible double cycle.

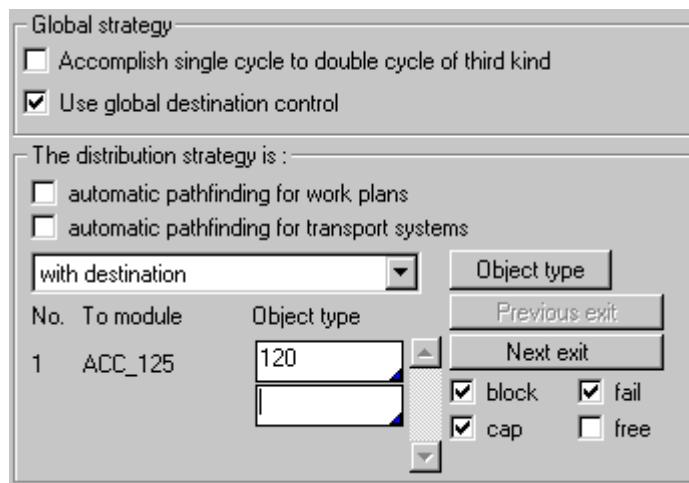


Figure 4.74: Global destination control

If no couple formation is possible, the control is made by the strategies of the single cycles. Nevertheless a single cycle can be extended to a double cycle of the third kind, if a further entrance is first reached before the delivery, so that the double cycle is made possible. For this the switch **Accomplish single cycle to double cycle of third kind** is to be activated.

4.24.2 Double Cycles

Furthermore, the user must define all the connection pairs which are likely for one of the three double cycles. First, one of the three alternatives must be selected under **Control of double cycles** after which the relevant list of connection pairs appears which are then completed by the user as follows.

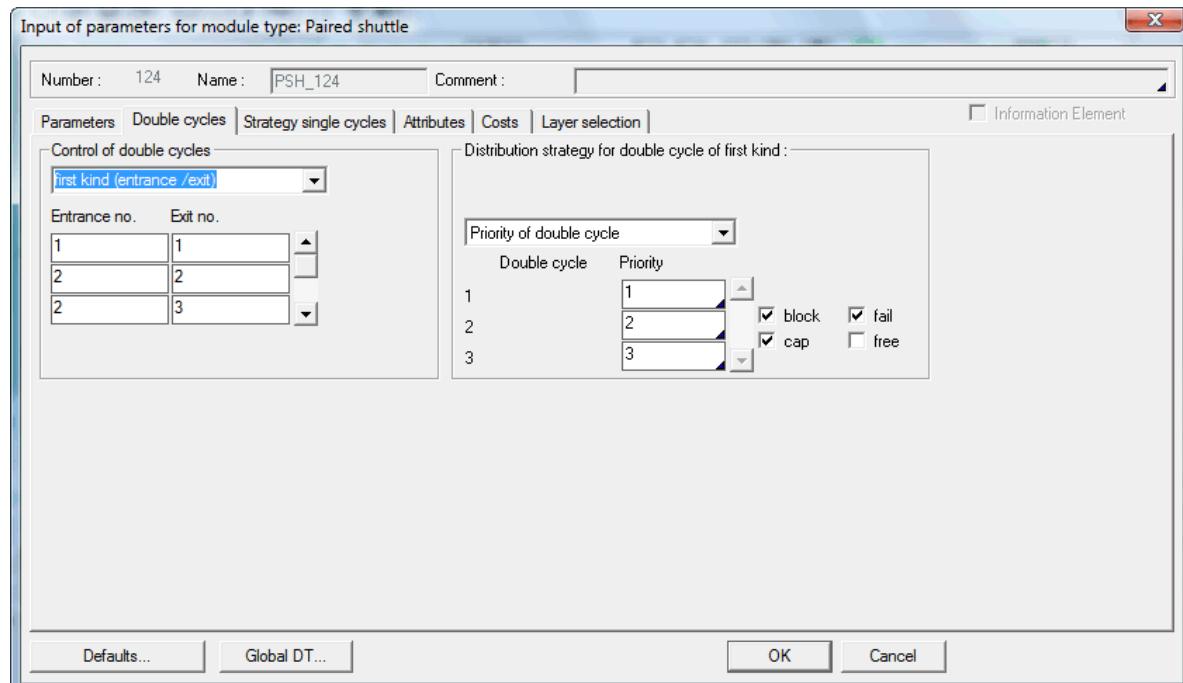


Figure 4.75: Parameters of double cycles

For double cycles of type 1, the pairs to be entered consist of entrance and exits which are next to one another, whilst double cycles of types 2 and 3 require all entrance and exit pairs



that are next to one another to be entered. Here it is especially important to observe that each connection can only be served by one of the two tables.

If, for example, a paired shuttle has 12 connections, distribution as follows:

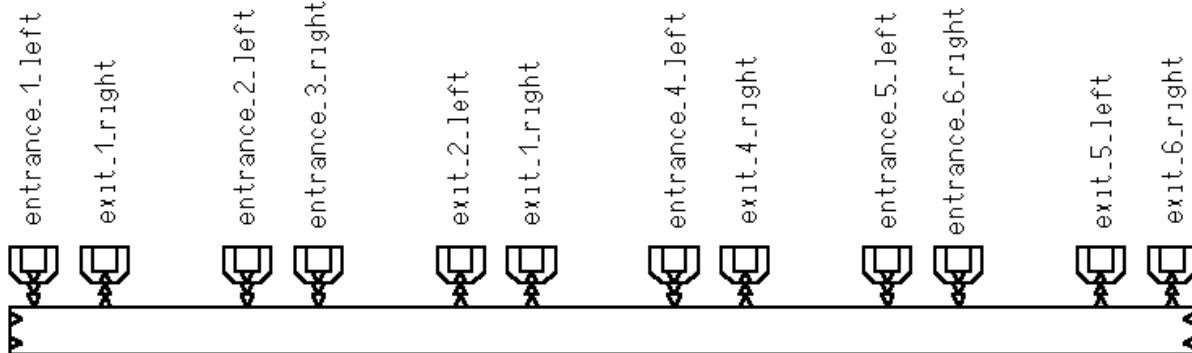


Figure 4.76: Example of a paired shuttle

There are exactly two connection pairs for each double cycle, resulting in the following parameter areas.

Control of double cycles		Distribution strategy for double cycle of first kind :	
<input type="button" value="first kind (entrance /exit)"/>		<input type="button" value="Minimum occupation"/> <input checked="" type="checkbox"/> block <input checked="" type="checkbox"/> fail <input checked="" type="checkbox"/> cap <input type="checkbox"/> free	
Entrance no.	Exit no.		
1	1		
4	4		

Figure 4.77: List for double cycle of first kind

Control of double cycles		Right-of-way strategy for double cycle of second kind :	
<input type="button" value="second kind (entrance /entrance)"/>		<input checked="" type="checkbox"/> Objects are allowed to stop <input type="checkbox"/> Check entrance <input type="checkbox"/> Priority last entrance <input type="button" value="FIFO"/>	
Left table	Right table		
2	3		
5	6		

Figure 4.78: List for double cycle of second kind



When no combination of the left and right table is possible, no double cycle can be defined. This is, for example, true when only left tables are defined in the position list.

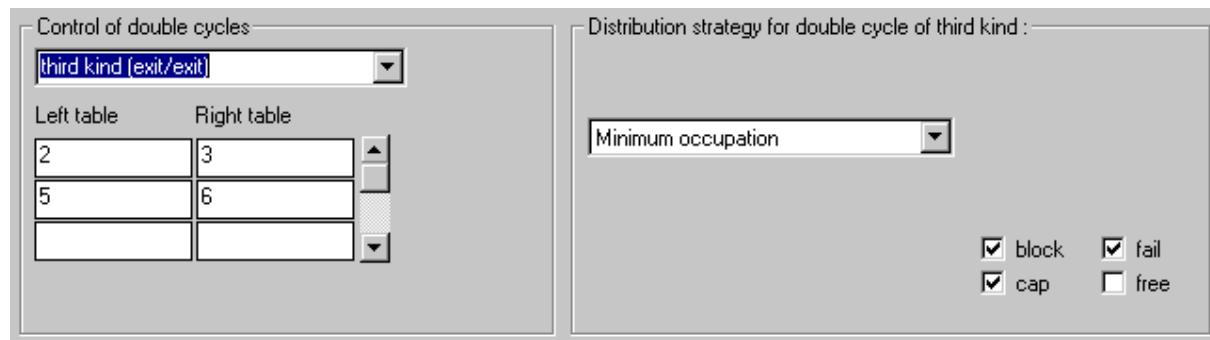


Figure 4.79: List for double cycle of third kind

After entering the respective connection pairs, the corresponding distribution and/or right-of-way strategy can be selected if there are several alternatives for each of the double cycles as in the case present. After clicking the relevant bar, one of the strategies can be selected. Thus one specifies the double cycle, which is to be used, if several are possible. A special strategy is the priority of the double cycles, which exists only for the paired shuttle.

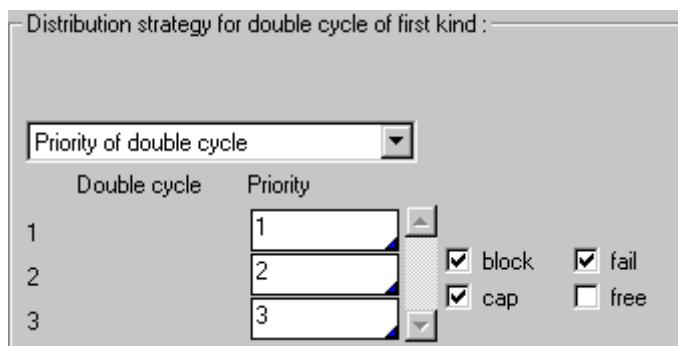


Figure 4.80: Priority of double cycles (only paired shuttle)

4.24.3 Functionality

As two objects can be conveyed simultaneously, object pick-up and the object delivery must be differentiated. An object can be picked up when the vehicle is empty or only occupied by one object whereas it can be delivered when both of the vehicle tables or only one table is occupied.

Case 1: The paired shuttle is empty.

First a check is made whether there is an entrance pair, with which a double cycle of type 2 can be carried out. If so, one of the entrance pairs is determined by means of a right-of-way strategy. Only when that is not possible, is an attempt made to carry out a single cycle. An entrance from which an object is to be picked up as a single cycle is fixed by means of the **right-of-way strategies for single cycle** defined after clicking the relevant bar.

Case 2: The paired shuttle is occupied with one object.

In this case, an attempt is also made to carry out a single cycle by looking for connection pairs, on which, parallel to the delivery of the object another object can be picked up. If such a double cycle is not possible, the user can influence the further transport of the vehicle by determining whether a completion to double cycle 3 is



desired. If this is the case, the entrance, from which the second object is to be picked up, is determined by means of the right-of-way strategy for a single cycle. A prerequisite is that a suitable object is standing by at one of the entrances and that afterwards, delivery can be made to a connection pair for double cycle of type 3.

Otherwise the object is delivered as a single cycle where the destination is determined analogue to the right-of-way strategy with a **distribution strategy for single cycle** which is to be defined separately.

Case 3: The paired shuttle has two objects.

If a double cycle of type 3 is possible, the exit pair with the relevant distribution strategy is determined. Otherwise, one of the objects is delivered as a single cycle, where that object with the shortest route to its destination is given priority.

Finally, case 2 exists again.

With an object transport, the relevant entrance link is released as soon as the object is completely standing on the vehicle table. After transport, a report is made to the successive module as soon as the unloading route has been completed. If the successive module is ready for acceptance, the object is delivered and after release of the exit link, the vehicle is then ready for the next transport. Analogue to the shuttle, the successor module is checked for acceptance before loading, normally excluding a blocking of the paired shuttle. However, a blockage can occur if the successor reaches a state before the arrival of the object preventing delivery such as failure or standstill of the conveying track. In such a case, the vehicle remains blocked until the object can be delivered. Otherwise regarding the right-of-way rules only for objects, which probably can be delivered, and successor modules ready for acceptance concerning distribution strategies are considered.



5 Transport Systems

5.1 Introduction

This chapter deals with the modeling and simulation of **transport systems** (TS). Due to the increasing complexity of and the demand for higher flexibility in MFS which can only be met by the increased use of **transport vehicles** (TV) these vehicles have been gaining steadily in importance.

The transport system level as supplied by the DOSIMIS-3 simulator allows a quick and easy generation of transport system modules which can be combined with the other DOSIMIS-3 modules and their respective strategies without any restrictions. Furthermore, the user has the option to operate the transport system in the fully automatic mode which means that an automated guided vehicle (AGV) under certain conditions autonomously selects the shortest route to its destination as it enters the junctions in the system. Vehicles are scheduled automatically, i.e. the simulator assigns vehicles to orders and vice versa by itself in compliance with strategies defined by the user. The automatic routing saves the user having to specify the path through each individual junction.

To give the user a clear overview of the transport system level description, this chapter covers some topics that will be dealt with again in other chapters dedicated to the other DOSIMIS-3 modules, thus explanations of the strategy level for vehicle scheduling and of the corresponding statistics results are included here in addition to the description of the transport system modules. The user will find information on individual orders and vehicles that go beyond the standard statistics.

5.2 Modules of a Transport System

The modules utilized for the representation of transport systems are:

- Track
- Loading station
- Unloading station
- Workstation and transport system
- Service station
- Crossings with additional function (See chapter automatic routing).

These transport system modules are comparable to the following modules

- Accumulation conveyor
- Assembly
- Disassembly
- Workstation
- Crossing

But they allow the user to model a complex transport system in a quick and simple manner and, at the same time, to consider special functions such as energy restrictions.



Just as buffer sections can be initialized with objects, a track can be initialized as well. This is the only way to direct vehicles to a section because a source can only generate standard objects, e.g. loads for transport systems, which differ from transport vehicles. Even though normal objects can be loaded with other objects in assembly units like loading a vehicle in an loading station, the decisive difference is the way loaded vehicles behave in one of the transport system-modules.

The transport parameters are designed to assign to each loading, unloading and transport system station a specific destination code. The transport scheduling then allows for each vehicle to be directed to any destination in the system. Only vehicles with a destination that matches the destination code will be processed automatically, i.e. without further specifications, by the above three modules. In the fully automatic guidance mode, every vehicle will find the path to its destination; otherwise the user has to arrange the correct route by setting suitable parameters of a destination-specific distribution in the intersection. Besides automatic routing, transport system control, comprised of vehicle scheduling and energy monitoring, is part of the fully automatic operation. Vehicle scheduling is activated when a load is reported for loading at a loading station or when a vehicle is reported for loading within the scope of energy monitoring. More information on this subject is to be found in the chapter on transport strategy levels.

5.2.1 Track

In addition to conveying objects and vehicles, another block section function called **TRACK** is used to generate the vehicles necessary for a transport system. This is done by initializing the track at the start of the simulation.

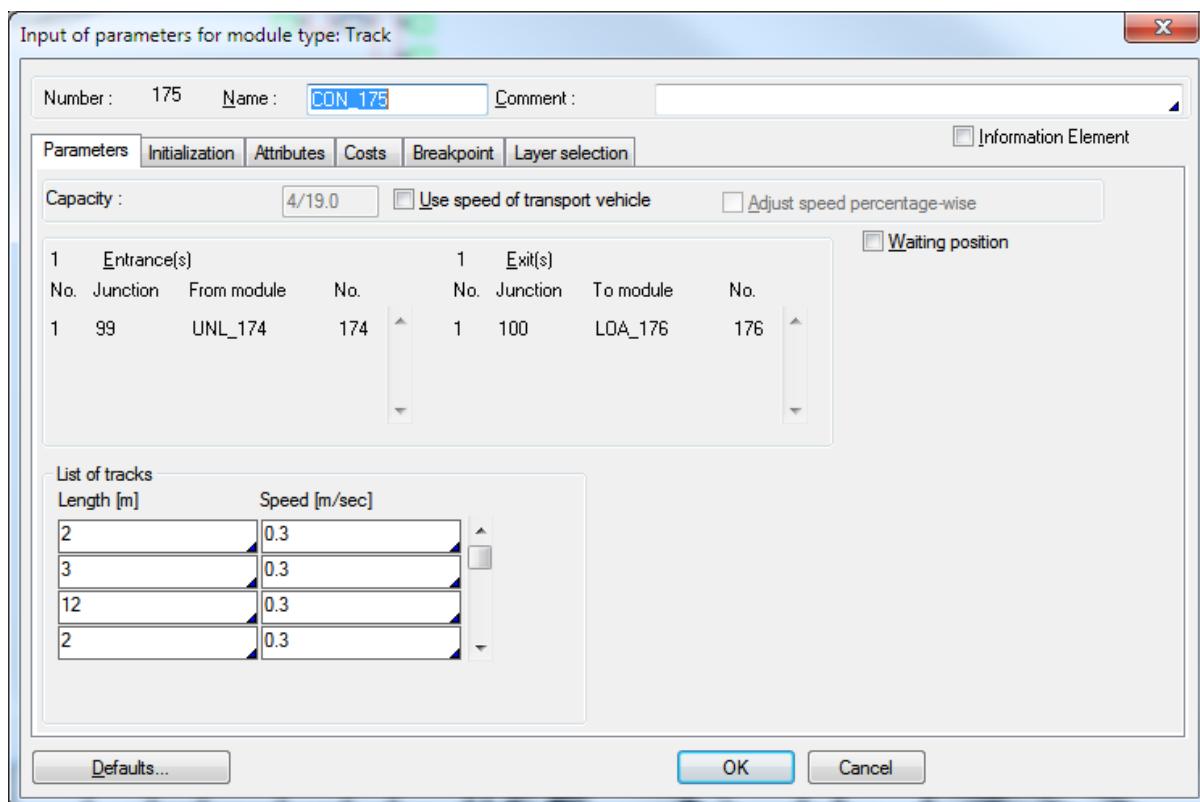


Figure 5.1: Parameters of a track



The parameter dialog records the standard parameters of **Name** and **Number**. The vehicle itself can be initialized as required. To do this, the corresponding dialog is set and the number of vehicles to be initialized is defined. At this point, each vehicle is assigned a type, which at the same time is the first primary destination to be reached (cf. [transport strategies](#)).

The track can be split into as many segments as needed, the number of which is entered. These track segments are comparable to segments of other tracks. Segments of a track, however, can be of variable lengths and of variable conveying speeds. Should the number of segments exceed the number of initialized vehicles, the vehicle would be located in the front-most one closest to the exit. There can be no more vehicles in a track than it has segments.

If the speed of the vehicles is to be considered, a correcting factor can be indicated in the second column (percentage). If the switch **Adjust speed percentage-wise** is active, this determines the portion by which the speed is corrected in the individual segments. Thus slow distances covered or speed decrease in curves can be modeled, for example.

If the switch **Waiting position** is activated, all vehicles, which enter this track, are treated as waiting vehicles with reference to the statistics. This happens independently of the actual condition of the vehicle (loaded, empty...). Note: The vehicle drives through the track as usual. Thus as if waiting position only makes sense in combination with a following work station, which is parameterized as waiting position.

In a track with several segments, the preceding module is only released if an object has reached the end of the first segment.

5.2.1.1 Initialization

The vehicles can be initialized when needed. In addition the switch **Use initialization** in the sub-dialog *Initialization* is activated and the number of vehicles which can be initialized is defined. Here also, a type is assigned to each vehicle. If no transport control is used, it simultaneously deals with the first primary destination which can be achieved by the vehicle (see [transportation strategies](#)).

If more partial tracks than initialized vehicles are present, then the vehicles are in the front segments, i.e. those closest to the exit. More vehicles than segments cannot be initialized in a track.

In the initialization of tracks, besides the number of vehicles, a speed can also be assigned to each vehicle. This can be used when driving through tracks for the determination of the duration of driving through.

Use initialization :	<input checked="" type="checkbox"/>				
The module is initialized :	<input checked="" type="checkbox"/>				
Object type	Quantity	Speed	Acceleration	Capacity	
99	2	0.5	.2	4	<input type="button" value="▼"/>
					<input type="button" value="▲"/>

Figure 5.2: Initialization of the track



In the initialization of a track beside the number of vehicles also a speed be assigned to each vehicle. This can be used when driving through a track for the determination of the driving through duration.

If a positive **acceleration** and a positive speed is indicated, then the acceleration is consulted with the computation of the driving through time through one segment of the track. If exiting from a segment inside the track is not possible (not the foremost), then the vehicle needs an additional time for braking. Subsequently, the vehicle stands and must from there, if drive on is possible, be accelerated once again.

With **capacity** the maximum number of load can be indicated. In order to also use this capacity, a multiple loading must be parameterized in the loading station accordingly.

5.2.2 Loading Station

Within a loading station, an object on stand-by at the station entrance (entrance 2) is loaded onto the empty vehicle and assigned a delivery destination, i.e. the destination code of the unloading station to be supplied. In compliance with the chosen scheduling strategy, the destination is either assigned to match the type of load or not. A vehicle will only be cleared for loading when the designation codes of the destination and of the vehicle destination, i.e. the vehicle type, match. Any other vehicles and objects will be forwarded at the specific speed of the station.

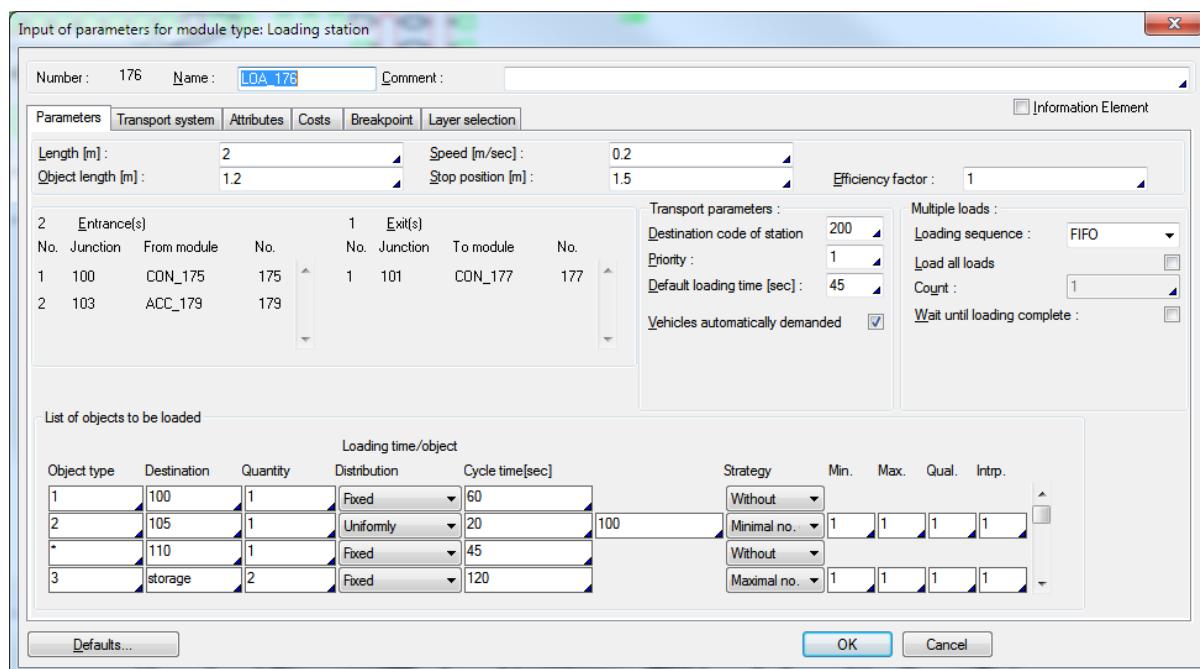


Figure 5.3: Parameters of a loading station

ATTENTION! When connecting the modules you have to pay attention that entrance 1 is to be connected with the module, from which the vehicle reaches the loading station. Accordingly the entrance 2 is the entrance, from which the load reaches the module.

Standard parameters of **Length**, **Speed**, and **Object length** (i.e. or **vehicle length**) and a **Stop position** must be entered. Figure 5.4 illustrates the significance of the stop position. An incoming vehicle no longer automatically advances to the module exit but only to the



specified stop position which may not be located far enough inside the module to allow the vehicle to clear the module boundary with its entire length.

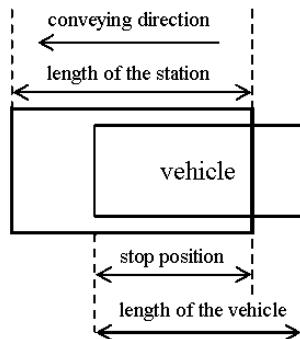


Figure 5.4: Stop position

The destination code required for each station to make identification of vehicles to be loaded possible is entered as **Destination code of station**. The wording emphasizes that only vehicles of the type specified here are cleared for loading. Furthermore this area includes data on the **Default loading time**.

Transport parameters :	
Destination code of station :	200
Priority :	1
Default loading time [sec] :	45
Vehicles automatically demanded <input checked="" type="checkbox"/>	

Figure 5.5: Global strategy of the loading station

This information would be sufficient for parameters in the fully automatic control mode. The vehicle will have the exact type of load object assigned to it as a destination and the loading speed will not depend on the object but is identical for all types to be loaded (cf. transport strategy level).

If the switch **Vehicles automatically demanded** is active, the scheduling of the vehicles is made by a transport control. Otherwise the vehicles must be scheduled with the help of subsequent dispatch or decision tables.

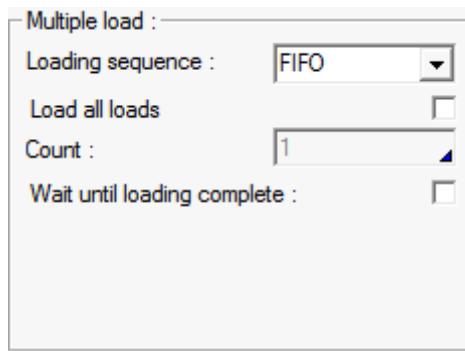


Figure 5.6: Multiple load in a loading station

In the selection of **Loading sequence** (sorting) several alternatives are available with the multiple loading of loads. The sequence of unloading can be defined here.

- **FIFO** : The sequence of the load on the vehicle is the same as arrival of the load at the station. Thus the destination of the load, which was first taken up is encountered first.
- **LIFO** : Loading of the load takes place in reverse order of the arrival of the load. Thus the destination of the load, which was taken up at last is encountered first
- **Minimal route** : The sequence of loading takes place according the distance to the destination of the objects. Thus the object with the destination nearest to the loading station is encountered first.

If the switch **Load all loads** is activated, that many loads are taken up, as indicated in the field **Count**. It remains unconsidered whether the type of load changes. The destinations are determined in accordance with the list of the objects which are loaded. First the destination of the load is approached, which was first taken up. First the destination of the load which stands in front in accordance with the load assortment is to be reached. At the unloading station it can be specified whether all objects intended for this destination will be unloaded.

If the switch **Waiting until loading complete** is activated, the vehicle remains in the station until the indicated the number of loads were taken up or the capacity of the vehicle is reached. This waiting procedure can be only broken off then by decision tables (by blocking the junction). Otherwise the load procedure is broken off, if after a load procedure no further object waits at the load entrance of the station.

In the area **List of the objects to be loaded** the possibility exists to set the new **destination** of the vehicle and to set the load duration as a function of the type of the object which is to be loaded.

List of objects to be loaded						Strategy	Min.	Max.	Qual.	Intrp.
Object type	Destination	Quantity	Distribution	Cycle time[sec]						
1	100	1	Fixed	60		Without				
2	105	1	Uniformly distribu	20.0	100.0	Minimal no.	1	1	1	1
*	110	1	Fixed	45		Without				
3	*	2	Fixed	120		Maximal no.	1	1	1	1

Figure 5.7: Parameters of loading

When the value in the field **Quantity** is larger than 1, several objects are loaded on the vehicle. This occurs as long as objects of the same type are at the input and the connecting



node is not blocked. Blocked means that the node is not blocked and the module is not disturbed.

When the type of object is a wildcard (*), this list applies for all objects, which are not explicitly specified. When the destination is a wildcard, then the vehicle receives the type of the charge as destination. Otherwise the value, which is indicated there, is taken.

Each loading time can be defined by an individual distribution function. Worker requirements as well (see work area) can be defined.

5.2.3 Unloading Station

The **UNLOADING STATION** has one entrance and two exits. One of the exits (exit 2) is for the forwarding of unloaded objects, and the other (exit 1) for unloaded and not unloaded vehicles. An unloading station will only unload a vehicle when the destination code of the vehicle matches its own code. Empty vehicles or vehicles, which do not match the destination code, will cause to be passed on at the specific conveying speed for that station.

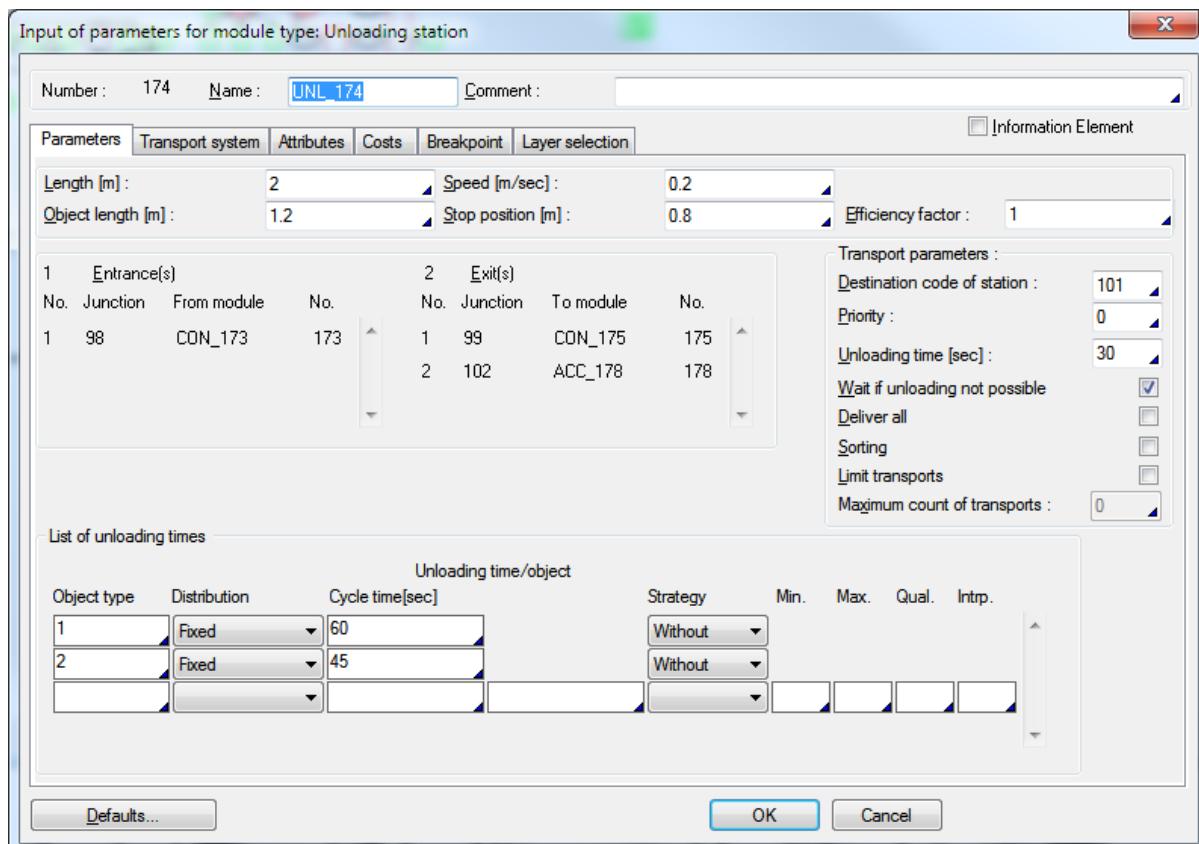


Figure 5.8: Parameters of the unloading station

ATTENTION! When connecting the modules you have to pay attention that exit 1 is to be connected with the module, to which the vehicle leaves the unloading station. Accordingly the exit 2 is the exit whereto the load leaves the module.

The destination code of an unloading station is entered in the parameter dialog under **Destination code of station**. Only vehicles with the destination specified here will be processed. The option to unload manually can be additionally specified. **Wait if unloading not possible** is an option available here. This means that the vehicle can leave the station, if



the successor does not accept the load. Then it can be unloaded at a station with the same destination code or drive a loop to come back later, thus not blocking vehicles behind. Otherwise the unloading starts as soon as the vehicle has reached the unloading position. But the process only begins when the object can also be submitted. Thus, the following module must have enough space to include the object too. Up to this time, the module is blocked.

An unloading action will take the load object automatically off the vehicle. As soon as the action is concluded, the type of the load object can be altered and any other type can be entered. Furthermore and independent of the fully automatic mode, a type-specific unloading time can be declared for each load object. Otherwise the standard **Unloading time** is in effect.

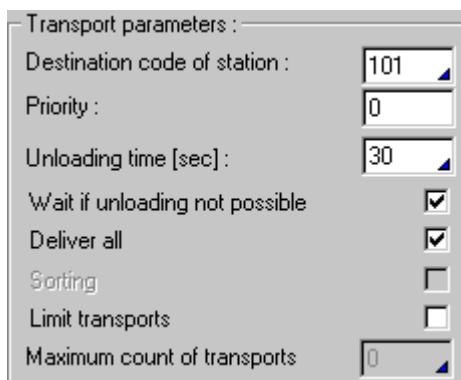


Figure 5.9: Global strategy of the unloading station

If the switch **Wait if unloading not possible** is active, the vehicle stops in the station, even if the delivery cannot take place. Otherwise the vehicle drives out of the station without change of the destination. It tries then to reach the next station with the same identification.

If the vehicle delivers several loads it can be specified here, how to proceed. **Deliver all** means that all objects will be unloaded, independently of whether the unloading station is the final destination for all loads. If this is not selected, it can be determined via **Sorting** whether all loads, which are intended for this destination, will be unloaded. If sorting is not possible, only those objects are unloaded, which are intended for this station in accordance with the order of loading.

By **Limit transports** it can be guaranteed that only a limited number of vehicles are on the way to this station.

Further disposition can be made in the sub-dialog Transport system. So, for example vehicles, which do not receive a job after the discharge process, can be taken out from the system into a fictitious bypass, so that they do not block following vehicles (see also Vehicle-Bypass).

Independent of a fully automatic control for each load object, the possibility of defining a type-dependent unloading time (see also distribution functions) and personnel requirements (see work areas) for this activity exists.



List of unloading times									
Object type	Distribution	Cycle time[sec]	Strategy	Min.	Max.	Qual.	Intrp.		
1	Fixed	60	Without						
2	Fixed	45	Without						

Figure 5.10: List of unloading times

When the type of the object is a wildcard (*), then this list is applied for all objects, which are not explicitly specified.

5.2.4 Workstation with Transport Systems

In order to process the charge of a vehicle in a workstation, two versions are intended. On the one hand, the vehicles dispatched explicitly to this station are handled. In the field **Destination code of station**, the user determines which vehicles can only be served. A vehicle cannot be processed in a workstation, only the objects the vehicle has loaded.

Input of parameters for module type: Workstation

Number : 209	Name : WST_209	Comment :	<input type="checkbox"/> Information Element
Parameters		Transport system Attributes Costs Layer selection	
Length [m] : 1	Speed [m/sec] : VMax	<input type="checkbox"/> Forward control	<input type="checkbox"/> Consider all work procedures
1 Entrance(s) No. Junction From module 1 291 SOU_387		1 Exit(s) No. Junction To module 1 128 WST_208	
Transport parameters : Destination code of station : Priority : 1 Limit transports Maximum count of transports 0 Handle load separately Handle all loaded vehicles			
Configure <input checked="" type="checkbox"/> Worker <input checked="" type="checkbox"/> Transport parameters <input checked="" type="checkbox"/> Work procedures <input checked="" type="checkbox"/> Set-up <input checked="" type="checkbox"/> New object type			
Work procedures Name : ASS_1 No. : 1/2 New Delete Comment :			
Distribution of working time Normally distributed Edit working time DT Delete DT Employment of workers			
Object type	Distribution	Mean[sec]	Deviation[sec]
1	Normally distribu	work	5
10	Normally distribu	work	5
2	Normally distribu	80	5
-	Normally distribu	80	5
Strategy Min. Max. Qual. Intrp. Maximal no. 1 3 2 32000			
Initial object : 1 Kind of processing : Random New object Probability 1 85 10 15			
Set-up times Fixed From object To object Distribution Cycle time[sec]			
1	10	Fixed	setup
10	1	Fixed	setup
2	-	Fixed	60
-	-	Fixed	60
Strategy Min. Max. Qual. Intrp. Maximal no. 1 3 4 32000 Without Maximal no. 1 2 2 32000 Maximal no. 1 3 4 32000			
Defaults...		Global DT...	
		OK	Cancel

Figure 5.11: Parameters of the workstation

Even when such vehicle loads that were not explicitly dispatched to this station are to be processed, the switch **Handle all loaded vehicles** is to be activated. Then all those vehicles are managed which reach this station. Otherwise only the vehicle that was explicitly dispatched with the help of the identifier to this station is managed.



If the load object of a vehicle is to be processed, then the station is to be parameterized similarly to a processing station. All specifications then refer to the load object. However the user has to make certain that the vehicles with appropriate destination identifiers carry also only those load objects that are specified in the processing list. Otherwise the vehicle drives through although the destination identifier of the station is correct.

Since vehicles can carry several charges, which can vary depending upon situation at the loading station, the work time can be defined per charge. For this, the switch **Handle load separately** from the vehicle parameters is to be activated. Otherwise, the operating time per vehicle is assumed.

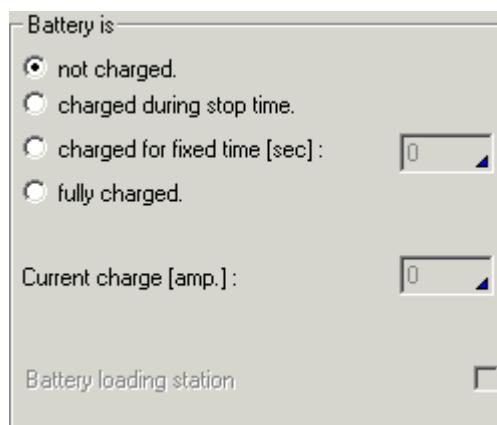


Figure 5.12: Battery parameters of a workstation

If the load object is to be only processed, then in the section regarding battery charging in the sub-dialog *Transport system*, the field **not charged** is to be selected.

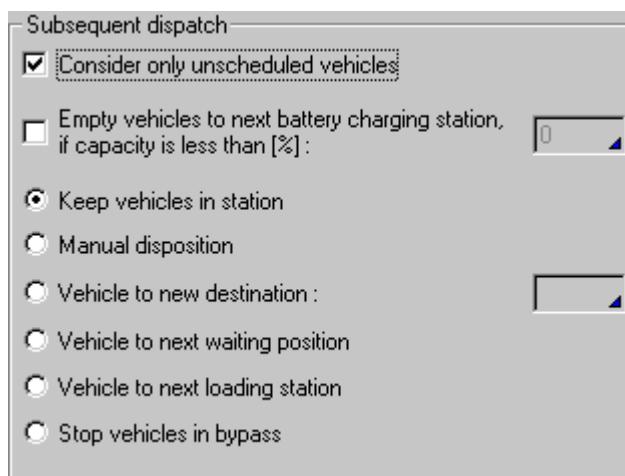


Figure 5.13: Parameters of the subsequent dispatch of a workstation

In the case of the fully automatic control like also with the remaining vehicle modules, the field of the **new destination** can be left free. Otherwise in the field **New destination** the next goal of the vehicle must be defined.



5.2.5 Service Station

The battery-charging procedure can take place in every one of the three processing stations (loading station, unloading station and workstation) of the transport system. For this, in the area **Battery is**, the type of the battery-charging procedure has to be defined.

If the vehicles in a transport system should be subject to energy monitoring, then they must be charged at certain times. For this purpose, in the input area, it is determined whether and in which way the battery of a vehicle is to be charged. It can be **fully charged** either with the load rate of the **Current charge** or be **Charged for fixed time**, or only **Charged during stop time** for handling of the load object. Volume loading of the battery can take place also during load object handling. If necessary the vehicle in the station waits, until it again possesses the maximum energy capacity.

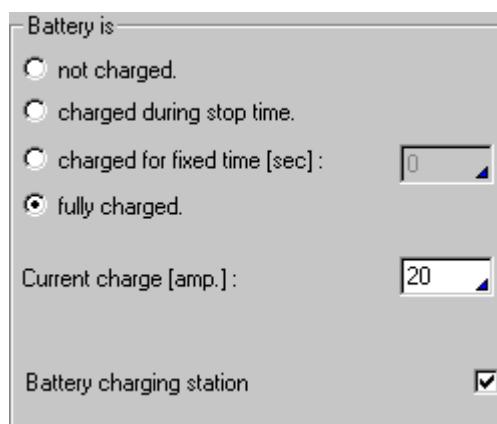


Figure 5.14: Battery station

If vehicles are to be dispatched explicitly to a loading station, then the switch **Battery charging station** is to be activated and a **Destination code** is to be assigned to the station for planning. Otherwise, the loading process takes place only if the vehicle achieves this station in the context of a transport.

In the case of the fully automatic control like also with the remaining transport system modules, the field of the manual destination can be released. Otherwise, **New destination code** of the vehicle must be entered in the field.

Vehicles, which do not receive an order after the battery-charging procedure, can be taken from the system by a fictitious **bypass**, so that they do not obstruct following vehicles (see also the [Vehicle - Bypass](#)).

5.2.6 Waiting Position

Vehicles can under normal circumstances be also scheduled to a waiting position. This can be to each workstation that possesses a destination-identification. However, no processing of objects may be parameterized. Instead of parking vehicles now in a bypass, it can be specified in the subsequent dispatch that a waiting position is to be headed for. Vehicles, which reach this station, are set into the waiting mode. These vehicles can be activated with the next schedule again. While driving into the waiting position a scheduling is not possible. The tracks at the entrance of a waiting position can be used as an extension of the waiting position. For this the switch *waiting position* is to be activated in the track.



5.3 The Transport-Strategy Level

The main function of strategies in transport systems is to establish the routing of the vehicles by assigning destinations to them. The assigned destination has to match the code of a transport system module, either loading station, unloading station, or workstation. Under fully automatic control, a vehicle is routed to its destination by automatic routing. Otherwise, the user has to implement a destination-oriented distribution strategy that ensures the vehicle's arrival.

A destination is assigned to a vehicle either manually in the fields provided in the transport system modules or with fully automatic control. The latter is always implemented when the fields mentioned are left blank. A third alternative allows the user to combine the first two options for destination assignment by leaving the field **New destination of vehicle** blank in some unloading stations, but filling in others.

5.3.1 Fully Automatic Mode

The fully automatic mode features both automatic routing and transport control with its components energy monitoring and vehicle scheduling. Energy monitoring ensures that a vehicle is recharged in time at a battery charging station. Vehicle scheduling handles the assignments of vehicles and orders.

The following four basic rules apply for the parameterization of models with fully automatic control:

- The fields for manual destination assignment must be left blank at the loading stations, unloading stations, and workstations.
- Only objects, for which destinations exist, are to be loaded.
- Destinations must be listed only once unless they are located along an unbranched track.
- At crossings (crossings, distributors, etc.) **automatic path finding** is to be selected.

No.	To module	Priority	block	cap	free
1	ACC_126	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	ACC_127	2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	ACC_128	3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 5.15: Automatic path finding

- Further information on the behavior with combination of the automatic path finding is to be taken from the section [automatic routing](#).

5.3.2 Transport Scheduling

The primary task of transport scheduling is to assign orders to the vehicles. Depending on the situation in the system, scheduling reacts in one of two ways. If the quantity of orders exceeds the supply of empty vehicles, an order is selected to match a vehicle. Otherwise, a vehicle is selected to match an order.



Order means that an object is waiting for loading onto a vehicle. The destination code of the station that an empty vehicle is heading for is the **primary destination** or **pick-up destination**, whereas the code of the station designated to unload at is the **secondary destination** or **delivery destination**. In this manner, pick-up as well as delivery destinations are specified under the fully-automatic control mode as soon as an order is generated, since the drop-off destination will be automatically specified together with the type of the load objects to be picked up.

Furthermore, scheduling is treated as an order on the situation of a vehicle requiring recharging after it has unloaded an object. The menu *Model* has the item **Transport Strategy**.

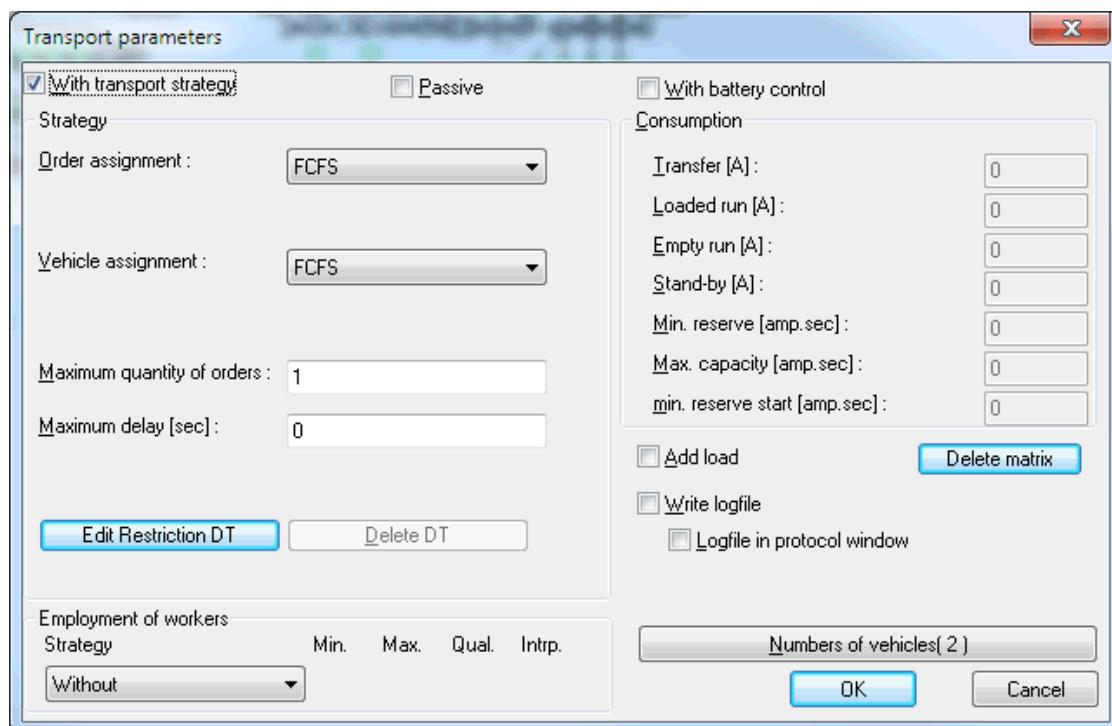


Figure 5.16: Transport Strategy

Here can be determined by the user, which strategies in the case of the vehicle job allocation are to be supposed. Moreover, the number of vehicles that are initialized in the system can be seen on the button **Number of Vehicles**. These modules are selected by clicking the button, so that these can be parameterized after leaving the dialog.

If an order is to be looked for on the basis of a vehicle, the following strategies are available:

- | | |
|--|--|
| FCFS | This means first come first serve . The order which has been put on hold for the longest time will be assigned to the vehicle declared as empty. |
| Minimal route | Selects the order with the nearest pick-up point to the current position of the vehicle. |
| Priority primary destination | Chooses the order with the lowest pick-up destination code number. |
| Priority. secondary destination | Selects the order with the lowest delivery destination code number. The lower the code number for either one of the last two strategies, the higher a priority it indicates. |



Self-defined	Means that the user has developed a personal strategy at the programming level.
---------------------	---

If a vehicle is required for an order, the strategy options are reduced to pick-up and delivery destinations and no longer affect the scheduling. What remains is:

FCFS	Which favors the vehicle which has been standing empty the longest
Minimal route	Gives the assignment to the vehicle nearest to the pick-up point for the order
Self-defined	Means that the user has developed a personal strategy at the programming level.

The function circuit of transport systems control is illustrated in the flowchart in Figure 5.18: The transport . The circuit is entered at the beginning of a simulation either as one of the vehicles initialized on the track reports in ready for the next run or as vehicles that are screened for availability for an order. In both cases, an effort is made to assign a vehicle or an order employing one of the above strategies. If this succeeds the particular vehicle is directed towards the assigned primary designation, if not, two cases are distinguished:

If no vehicle is found available to make the pick-up, the order will be entered into a line of pending orders which contains all those orders still waiting to have vehicles assigned to them. This list is stored in a file *.aws. An order remains in this list until it has a vehicle assigned to it that has reported a ready status.

A vehicle that has reported empty status upon completion of either off-loading or recharging its batteries will be temporarily decommissioned in the absence of a new order. This action will cause it to either remain at its current position until a new order emerges or to move to a newly -assigned alternate work station and wait for an order there.

Once order scheduling has assigned an order, the dispatched vehicle heads for the uploading station designated as primary destination to take on a load object. It then advances to a single or several work station(s) for load processing or to an unloading station as secondary destination.



5.3.3 Local Transport Control

There are two alternatives. The global transport control manages all vehicles in the system. This is called by the menu Model/Transport strategy. If however different transport systems are contained in a model, this can be provided with local transport controls. For this, there is the transport control in the control pallet. All loading and unloading stations that belong to the same control are to be connected with the control. In addition, the tracks, in which the vehicles are initialized, which are to be managed by this control, are to be connected.

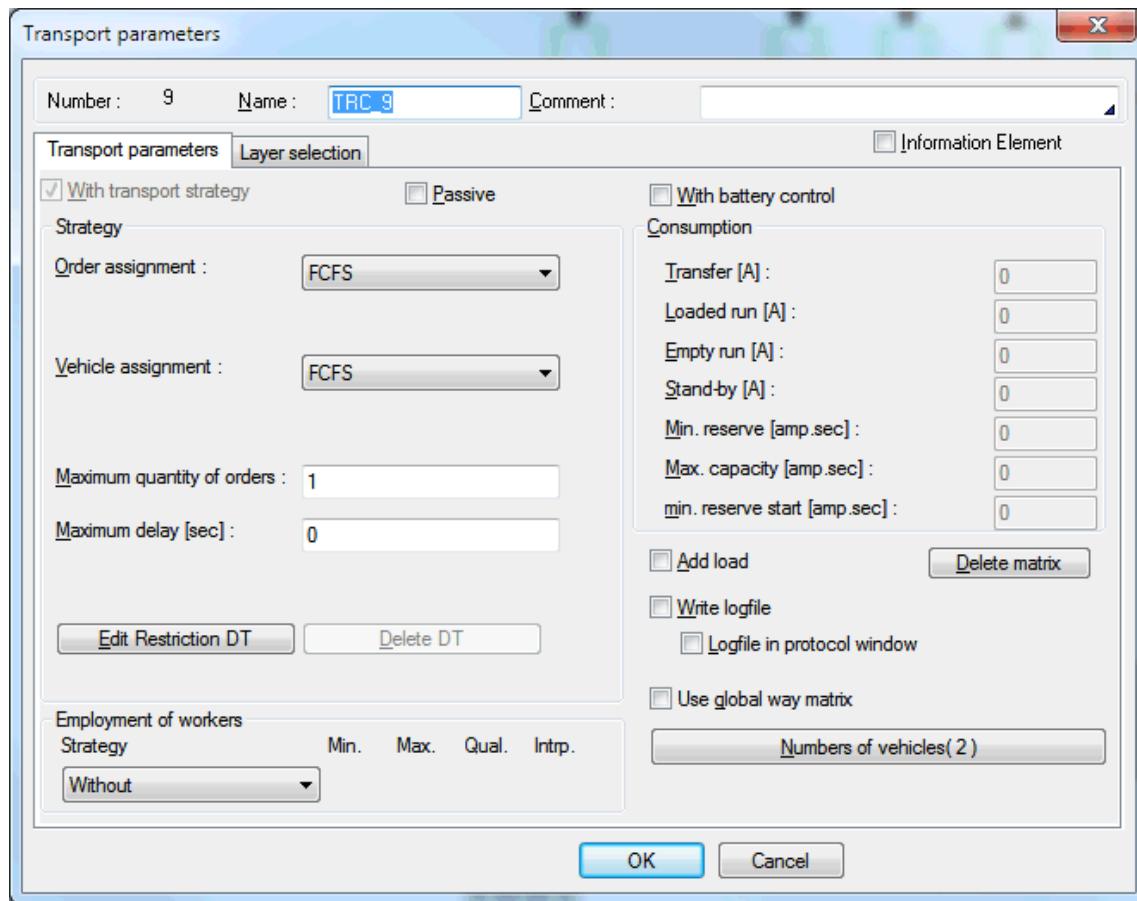


Figure 5.17: Transport parameters

All parameters correspond to that of the transportation arrangement. Only the switch *With transport strategy* is disabled. A local transport control and the global transport control are mutually exclusive. Then the global transport control is to be deactivated by the switch *With transport strategy*.

If different vehicles should be examined with different transport controls on the same route system, a **global way matrix** can use to reduce the number of matrices. Then, a way matrix is produced for the entire model, rather than for each transport-control.

A local transport control and the global transport control are mutually exclusive. Then the switch *With transport strategy* is to be deactivated in the transport-strategy.



5.3.4 Scheduling

Immediately following offloading, the vehicle reports a cleared status to scheduling. Further course of action may be adjusted by energy monitoring depending on whether it is activated or not. Should the battery-charge status have dropped below a value that is to be instituted under energy monitoring, the vehicle is directed to steer to the nearest charging station. Only after this has been done, is the vehicle cleared for a new pick-up order.

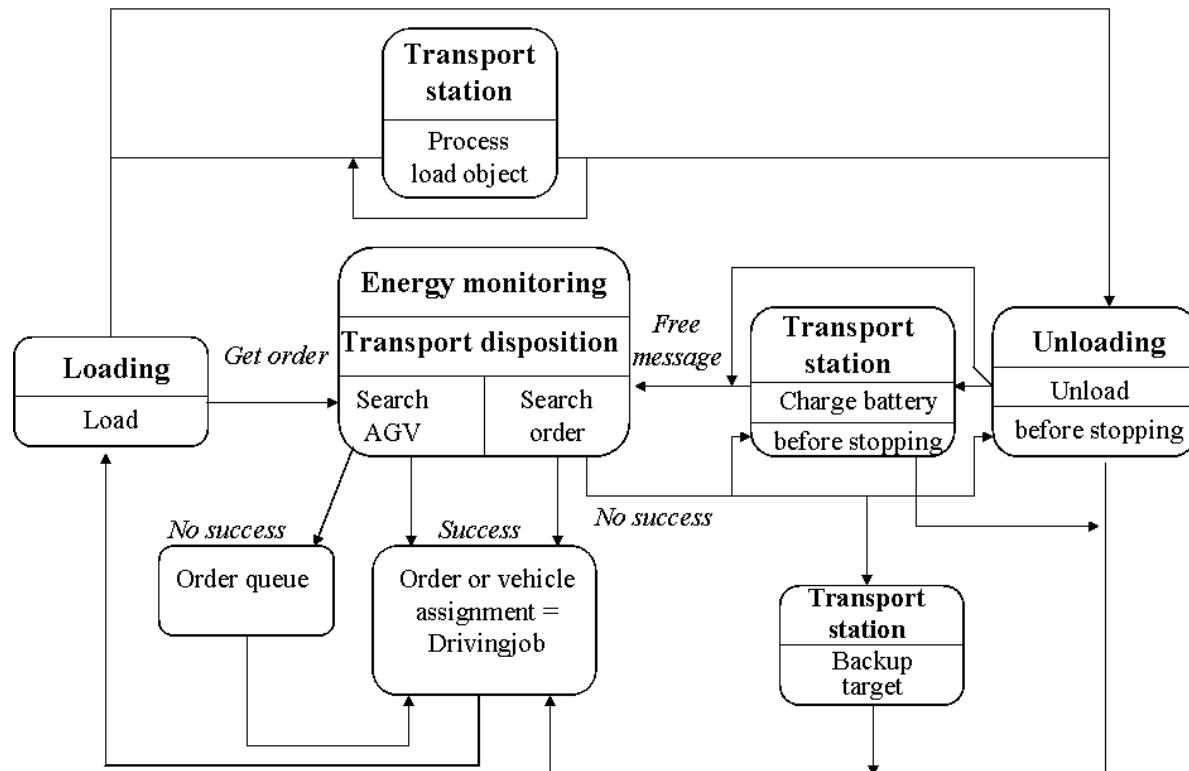


Figure 5.18: The transport strategy

The user can define his own restrictions in a decision table. By clicking on the button **Edit Restriction DT** a new restriction decision table is created, if not previously existed. He can access all vehicle and order data as well as all combinations of vehicles and orders. With the return value of the decision table (*return (value)*) the user can exclude certain combinations. This is the case, if the return value is not equal to zero.

Independent of the mechanism of the transport control, a single vehicle may be assigned several orders at once. This means that scheduling does not only consider those vehicles that are reported clear when it is screening vehicles in compliance with the strategy chosen, but all those that are tasked with less orders than the number specified in the transport strategy menu under the entry **Maximum quantity of orders**. The choice of vehicle is based on the actual position at the time the order is issued. The route of the transport systems strategy as illustrated in Figure 5.18: The transport not affected by the maximum possible number of orders for a single vehicle. With the option **Maximum delay**, the user limits the duration for which an order may be kept on stand-by. When it is exceeded, the order will be assigned to a vehicle automatically regardless of the assignments chosen.

When the switch **Add Load** is activated, vehicles with a capacity of > 1 can be loaded, even they are not empty. This happens when a loaded vehicle reaches a loading station, where the



destination identifier matches his destination. This is possible only if the destination of the vehicle is changed by a decision table (obj.new_destination (value)).

Yet another feature of the transport systems strategies is the option of installing energy monitoring. The corresponding choice is entered by activating **With battery control**. Energy control itself is explained comprehensively in chapter 5.3.7.

5.3.5 Automatic Routing

In order to quickly establish a model for simulating a transport system, the time spent by the vehicle transitioning from the primary to the secondary destinations need to be evaluated thoroughly. A primary destination, also called a pick-up destination, is a loading station. An unloading station is a secondary destination (delivery destination). If there is no order imminent for a vehicle, the battery charging station (workstation in the transport system) will be designated as primary destination.

The task of the automatic routing is to determine and to select the routing from all available track options to a given destination that a vehicle can reach in the shortest time.

No.	To module	Priority
1	ACC_126	1
2	ACC_127	2
3	ACC_128	3

Checkboxes: block (checked), cap (unchecked), fail (checked).

Figure 5.19: Automatic routing

Each route traveled consists of modules with single entrances and single exits as well as crossings with multiple entrances and exits. The running time in modules with only one entrance and one exit is the quotient of length divided by velocity; the running time at crossings is set to be the mean value of the times of all conceivable routings.

The total running time is the sum of all singly-calculated running times. This procedure will be repeated for all track options available towards a specified destination. Among all the total time figures arrived at, the lowest one will indicate the routing with the shortest time in transit and this will be selected.



5.3.6 Subsequent Dispatch

The vehicles of a transport system are dispatched automatically after the processing in a station. In the subsequent dispatch, the strategy **Keep vehicles in station** must be chosen. Automatic disposition can be accessed via the subsequent dispatching. Vehicles can thereby be dispatched to a new destination. Alternatively, the vehicles can be steered to a waiting position or to next loading station. Alternatively, these vehicles can also be parked in a bypass, so that following vehicles are not obstructed.

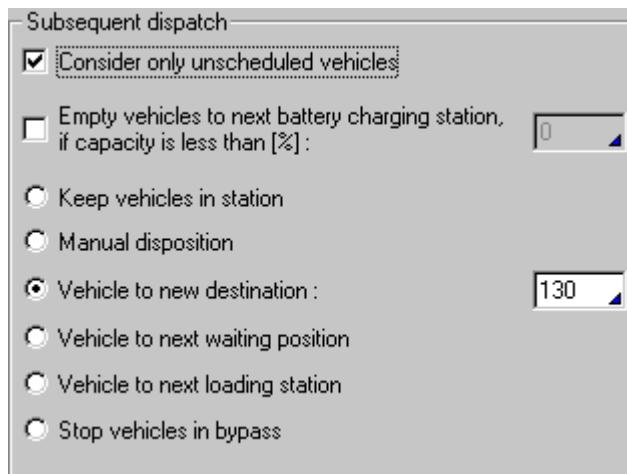


Figure 5.20: Parameters of subsequent dispatch in a transport system

5.3.7 Energy Monitoring

As the vehicles of the transport systems use up energy on their transits, activities etc., they face the real danger of running out of power once their batteries are exhausted. To avoid this situation, the batteries need to be recharged in time. The simulator provides the option of directing the vehicles to a charging station as needed, i.e. the option of monitoring their energy status. An order to recharge can only be issued from the transport scheduling. Therefore, energy monitoring can only take effect in the fully-automatic control mode and then it monitors the energy statuses of the vehicles continuously.

Since a charging order can only be assigned in the context of the transport control, the power monitoring of the vehicles becomes meaningful only within the fully automatic control. The energy level of the vehicles is continuously checked.

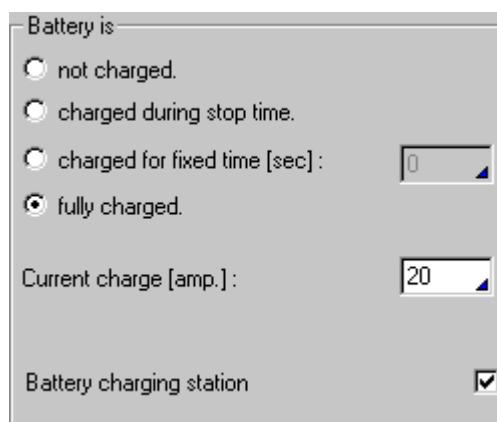


Figure 5.21: Battery-charging parameters in a workstation



There are two possibilities in DOSIMIS-3 of supplying the consumed energy to the vehicle again. The first consists of loading the battery during the time of the loading, load delivery or load handling. In addition, the battery-charging strategy is to be activated in the appropriate stations. Either it is charged as long as the load object is processed (during **stop time**), or for a **fixed time**, or while the vehicle waits also after termination of the load object handling, until the maximum loading capacity is achieved, if the button **fully charged** is selected. Basically the energy level can be maintained via this strategy, since a battery-charging possibility exists.

The second possibility is it to place a battery-charging station in the layout. If a vehicle has a lower energy reserve than the lowest limit inputted in the battery parameters after the delivery of its load-object to an unloading station, it is sent immediately to the next battery-charging station. For this, the switch **Battery-charging station** in the parameter of the modules is to be activated. This is possible with all processing stations of the transport system, particularly meaningful however only with a workstation. This must be additionally provided with a **destination code** then.

The user comes to the mask with the battery parameters by activating the switch **With battery control** of the dialog *Transport strategies*. At the same time, this activates energy monitoring with the required parameters as listed in Figure 5.22.

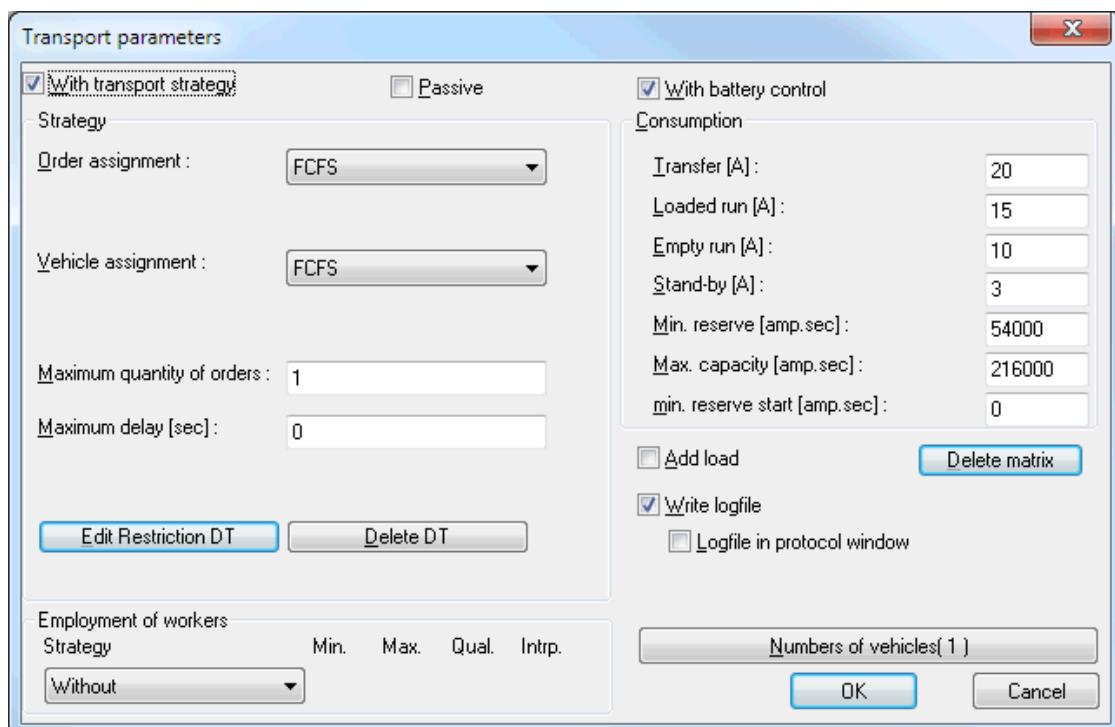


Figure 5.22: Battery parameters

The user has to specify the energy consumption values of a vehicle for *Stand-by*, *Transfer*, *Loaded run*, and *Empty run*. The simulation utilizes these data to control the energy statuses of the vehicles. In principle, the period, in which the vehicle is in the appropriate condition, corresponds to the period, in which this power is used. However, the following situations are to be considered.

- During the disturbing and blocking times, only the stand-by power is used



- The transfer power is spent only during the time indicated in the (Un)Loading stations. When waiting for entry into the successive module, the vehicle is again in the status stand-by
- Similarly during processing in a work station, only the stand-by power is used

Example:

Load/NoLoad/Transfer/Stand-By - 15A/10A/20A/3A

Tour: 20[s]Transfer, 120[s]Loaded run, 20[s]Transfer, 180[s]Unloaded run

$$\begin{aligned} \text{Consumption: } & 20[\text{s}] \times 20[\text{A}] + 120[\text{s}] \times 15[\text{A}] + 20[\text{s}] \times 20[\text{A}] + 180[\text{s}] \times 10[\text{A}] \\ & + (20[\text{s}] + 120[\text{s}] + 20[\text{s}] + 180[\text{s}]) \times 3[\text{A}] = 5420[\text{As}] = 1,5[\text{Ah}] \end{aligned}$$

The battery will be recharged once the level drops below the value set as **Min. reserve**. The station that charges needs a **full charge** instruction to continue up to **Max. capacity**.

5.3.8 Vehicle Bypass

The parameters of the modules unloading station and workstation possess a switch, with which the vehicles, which do not receive a new job, can be parked in a (fictitious) bypass. Thus, they are no longer at the station and block subsequent vehicles.

Vehicles, which are parked in a bypass, are further considered during disposition. They are treated the same as vehicles, which are at the appropriate module, without having been parked in the bypass.

5.3.9 Forcing of Double Cycles

Forcing of double cycles serves to avoid unnecessarily idle running of the vehicles and long waiting periods of the load objects on the loading stations.

In the simulation model, the loading and the unloading station create a docking station. At such a docking station, this kind of double cycle is to be forced. So that this case can occur, the following situation must be given: A loaded vehicle has the unloading station as its target and fulfills its task there. The object is unloaded. A load object is ready in the loading station of the same docking station simultaneously and triggers a pick-up-task there which is assigned to another vehicle. Forcing of the double cycle occurs at this point. The vehicle discharged just at this time receives the task to pick up the object in the loading station. The vehicle with the previous pick-up destination of the loading station receives a new destination. This is either the pick-up destination of a new task or - if no task is available - an idle run to a battery station.

5.4 Deadlock Control

In the simulation of transport systems, deadlocks occur primarily as the result of too many vehicles occupying a particular region at the same time. Local control avoids this type of situation by assigning a maximum total to the number of vehicles present in a certain module group. When more vehicles are registered in this group, signaling the risk of a deadlock, the entrance junction(s) to the area will be blocked, except for the assembly entrances of the assembly modules and the loading entrances of the loading stations. As vehicles depart the controlled region and the balance of vehicles drops below the limit specified, the blocked nodes will be released and vehicles put on stand-by can proceed again. Detailed information



on the sequences of blocking and successive releasing is provided in the chapter dedicated to capacity monitoring.

The application of capacity monitoring is not limited to closed circuits, but the modules effective in the restriction of vehicle flow can also be selected arbitrarily from the model to suit the situation.

5.5 Result Graphics

5.5.1 Vehicle Statistic

In the transport statistics, the status proportions of the individual vehicles, for example, are represented. Since in this case the blocking time additionally results (the vehicle in loaded run is blocked), this proportion is additionally entered as narrow beams. This value is however not displayed for display reasons in the picture below. They are to be taken from the statistics file.

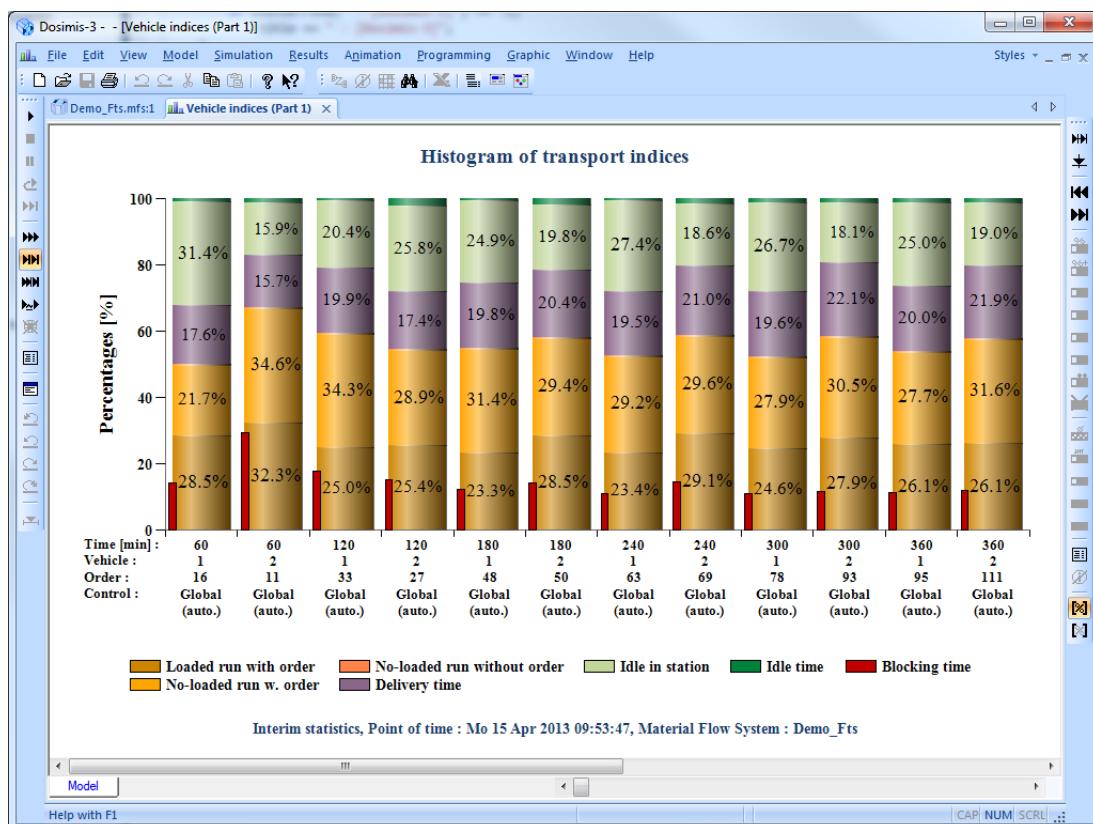


Figure 5.23: Histogram of –vehicle indices



5.5.2 Order Statistic

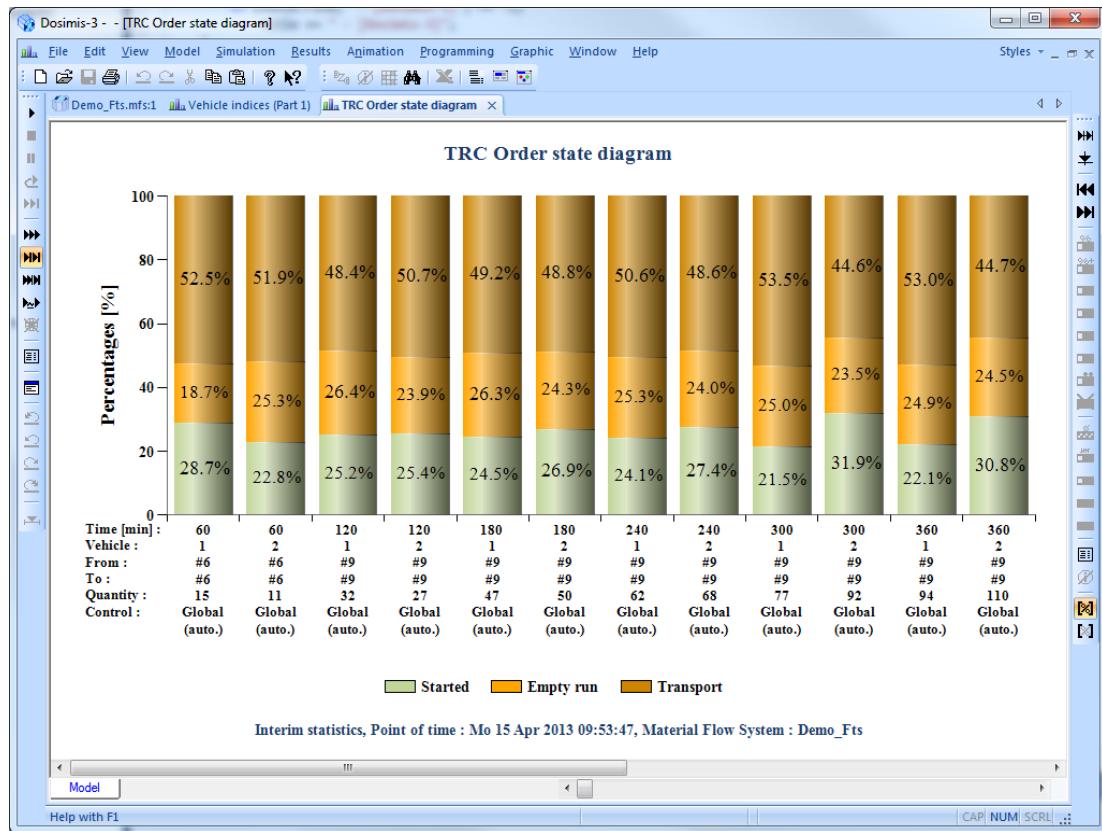


Figure 5.24: Histogram of state of order in a transport system



5.5.3 Order Throughput Time

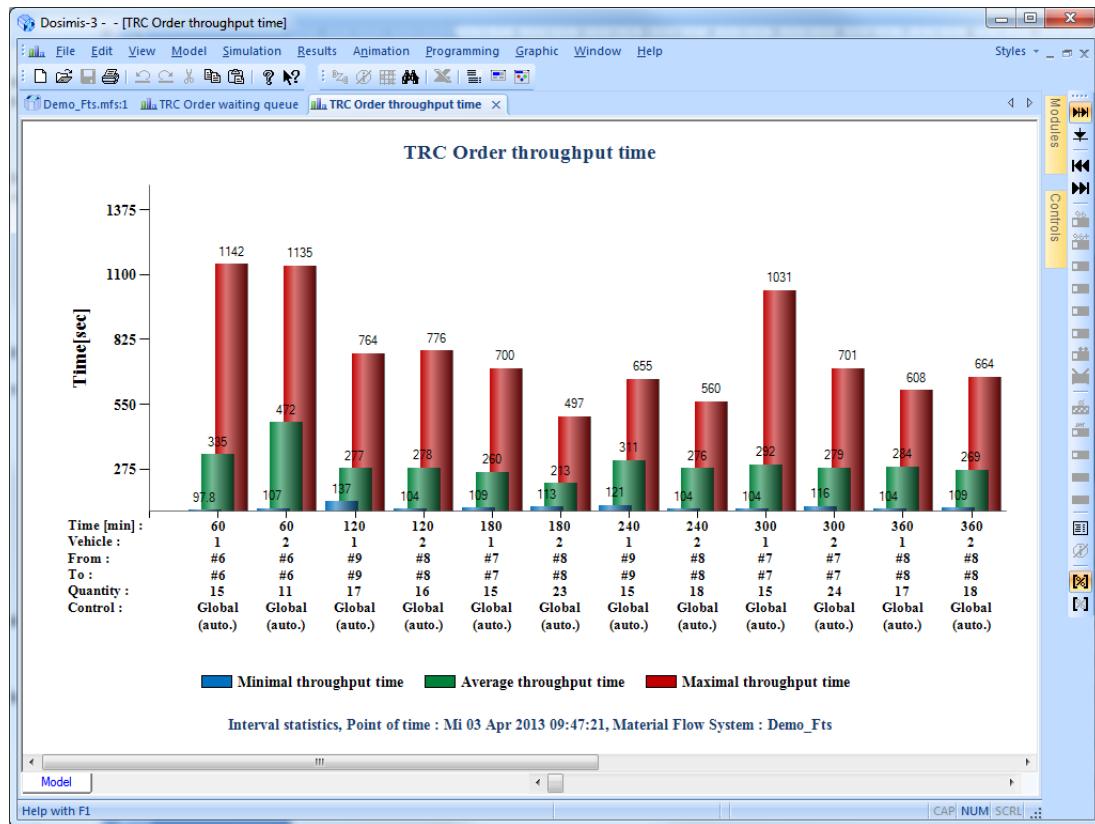


Figure 5.25: Histogram of throughput time of order in a transport system



5.5.4 Order Waiting Queue

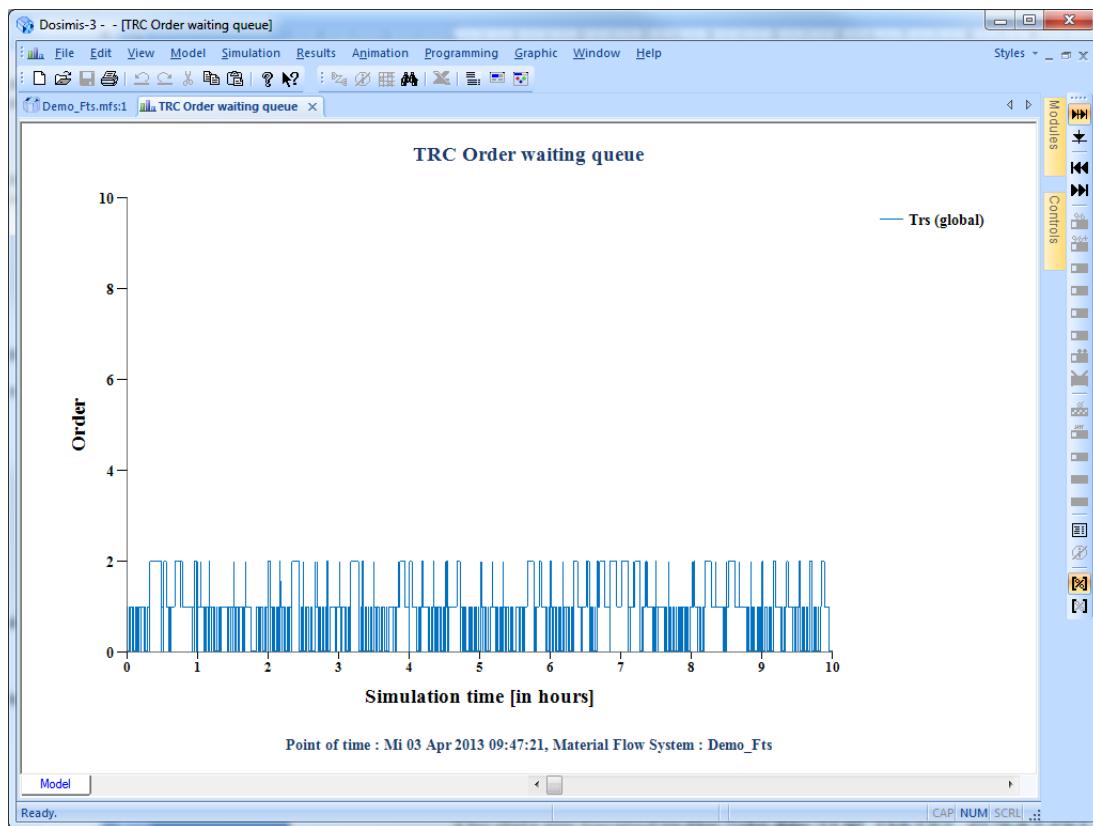


Figure 5.26: Histogram of waiting queue of order in a transport systems

The data are located in the *.slg-file below the modules statistics.

5.6 Result Table

Chapters *Simulation Run* and *Simulation Results / Output* cover the DOSIMIS-3 standard functions and familiarize the user with the output and interpretation of the simulation results. The statistics generated by these results are stored in the *.slg file together with the vehicle data. For each recorded statistics period (pre-, interval, interim and total statistics) order statistics and vehicle statistics are differentiated. Figure 5.28 shows a typical extract from a transport statistic.

5.6.1 Order Statistic

According to the orders the following parameters have been taken in the statistics:

- Vehicle number
 - Number of orders and loads
 - Average, minimum and maximum transport time
 - Average, minimum and maximum order time
- One order can contain several loads

Time between loading and unloading event

Time between order activation and unloading event (like above, but additional empty run of vehicle to loading station)



```
*****
Interval statistics : Order statistics (data in minutes)
Material flow system : Demo_Fts
Statistics for time interval : 0.0 - 60.0
*****
Vehicle Order Load Avg. order Min. order Max. order
number Count Count duration duration duration
-----
1 10 10 5.97 1.63 13.40
2 9 9 4.00 1.86 10.62
3 7 7 8.45 1.58 22.74
*****
*****
```



```
*****
Interval statistics : Transportation duration (data in minutes)
Material flow system : Demo_Fts
Statistics for time interval : 0.0 - 60.0
*****
Vehicle Avg. trans- Min. trans- Max. trans-
number port dur. port dur. port dur.
-----
1 5.97 1.63 13.40
2 4.00 1.86 10.62
3 8.45 1.58 22.74
*****
```

Figure 5.27: Order statistics of a transport system

All time values are given in minutes. In the example given, transport system data for the interval beginning at 600 minutes is recorded. During this period, 3 vehicles are required from transport scheduling. For each vehicle it is logged how many jobs and charges were processed by one vehicle. In the order statistics, a difference is made between **transport time** and **order time**. There is an order as soon as a load is reported at the loading station or a vehicle must go to the transport system station for battery charging. The transport schedule then tries to assign the order to a vehicle. If, for a number of reasons, this is not possible, an order is placed in the line of orders pending (see chapter 0). After the order has been carried out, the vehicle reports back to scheduling as being free. **Order time** is the period between the order arising and the report being free, while **transport period** begins when the order is given to a vehicle and ends when the vehicle is reported as being free. The minimum, maximum and average values of each parameter are recorded in the protocol interval.

5.6.2 Vehicle Statistic

Vehicle data comprise of the following parameters:

- Empty run with order Travels without load to loading station.
- Loaded run with order Travels with load to unloading station.
- Empty run without order See Empty run with order and below.
- Loaded run without order See Empty run with order and below..
- Battery needed The vehicle had to go the battery-charging station, since the minimum capacity was reached.
- Battery scheduled The vehicle is dispatched to a battery-charging station, since no job was present. This can be also a workstation with destination code and without battery-charging parameters (waiting position).
- Waiting time Waiting without order
- Failure time Times when a module is failed and the vehicle is in that module.
- Delivery time Times of (un)loading times to according the



- Battery charge parameters of the (un)loading station.
 - Wait in station Times of explicit battery charging. Parallel times while working or (un)loading charge are not considered.
 - Blocking time Waiting times in loading stations, e.g. the vehicle was disposed by DT.
 - With order Waiting time in unloading station, until the charge has completely entered the following module. This is according to the parameters of speed and length or by failures in the successive modules.
 - Without order Times, in which the vehicle was obstructed. These arise as a result of reverse jam on the track or waiting periods before (un)loading stations or workstations.
- The vehicle was controlled by scheduling. This can take place both by means of the job and with subsequent destinations in the stations or battery-charging trips or empty runs after the discharge process.
- The vehicle was not controlled by scheduling. This status can be caused if, for example, the vehicle will be provided with a new destination directly after the unloading process via the decision table.

```
*****
Interval statistics : Vehicle indices (Part 1)
Material flow system : Demo_Fts
Statistics for time interval : 0.0 - 60.0
*****
Vehicle- Empty-run Loaded-run Empty-run Loaded-run Battery Battery
number with order with order no order no order needed scheduled
-----
1 13.50 36.37 0.00 0.00 0.00 0.00
2 53.79 24.02 0.00 0.00 0.00 0.00
3 10.03 30.69 0.00 0.00 0.00 0.00
-----
Min. Time 10.03 24.02 0.00 0.00 0.00 0.00
Max. Time 53.79 36.37 0.00 0.00 0.00 0.00
Avg. Time 25.77 30.36 0.00 0.00 0.00 0.00
*****
```



```
*****
Interval statistics : Vehicle indices (Part 2)
Material flow system : Demo_Fts
Statistics for time interval : 0.0 - 60.0
*****
Vehicle- Waiting Failure Delivery Battery Waiting Blocking
number time time time charge in station time
-----
1 21.47 0.00 12.97 0.00 15.68 23.06
2 3.90 0.00 10.69 0.00 7.60 56.81
3 18.75 0.00 9.17 0.00 31.36 21.56
-----
Min. Time 3.90 0.00 9.17 0.00 7.60 21.56
Max. Time 21.47 0.00 12.97 0.00 31.36 56.81
Avg. Time 14.71 0.00 10.94 0.00 18.21 33.81
*****
```

Figure 5.28: Excerpt from the statistics protocol



Vehicle statistics separate the behavior of each vehicle within a statistic interval. Under **Empty run with order**, the time needed for a vehicle to reach the battery-charging station after receiving an order is recorded. It carries out a **loaded run with order** as soon as it leaves the charging station. Loaded runs end with an entry into an unloading station. **Loaded run without order** or **Empty run without order** can only occur by skipping the dispatching.

Furthermore, a difference is made between two different runs to the battery-charging station; runs which are necessary and scheduled empty runs. In the former, order classification for the vehicle is **Battery needed**, in the latter, **Battery scheduled**.

Waiting **time** is the time during which the vehicle is stopped. Order classification is of no importance here.

The statistics also record **Failure time**, which arises when a vehicle is standing in a disturbed module. The vehicle has, in this case, a disturbed state.

Delivery **times** occur during a load change; and these are times needed to pick up a load object in a loading station and deliver it to an unloading station.

Battery **charge** is the time required for charging. The vehicle is in the battery-charging station.

The parameter **Wait in station** is the time spent by the vehicle in a loading or unloading station waiting to pick up or deliver a load.

Blocking **time** is the time that a vehicle is in the status blocked. Registration of the original condition is not interrupted, i.e. the blocking time is to be seen as a part of the total time.

For these values, the minimum, maximum and average times for all vehicles are listed separately.

5.7 Consistency Check at Transport System

The purpose of the consistency check (see chapter Simulation Run) is to control the parameterization of a system for completeness. If the consistency check is positive, the system is clear of failures, indicating complete parameters, otherwise there will be an error message.

This section only explicitly states those potential mistakes that are due to false or incomplete parameterization. Causes of mistakes can be:

The user fails to enter a new type into the parameter dialog that corresponds to the module unloading station and fails to select a transport strategy. Either a new type or a strategy must be specified.

The user has activated the option with battery control, but the model contains no battery-charging station.



5.8 Restrictions

An adverse situation can be encountered in a transport system operating under the **FCFS strategy** when different vehicles are initialized in succeeding tracks:

In two adjacent tracks, two different vehicles are initialized. Due to the strategy effective at a particular time, the vehicle on the front track may be passed over for an order in favor of the vehicle on the track directly behind. The result will be that the selected vehicle is blocked by the vehicle directly ahead.

This deadlock can be avoided in the following manner:

The vehicles are no longer initialized separately in several directly succeeding tracks. Instead, they are generated in a single track. In the type of situation described above, this arrangement allows the vehicle that is ordered to pass the one on standby without being blocked by it.

5.9 Files in the Transport System

In addition to the files that are created in the course of processing any material flow system, a model with a transport system generates these files:

- *.*mtx* The file ***.mtx** is an internal program file. It contains a so-called track matrix which allows the shortest route out of all possible routings between the present location of a vehicle and its scheduled destination to be selected.
- *.*aws* The file ***.aws** stores the number of orders counted but not yet processed in a list. The line of orders pending denotes the currently listed status of unprocessed orders. The list is made up of two columns. The first one records the exact times in seconds of order registration/order completion. The second one records the number of orders not yet processed. The number is increased by 1 when a new order comes in and accordingly decreased by 1 once an order is reported as completed. A task remains in this queue until it has been completely carried out. However, this is not equal to the time where a vehicle is assigned to this task but the task is removed only from the list if the vehicle achieved its destination.
- *.*log* The file ***.log** records all scheduling decisions that pertain to the transport system. The entire range of possible decisions will be demonstrated with an excerpt from a log-file.



Example *log-file

```

Transport-disposition version: 1
Parameters of transport-disposition:
Order capacity of carrier: 1
Energy monitoring: FALSE
VEH 1 start with charge 0.00
  0.00 VEH 1 Dest: 1 PLACE 36 BLK_36 A 0 Dest -1 (veh_set_destination)
  0.00 VEH 1 Dest: -1 PLACE 36 BLK_36 A 0 with type -1 initialised
VEH 2 start with charge 0.00
  0.00 VEH 2 Dest: 1 PLACE 36 BLK_36 A 0 Dest -1 (veh_set_destination)
  0.00 VEH 2 Dest: -1 PLACE 36 BLK_36 A 0 with type -1 initialised
VEH 3 start with charge 0.00
  0.00 VEH 3 Dest: 1 PLACE 36 BLK_36 A 0 Dest -1 (veh_set_destination)
  0.00 VEH 3 Dest: -1 PLACE 36 BLK_36 A 0 with type -1 initialised
  0.00 VEH 1 Dest: -1 PLACE 36 BLK_36 A 0 *** veh_state: stopped
  0.00 EL 60 *** veh_state: loadingstation_objectready
Order 1 Pd 3 Sd 31 Class loadstation
  0.00 ^ order_activate
  0.00 Order 1 (Transport-disposition)
  0.00 VEH 1 Dest: -1 PLACE 36 BLK_36 A 1 Order 1 (veh_order_assign)
  0.00 VEH 1 Dest: -1 PLACE 36 BLK_36 A 1 Order 1 Dest 3 (veh_set_destination by dispo)
  7.50 VEH 2 Dest: -1 PLACE 36 BLK_36 A 0 *** veh_state: stopped
18.67 EL 38 *** veh_state: loadingstation_objectready
Order 2 Pd 32 Sd 11 Class loadstation
18.67 ^ order_activate
18.67 Order 2 (Transport-disposition)
18.67 VEH 2 Dest: -1 PLACE 36 BLK_36 A 1 Order 2 (veh_order_assign)
18.67 VEH 2 Dest: -1 PLACE 36 BLK_36 A 1 Order 2 Dest 32 (veh_set_destination by dispo)
21.00 VEH 1 Dest: 3 PLACE 60 BEL_60 A 1 Order 1 *** veh_state: stopped
21.00 VEH 1 Dest: 3 PLACE 60 BEL_60 A 1 Order 1 *** veh_state: waiting_in_loading
26.00 VEH 1 Dest: 31 PLACE 60 BEL_60 A 1 Order 1 *** veh_state: end_loading
26.17 VEH 3 Dest: -1 PLACE 36 BLK_36 A 0 *** veh_state: stopped
67.00 VEH 1 Dest: 31 PLACE 37 ENT_37 A 1 Order 1 *** veh_state: stopped
75.00 EL 60 *** veh_state: loadingstation_objectready
Order 3 Pd 3 Sd 11 Class loadstation

```

*Figure 5.29: Extract from *.log file*

The first feature is the uniform structure of this file, organized according to entries. Entries, that specify the same decision, are located on the same line unless they are excessively long, in which case they are continued in the next line.

Example: 26.00 **VEH 1 Dest: 31 PLACE 60 BEL_60 A 1 Order 1 En[As] 116473.50**

An entry begins with the time at which a scheduling decision was taken. The ID of the vehicle (VEH) and the destination code follow the point in time, to which module the vehicle is on the way. Under PLACE, the number and name of the module is found, where the vehicle currently stands. These data follow the number of assigned orders and the number of the first order of the vehicle. The last value indicates the current battery charge in Ampere-seconds, if battery monitoring has been activated.

The following decisions are possible:

- **initialized:** The vehicle will be initialized at the start of the simulation, in the above case vehicle 1 is initialized as type 1. More vehicles are initialized simultaneously, but deleted in the example for the sake of simplicity. Vehicle designations are always abbreviated with **Transport system** and are displayed in the second column. The type is indicated by **type-number**.
- **veh_order_assign:** An order from the list of orders pending will be assigned to a vehicle on standby. Orders are identified with **order number** and they are listed in the third column.



- **veh_set_destination by dispo:** A pick-up or a drop-off destination will be assigned to the vehicle. The destination is a module, but the identification number **destination number** is not identical with the module number.
- **end_loading:** This message reports a vehicle status. The vehicle has been loaded at the pick-up destination and is now waiting to be dispatched to a new drop-off destination.
- **end_unloading:** This message is displayed once the unloading process is finished.
- **order_ready:** An order is reported finished and the vehicle stands ready for the next assignment.
- **emptyrun_disposition:** Transport system operating under battery control will be charged up to a level that equals the vehicle capacity. As soon as the charging status drops below a certain value the battery needs to be recharged. In that case the vehicle will be sent without a load to a battery charging station.
- **start_charging:** This is another vehicle status and it reports the beginning of the recharging process.
- **end_charging:** Once this message is displayed the vehicle is ready for the next order.

Other entries in the log file are error messages and warnings. A warning does, in contrast to an error message, not cause the simulation to be terminated.



6 Petri Nets

6.1 Introduction to Petri Nets Theory

The following introduction is to give the user some brief information about the Petri nets system. It was developed "to describe as many phenomena as possible during transfer and conversion of information in an exact and uniform way", to quote C.A. Petri in '*Kommunikation mit Automaten-Schriften des Institutes für instrumentelle Mathematik*', Bonn, 1962. To do this, different classes of nets were developed and investigated. This introduction describes only two of the essential network classes. For those users who wish to work on this subject in more detail, there is a wealth of advanced literature available.

6.2 Condition/Event Nets

The structure of the Petri network is simple and easy to follow. It has the following modules:

- State node
- Event node
- Directed edges

A **state** (also called **condition**, position or place) is the momentary state of a process. A process in the language of the Petri network theory is known as state/event links or event/state links. The state node is shown graphically as a circle. Events (also called transitions or actions) cause a change in the state. This type of node is shown as a rectangle. A directed edge, shown as an arrow, always links two different types of nodes i.e. an event node with a state node (and vice versa).

Furthermore, when modeling, the following restrictions must be observed:

There are to be no isolated nodes, i.e. there must always be at least two connected nodes.
A state node in a model cannot simultaneously be an event node (or vice versa).

Within this condition/event network (C/E network), alternative and parallel runs are possible: several edges can go out from one node (with several successor modules) and lead to it (with several predecessor modules). A formal difference is made between taking and giving edges. Taking edges connect a state node (condition) with an event node (event). The event node is here the successor module. Edges which connect event nodes with state nodes are called giving edges. In this case, the state node is the successor module of the event node. NB: Taking and giving edges are one and the same module - directed edges!

As already mentioned, events effect changes in states, which means each state is cancelled or initialized by at least one event, known as realizing of states. Realizing of states means that switching of succeeding events (can take place) is possible. This is done by means of a so-called 'marking'. Each state realized is given a mark. This is why a C/E net is also known as a 'one-mark Petri net'. A mark is indivisible and of only one single type. An event can only take place if the preceding module (state node) is marked, i.e. has been given a mark. If the event node has several preceding modules they must all, without exception, be marked, i.e. all pre-conditions must be true. In addition, a/all successor state/s must be free of marks, i.e. the post-conditions must be untrue. If an event is now activated, a mark is removed via the taking



edge(s) from the successor state and given to the successor state via the giving edge. The state in such a successor module is now taken to be realized, i.e. it is true. This redistribution of marks is known as a switching rule where the switching processes are assumed to be timeless, that means without time delays.

If a marked state node has several event nodes which can all be switched, this results in conflict. As a mark is indivisible but several events can be selected, a random selection is made to activate those events possible.

6.3 Positions/Transitions Networks

This class of net is an extension of the C/E net. It was introduced to make the nets more compact as C/E nets easily become too large and non-transparent.

A P/T net is a 'multi-mark Petri net'. A state (also called position) can have several marks, unlike the C/E net. Another difference is the switch process, i.e. when an event (also called transition) is activated when several marks can be transferred via the giving and the taking edges. For this purpose, the edges are given a so-called edge importance which fixes the number of marks which can be given via this edge. This influences the switch rule greatly: a transition can be activated in a P/T network if the number of marks in the preceding module(s) is at least as high as the edge weight of the taking edge(s). The occupancy of the successor module is not taken into account with this switch rule.

6.4 Petri Nets in DOSIMIS-3

As previously mentioned, Petri nets have various modules: directed edges, event nodes and state nodes. They are to be found in DOSIMIS-3 under the following:

Directed edges This module is shown by a simple DOSIMIS-3 link node.

Event node This module is symbolized with the module **PN_EVENTHIDD PN_EREIGNIS**, using a rectangle.

State node This is shown as the module **PN_STATEHIDD PN_ZUSTAND**, where the circle - the graphic symbol usually used in the Petri net was replaced by a rectangle whose length can be determined at will when modeling. This deviation was made so that during animation it is instantly recognizable how many marks or objects the module contains.

The following table shows the connections:

General names	General symbol	DOSIMIS-3 names	DOSIMIS-3 symbol
State (position)	○	PN_STATE	---
Event (transition)	□	PN_EVENT	□
Directed edges	→	Usual DOSIMIS-3 junction	>

Figure 6.1: General Petri-Net symbols and DOSIMIS-3 Petri-Net symbols



The Petri net in DOSIMIS-3 differs from the net classes of the Petri net theory; rather it is a combination of a C/E net and a P/T net. In DOSIMIS-3, the state nodes can have several marks (as in P/T networks). The switch rule is then as follows: to activate an event all pre-conditions must be true, i.e. all preceding modules must have been allocated at least one mark. The marking allocation for the successor modules is irrelevant. In a switch process, a mark is removed from the preceding module and given to a successor module via the giving edge. Contrary to general theory, the selection is not random for branching points with several switchable events. In DOSIMIS-3, an event is determined by the sequence of the exits. This sequence is determined during modeling by the sequence in which the single DOSIMIS-3 Petri net modules are connected to one another. A priority is formed as the exit with the lower number is always given preference. Figure 6.2 is given to illustrate this:

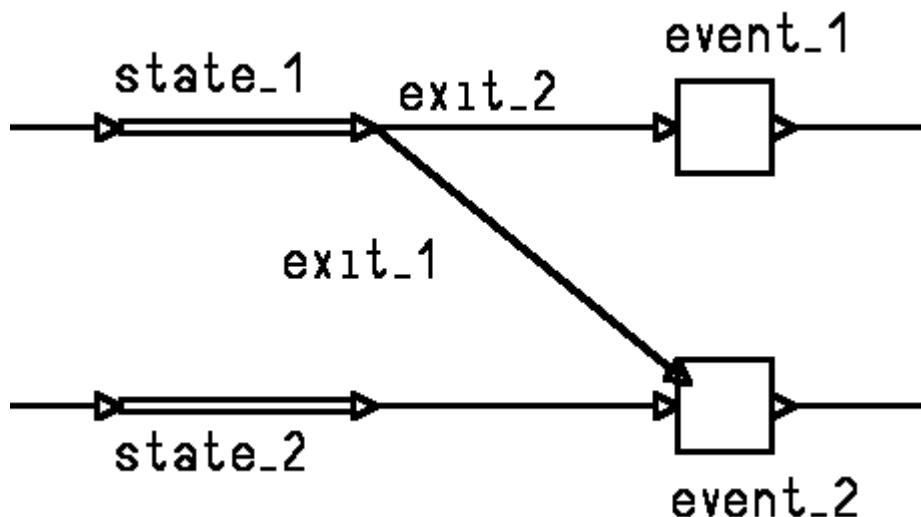


Figure 6.2: : Example of a Petri-Net

In the figure are (in DOSIMIS-3 symbols) each two state nodes (state 1 and state 2) and two event nodes (event 1 and event 2) (see below for an explanation of the modules). The modules are connected by the link nodes. The global connection of the process is irrelevant in this example as only the branching on state 1 is of interest. In the model, state 1 is first linked to event 2 (exit 1), then with event 1 (exit 2). State 2 is then linked to event 1.

The following cases are now conceivable:

- State 1 has no mark \Rightarrow no event can take place in any event node (even if state 2 has a mark).
- State 1 has at least one mark, state 2 has no mark \Rightarrow As **all** entry point modules must be marked in order to activate an event , the event cannot take place in event 2. Therefore exit 2 from state 1 is taken as a branching.
- State 1 and state 2 both have at least one mark \Rightarrow both events can take place. As branching, exit 1 from state 1 is selected, i.e. event 2 takes place.

When using Petri net modules in a DOSIMIS-3 model the following is to be observed:

The modules cannot be built into just any model. Only **exit 1** of the PN-event can be connected to the other DOSIMIS-3 modules. If the Petri net area is to be left, then only via



exit 1 of the PN event module. All other entry points and exit points can only connect Petri net modules with one another (taking into account the restrictions that only states and events can be connected, and vice versa).

An object entering a Petri net is converted to a mark. The object type is stored until it leaves the net, i.e. a mark is then converted back into an object of the same type.

Changes in marking situations, meaning adding or removing a mark, are defined in the Petri net theory as being timeless. Contrary to this, in DOSIMIS-3, these switch rules do not have to be regarded as timeless, but can be given a time duration by giving the PN-event a residence time.

6.5 Petri Net State

The module **PN-STATE** in DOSIMIS-3 corresponds to the state node. In the model, the length and progression in the simulation model must be first defined, followed by the number of entrances and exits. The following menu results are obtained for setting parameters:

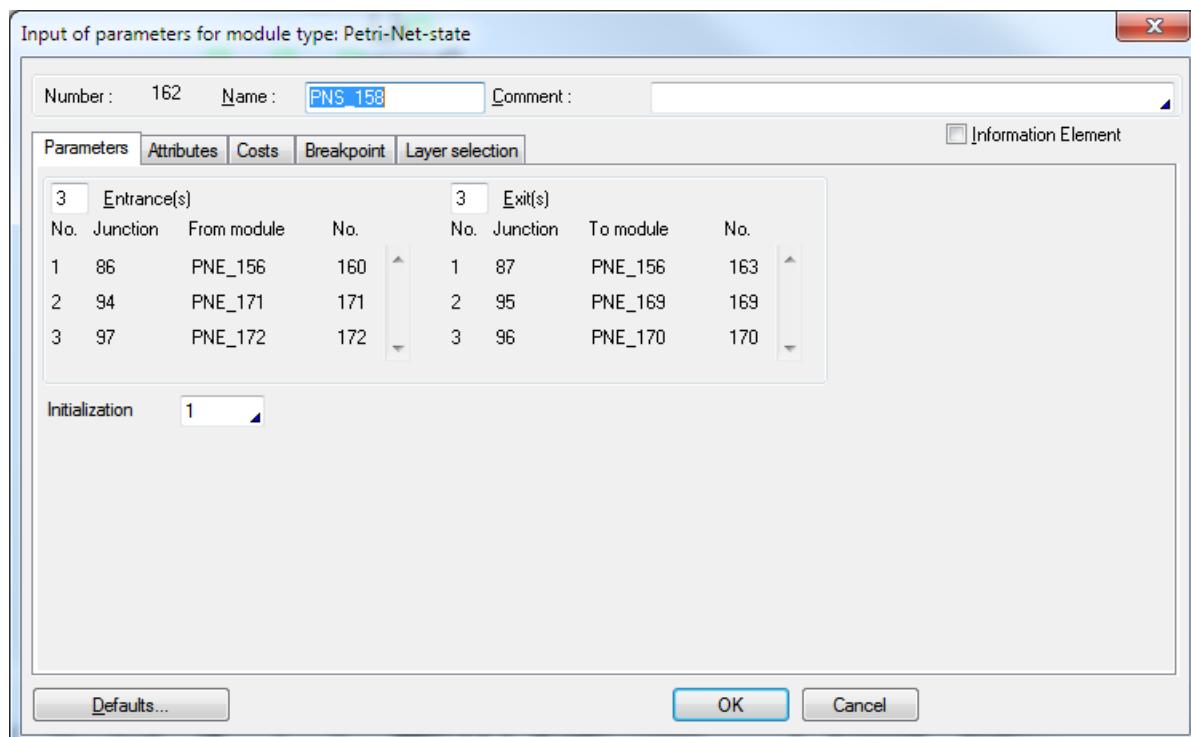


Figure 6.3: Parameters of a Petri-Net-state

A value for the initial occupancy can be given in addition to the standard parameters module number and module name. This value gives the number of marks which the state node has at the start of the simulation (zero = no marks). It is also possible to generate the value of the initial allocation in a decision table.

Additionally, the number of entrances and exits are given (this value can be changed later). Depending on this number, this results in a list of all entrances and exits with their relevant node numbers and modules



6.6 Petri Net Event

The module **PN-EVENT** corresponds to the event node. It has any number of entrances and exits which are to be given when modeling but which can be altered afterwards. The following menu appears when setting the parameters:

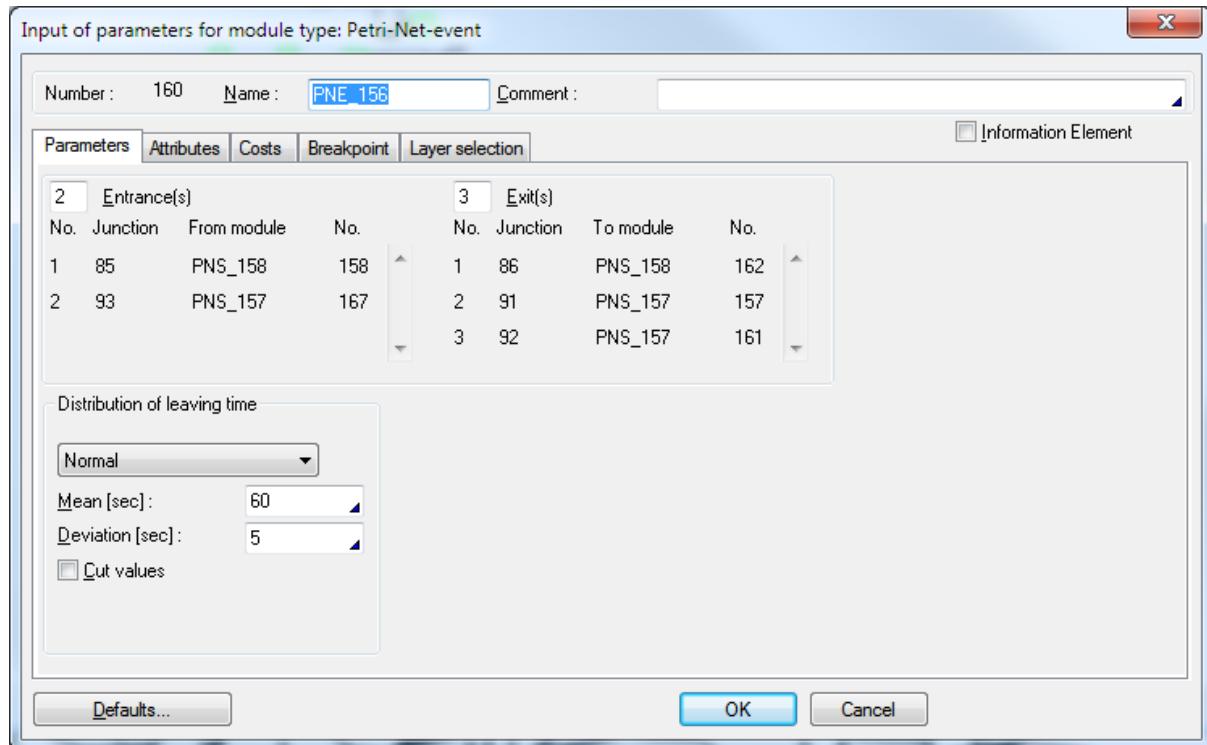


Figure 6.4: Parameters of a Petri-Net-event

At first, a module number and name can be given. The length of the residence time can also be given.



6.7 Examples

In conclusion, the function method of the Petri net modules and their effect with other DOSIMIS-3 modules will be shown in two examples:

6.7.1 Example 1

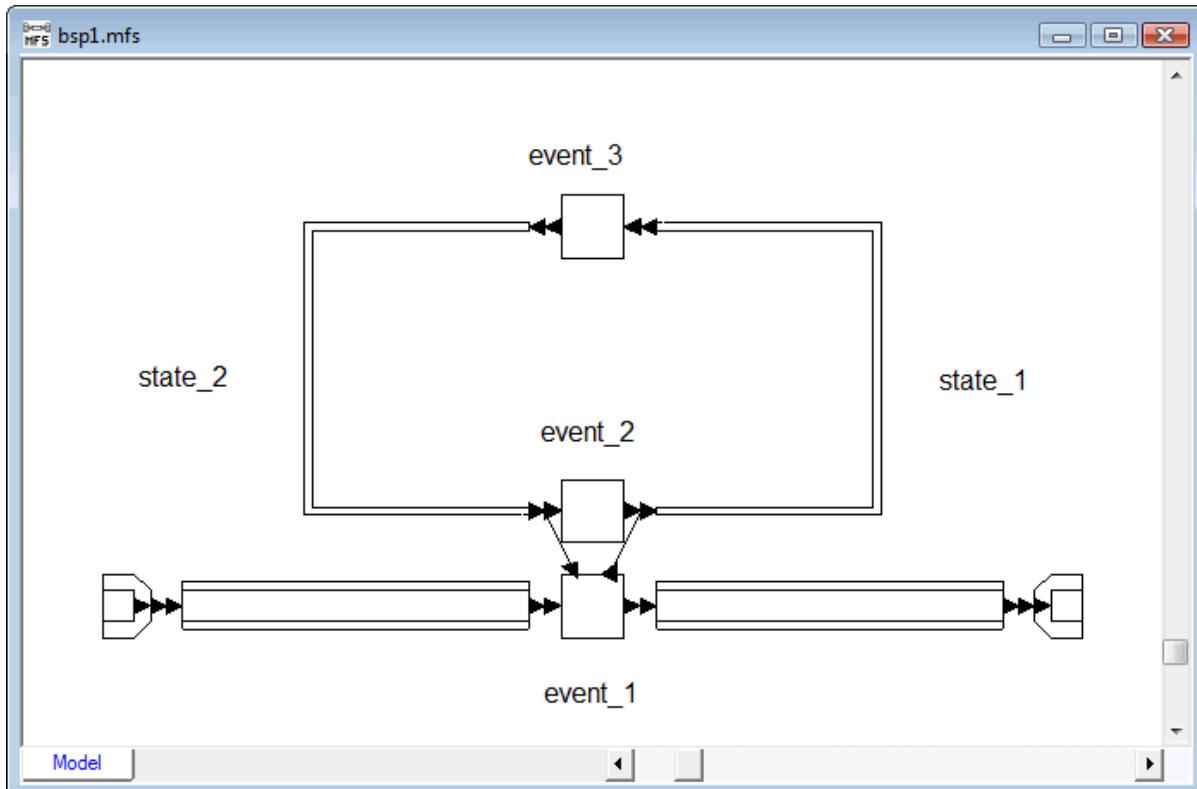


Figure 6.5: Example 1

This model shows the generation, transport and pick-up of objects. The focus is on the transport and here especially the area of PN event 1. Objects are only to pass through this module at certain times. This is achieved using the Petri net modules:

At the source, type 1 objects are produced at 30-second intervals. The produced objects enter the buffer section and wait at the end to enter PN event 1. However, the objects can only travel through this when it is activated, i.e. when all its entry modules, without exception, have been marked. PN event 1 has each two entrances and two exits. Entrance 1 is connected with the buffer section where the objects are waiting, entrance 2 with PN state 2, exit 1 with the second buffer section and exit 2 with PN state 1. PN state 1 is initialized with a mark, PN state 2 is not initialized. The markings play a decisive role in the Petri nets. The mark in PN state 1 causes PN event 3 to be activated which leads to an event. Thereby, one mark of the entry module (PN state 1) is removed. When setting the parameters PN event 3, a residence time of 30 seconds was given and after activating this module waiting for this residence time to pass occurs as an event. After the end of the event (after the time has passed), the successor module PN state 2 is allocated a mark. If an object is now waiting at the front of the buffer section, PN event 2 can be activated, the object can pass the node, reaches the second buffer section and from there, the sink. The time which the object needs to pass through the PN



event 1 is again given as a residence time (in this case, an extremely short time: 0.001 seconds). Finally, after the beginning of this event, one mark is removed from entry module PN state 2 and the exit module PN state 1 is allocated a mark. This mark can then in turn activate PN event 3.

Objects can only pass through PN event 1 if PN state 2 has been allocated at least one mark. If there is no object waiting at the front of the buffer section, PN event 2 is activated. In the same way, this results in a mark being removed from entry module PN state 1 after a certain given residence time (in this case, 0.0 seconds) and exit module PN state 1 is allocated a new mark.

If an object is waiting in the buffer section, PN event 2 cannot be activated. In DOSIMIS-3, it is prevented by a series of exits. PN event 1 is determined as exit for PN state 2; exit 2 is connected with PN event 2. If PN state 2 has a mark and an object is waiting in the buffer section (this is also to be understood as a mark), PN event 1 is automatically activated. If there is no object waiting in the buffer section, then not all entry modules of PN event 1 have a mark and PN event 1 cannot be activated. Instead, PN event 2 is activated.

6.7.2 Example 2

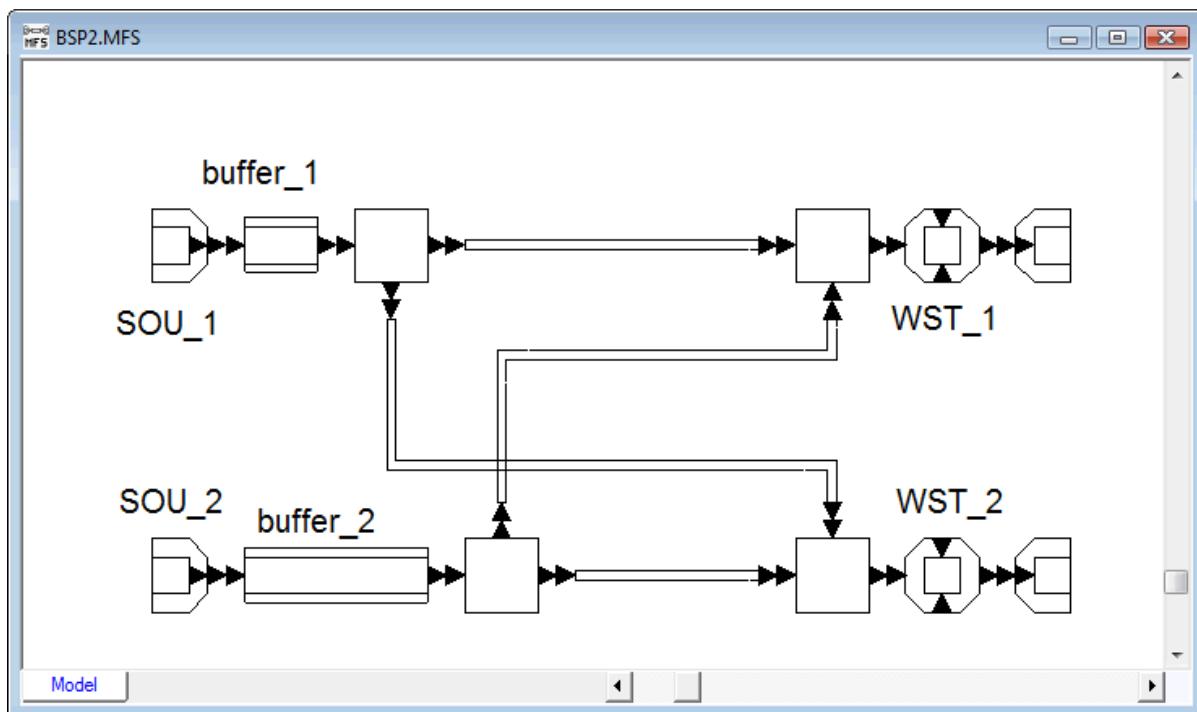


Figure 6.6: Example 2

The two sources 1 and 2 produce exponentially distributed objects 1 and 2 (mean value 60.0 seconds), causing strong fluctuations in object generation. The purpose of this model is to synchronize the entry into both work stations. (WST 1 and WST 2).

Both buffers are each connected with entry 1 of the relevant module PN event. Each of these modules has two modules PN state as successor modules. These modules are initially allocated with zero, i.e. they have no marks. A further pair of modules of type PN event joins them, each being connected via exit 1 with the final work station.



There now follows a description of the route of an object which has been generated in one of the two sources. The object leaves the source, e.g. source 1. It reaches buffer 1 and passes through up to the foremost segment of the buffer. Thereby, a PN event is activated and can switch. The entry of the object into the module and its immediate exit (residence time 0.001 seconds) represent the event. This switch process leads to the two successor modules each being allocated a mark. At this point, the object has been transformed into a mark. The event in the following module PN event cannot be switched. It can only be activated when an object is generated in the other source and passed through the buffer. Thereby it activates the other module PN event and the event can switch, thus allocating the successor modules each with a mark. Only now can both events in the second pair of modules Type PN event be activated. A mark is removed from each of the preceding modules and both work stations begin their work simultaneously.



7 Work Area

7.1 Introduction

A task begins when a worker goes to the specific module and ends when the time necessary for the work to be done according to the parameterization has passed.

At work area level, the modeler can define all the tasks in the real system carried out by workers. These include, for example, the manual recovery of failures, manual processing of objects in work stations, maintenance work, preparatory work and suchlike. Problems concerning the employment of personnel in material flow systems can thus be simulated. A so-called work area is a limited partial area of the total system, in which a definite number of workers, with varying degrees of qualifications, can be grouped.

7.1.1 Worker Request

In modules, where manual tasks can be processed, the requirements of workers for the respective task can be fixed in **Employment of workers**.

Distribution of working time				Employment of workers					
Objects	Distribution	Mean[sec]	Deviation[sec]	Strategy	Min.	Max.	Qual.	Intrp.	
1	Normally distributed	work	5	Minimal no.	1	3	2	10	
10	Normally distributed	\$Working	5	Maximal no.	2	2	1	12	
2	Normally distributed	=3600/144	5	Without					
*	Normally distributed	80	5	Maximal no.	2	3	1	5	

Figure 7.1: Parameters of manual tasks

If under **Strategy** a worker employment is selected, then the *Reference operator quantity* is specified at the same time, i.e. up to how much persons the work time refers. Furthermore it can be indicated, how many workers for **minimum** and **maximum** are used for this work and which **qualification** they have to have. If in the field **Interrupting (Intrp.)** a positive value is indicated, then the user specifies, starting from which task priority this work may be interrupted. The value 1 has the highest priority.

7.1.2 Assignment of Workers

During the work on tasks, two situations must be distinguished.

One worker per task:

Work is basically allocated as follows:

If a worker has finished one task, a check is made to see if there is a new one for him. If not, he either goes to his regular work place, or, if this has not been defined, he remains at the site of his last task. On the other hand, if there is a new task for to be done, a check will be made to see if there is a worker available to do it. If there is not, then the task remains characterized as 'not carried out' until a worker has been assigned to do it.

Several workers per task:

Fixed number of workers: The task is only carried out when the necessary number of



workers, i.e. a minimum number, is available to do the job. If necessary they will be released from other tasks, but the given maximum number of workers is not exceeded. If there are an insufficient number of workers, other idle workers can take on other tasks and the task cannot be started. In the case that one task is interrupted in order to begin another task for which only a few workers are required, and then the remainder of the workers can be elsewhere employed. It is therefore possible to deploy some of the workers for other tasks and the remainder continues the job already begun.

If no new task occurs, the number of workers can be increased after the beginning of a task if, in the meantime, further workers have become available and the maximum number of workers has not yet been reached.

7.1.3 Duration of Tasks

The duration of the task is the work time to be performed which is determined from **Given time*** **Max. number of workers**. This is reduced by parallel processing proportional according to the number of active worker, when several workers do this job.

Example: The given time in a workstation is 15 minutes. 2-4 workers are permitted. The time should refer to the maximum number of workers, so the total working time is 60 minutes. In this example 4 workers need 15 (= 60/4) minutes. If fewer workers are available, the time increases accordingly: 3 workers need 20 (= 60/3) minutes and 2 workers need 30 (= 60/2) minutes.

If the task is begun with 4 workers and one worker is taken off after 6 minutes, the remaining time is calculated thus: 60 minutes worker time less by $6 \times 4 = 24$ min. equals a remaining worker time of 36 minutes. The remaining 3 workers must work for a further 12 minutes.

7.1.4 Creating a Work Area

If a new work area is to be created, a symbol  must be positioned on the screen symbolizing the so-called basic position of the work area. It can be imagined as a recreation room where the workers spend their break. The circles represent symbolically the possible positions of the workers, the rectangle represents the table.

By the mode *Linking is active*, modules, in which a manual task will be carried out, are inserted into the work area by selecting the relevant modules in the layout. Elements which have already been selected appear in red.

7.1.5 Creating a Working Place

Placing of the working place symbols is done via their selection from the modules pallet and positioning it in the layout. If the symbol should be valid for certain modules only, then the symbol is to be connected with the appropriate components. Similar applies, if a firm allocation is to be made for one or more work areas. Here, it is to be noted that, when connecting 2 controls with the selection of the work area, the SHIFT key must be pressed.

The connections are relevant only during the animation. If work areas or modules are connected with the place, only the workers, who are assigned to one of the work areas and/or work at one of the assigned modules, are shown there. If one of the two lists is empty, the appropriate restriction remains unconsidered with the animation and the display takes place independently of work area/work place of the worker.



7.1.6 Parameters of a Working Place

The parameters of the working place have a mostly informative character. When the place is connected with a work area or a module, the name of the first element of the list will be displayed. If further elements of the same type are connected with this place, the name is followed by dots.

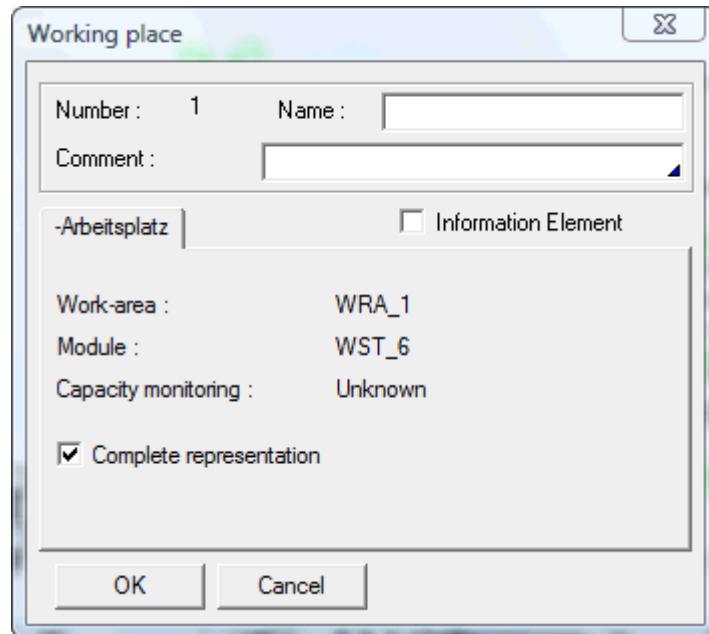


Figure 7.2: Parameters of a workplace

Additionally, it can be specified whether the working place is to be represented in a little more detailed way. For this, the switch **Complete representation** must be activated. Then the representation takes place not as circle but symbolically like a person with 2 arms.

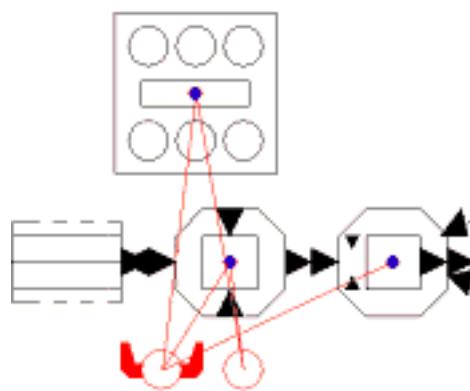


Figure 7.3: Different designs of a workplace

The processes within the work areas are animated. Further information can be found in the section [animation states](#).



7.2 Area Parameters

The real setting of parameters takes place in the parameter dialog of the work area. The mask can be filled up by clicking the module icon. In the parameter dialog is the name of the work area, which can be altered. The following illustration shows the parameter mask after clicking the control element of work areas.

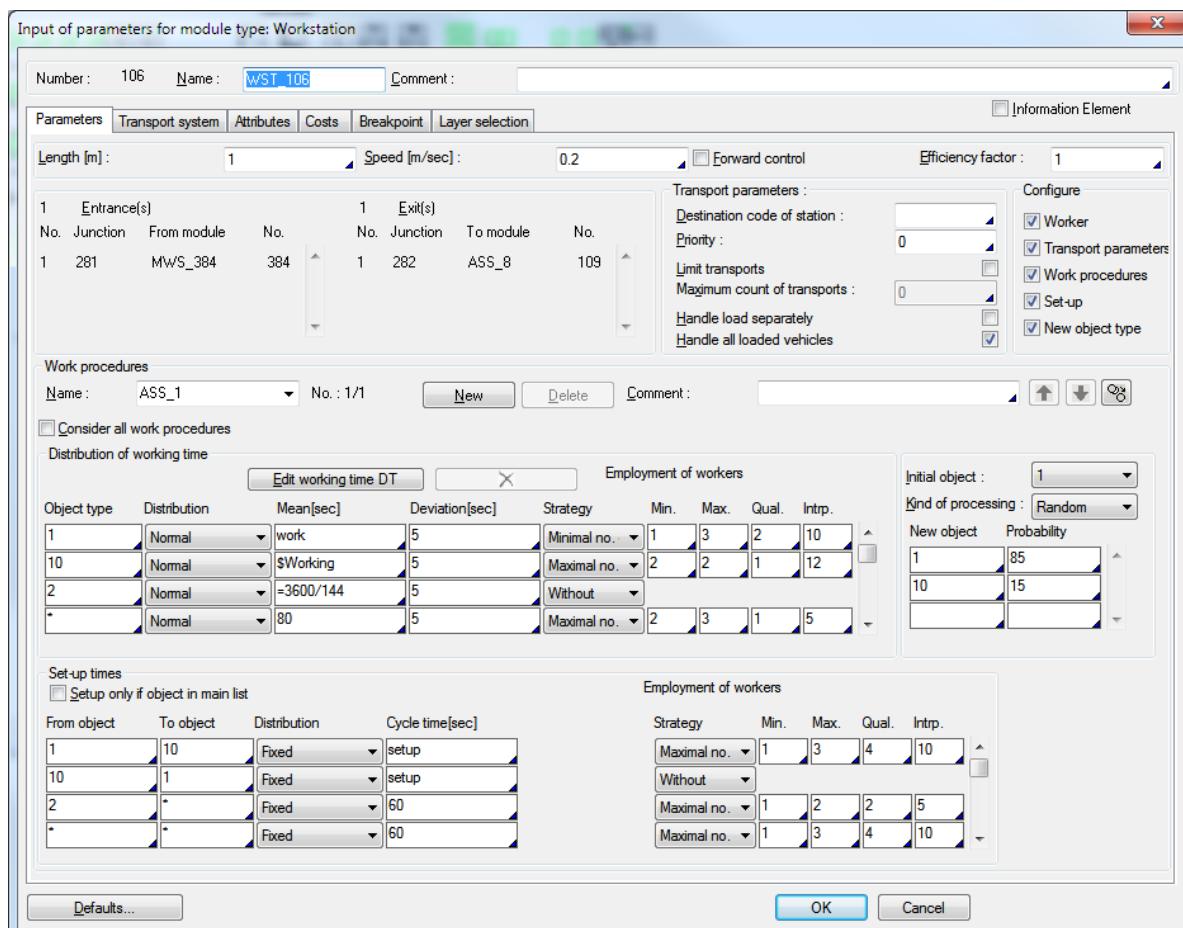


Figure 7.4: Parameters of the work area

By the switch **passive** the work area can be switched off. The workers defined in this area are not available for processing of orders. If there is no other work area, which may implement similar activities, the model is inconsistent. The investigation of a model without consideration of personnel can be activated for the entire simulation model in the simulation parameters.

If no modules are assigned to the work area up to now, a work area cannot be processed completely.

7.2.1 List of Workers

The parameters concerning the number of workers and their levels of qualification are to be entered as well as their master place and power factor.

The level of qualification is given as a number, where number **1** represents the highest level. Tasks are also assigned numbers and can only be performed by a worker whose qualification



level is higher than the qualification level of the task in question, i.e. when the number of his qualification level is lower than or the same as the qualification level of the task.

The master place is the place, to which the worker will go, if no further tasks results. Here 3 substantial alternatives exist.

- Last place: The worker stays at the place of the last work until a further activity results.
- Break area: After the task the worker goes back into the break area.
- Module: The worker goes to his assigned advantage module. Those can be any module of the work area.

In case of a work break the worker goes however to the break area.

It is also possible that a task can only be performed by a worker with the same qualification level. This is done by deactivating the option **Worker allowed to process task of lower qualification level**. This applies for all workers of the work area.

7.2.2 Scheduling Rules

By moving the criteria within the list with the help of the arrow keys the criteria can be brought into an order. The assignment from worker to the work is examined gradually. If after the evaluation of a criterion the allocation is clear, the arrangement is terminated.

By entering priority values, the sequence in which selection criteria are considered are predefined.

They are differentiated as follows:

Worker allocation: When selecting a worker for a task the following can be considered. The situation is: For a work several workers are available. The sequence, in which these factors are considered, is initialized according to standard in the following order:

- **Qualification:** If several workers are without activity and a further task appears, that the worker, which fits the qualification best (difference between worker qualification and task qualification), is selected.
- **Minimum path:** If two (or more) workers possess the same quality class, then that, which must put the shortest way back, is selected. This happens with the help of the way list (see above).
- **Maximum waiting time:** If now several workers with equivalent qualification are free and these had also still additionally equal long ways to the activity, the one with the longest waiting period is selected.
- **Minimum utilization:** If several workers are applicable for the activity, then the one with the minimal utilization is selected. Thus an even utilization of all workers can be promoted.
- **Maximum utilization:** If several workers are applicable for the activity, then the one with the maximum utilization is selected.
- **Priority at master place:** If several workers are applicable for the work, then the worker which master place agrees with the place of the work is selected.



Task allocation: When selecting a task for a worker, the following criteria can be considered. The situation is: For a worker several activities are available. The sequence, in which these factors are considered, can be likewise given by the user. There is a pre-setting also here:

- **Task priority:** If there are several tasks for an idle worker the one with the highest priority will be assigned to him.
- **Qualification level:** For two or more tasks with the same priority, the qualification level necessary is the second criteria when selecting. This is compared with the existing qualification levels of the available workers where the one with the highest possible qualification level is selected.
- **Shortest distance:** Analogue to the choice of worker, where there are two tasks to be performed with the same priority and the same qualification level the one where the worker has the shortest distance to go is selected using the route list.
- **Maximum waiting time:** Finally, if the distances to be covered are also the same, the maximum waiting time of a task is taken as the final criteria.
- **Interrupt task:** Tasks, which were interrupted because of the departure by workers (e.g. breaks or other activities with higher priority), are again taken up with highest priority.
- **Priority of master place:** If several works stand for worker to the selection, and then the work is preferred, that results at the master place of the worker.

Additionally it can be specified by a restriction decision table, which combinations of task and workers are inadmissible. By assignment to the variable *selfdef*, the user can exclude certain combinations. This is the case, if the variable supplies a value not equal to zero. Further information is to be inferred from the chapter on decision tables.

7.2.3 Task List

The list of tasks can be processed in the parameter dialog. In order to allocate tasks to certain workers, each task is given a priority number to ensure an unambiguous priority ruling. As already explained, the rule is: the lower the priority number, the higher the priority. The number of the activity is assigned automatically by Dosimis-3. Over this number the activity for the further parameterization can be identified.

In the next column, the user pre-defines the type of task by clicking the relevant line in the column **Kind** with the left mouse key. Several different types of tasks are listed there, from which the user must select one. These are

- 1) Object processing,
- 2) Set-ups,
- 3) Other activities,
- 4) Elimination of failures,
- 5) Maintenance,
- 6) Supervision.
- 7) Cleaning



For the simplification of the input for each module, which lies in the work area and where a manual work is foreseen, an entry for the object processing can be produced by the switch **Complete**.

When simultaneously the Shift or Ctrl-Key is pressed, further entries can be created automatically.

Shift	Ctrl	Task	Elements
No	No	Object processing	All modules of the work area with manual work
Yes	No	Cleaning & Setup	All modules of the work area, where a manual cleaning or setup process exists.
No	Yes	Failure repairs/Maintenance	All Failures, Maintenances and Pauses, where a manual repair is parameterized.
Yes	Yes	Supervising & Transport	All capacity monitoring and transport controls where a worker request is parameterized

7.2.3.1 Object Processing

DOSIMIS-3 has several modules in which manual tasks can be performed: work station, assembly and disassembly modules, loading and unloading stations. These tasks are found under the general heading of **object processing**, which means naturally that the module defined as work place must be the relevant workstations or the assembly/disassembly modules. If another module type is selected this is shown by the corresponding system message. In addition, object processing depends on the type of object so that a worker does not have to be permanently available at the above-named module. In the parameter dialog, additional worker deployment strategies can be given to regulate the task changes during object processing.



The user can select only one of the listed strategies for a change in work place. All of the three additional strategies can be selected if required.

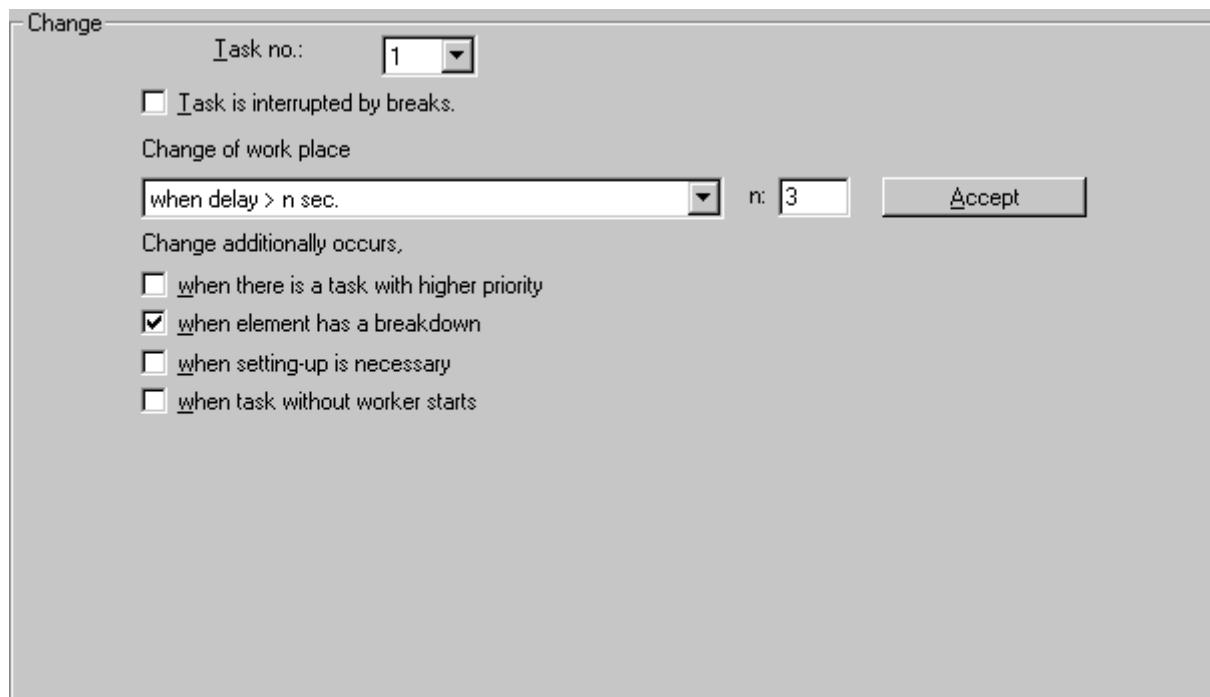


Figure 7.5: Worker deployment in object processing

Furthermore it must be stated whether or not each task may be interrupted by the worker taking a break. If so, the worker begins his break immediately and the interrupted task can be finished by another worker. Otherwise, the worker does not begin his break until the task has been concluded. The break is taken afterwards in its full length.

Object processing is performed thus:

Each object entering is checked to see if the type allows manual processing. If so, then a worker is called. If no worker is available at that moment, the task is marked 'unfinished' and the module is in a stand-by position, waiting for a worker to become available.

If a worker has been allocated to the object processing, he takes the route to the relevant module, which is still in stand-by state.

On arrival of the worker, the work time, which has been defined in the module and which, according to the described parameter on the parameterization level is necessary, begins.

When the task is concluded, the worker is immediately available for other tasks, if there is no more strategy for the change of the workplace is defined. The following options are possible:

- after each process
- after n processes
- when absolute occupancy of predecessor < n
- when relative occupancy of predecessor < n %
- when delay > n sec.
- when assembly is finished
- after process to next task
- to next task, absolute occupancy of predecessor. < n



- to next task, relative occupancy of predecessor. < n%
- to next task, when delay > n sec.
- after process to next working operation
- next working operation, absolute occupancy of predecessor.< n
- next working operation, relative occupancy of predecessor.< n%
- next working operation, when delay > n sec.

That means:

- *Occupancy of predecessor:* The occupancy of the module directly in front of the workstation.
- *To next task:* Go to that module where the task, which lies in the task list directly succeeding the current task, has to be worked.
- *Next working operation:* Go to that module where the task, which lies according to the work plan directly succeeding the current task, has to be worked

With the parameter *n* the strategy can be more specified.

During the strategic waiting period it can be determined with the help of the further options whether and under which conditions the waiting process is to be aborted:

- When module has a breakdown
The worker leaves his workstation, if the current module is disturbed.
- When setting up is necessary
The worker leaves his workstation, when a set-up process is necessary at the current module.
- When there is a task with higher priority
The worker leaves his workstation, when an activity exists, which is more urgent.
- A task without worker starts
The worker leaves his workstation, if at the current module an object handling comes up with no personnel needed.

The configuration for changing for all homogeneous activities can be produced by the switch **Accept**. Then the settings for all activities, which are from the same kind (object processing, set-up, etc.), are taken over from the current.

7.2.3.2 Cleaning

Cleaning procedures are definable only at multi-functional work stations. These possess the same parameters as object processing.

7.2.3.3 Set-up

Preparation processes are definable only at workstations. These do not possess further parameters.

7.2.3.4 Elimination of Failures and Maintenance

Furthermore, when defining failures and breaks these could be eliminated manually. The differences made in chapter 8.1 between failures and breaks are also applied at work area level insofar that fault elimination and maintenance in principle take place completely analogue to one another except that failures elimination, as the name implies, refers to the elimination of faults and maintenance refers to the breaks defined at fault level. There follows therefore only a description of failure elimination in detail.



For each failure-elimination, a failure must be defined at failure level and the module which pre-defines the work place must be contained in the list of the module affected by the failure. As soon as a failure occurs which can be manually eliminated, a worker is requested, whereby the following sequence of steps taken complies with those of the object processing.

The time taken for the work depends on the duration of the failure and is not calculated until the worker arrives so that because of the previous stand-by time, the module downtime is extended.

Failure elimination can only be interrupted by a break taken by the worker, during which the rest of the work can be carried out by another worker.

For maintenance work a break of the kind break or maintenance has to be defined at failure level and the work place contained in the relevant module list.

In the third column of the task list, the module to which the task shall refer is still to be given. By selecting the third column, the user gets a module list in which all the modules are given which up to now have been registered in the work area.

The failure, which may cause the worker requirement, is to be selected in the fourth column of the task list. Only the failures are offered, which are connected with the work area.

7.2.3.5 Other Activities

All other tasks which limit the availability of the workers but do not fall under the other categories are shown as *other activities*. These are tasks which do not influence any modules such as monitoring. In such cases, the task time must be defined separately. For this there are analogue possibilities available at failure level. The following menu appears.

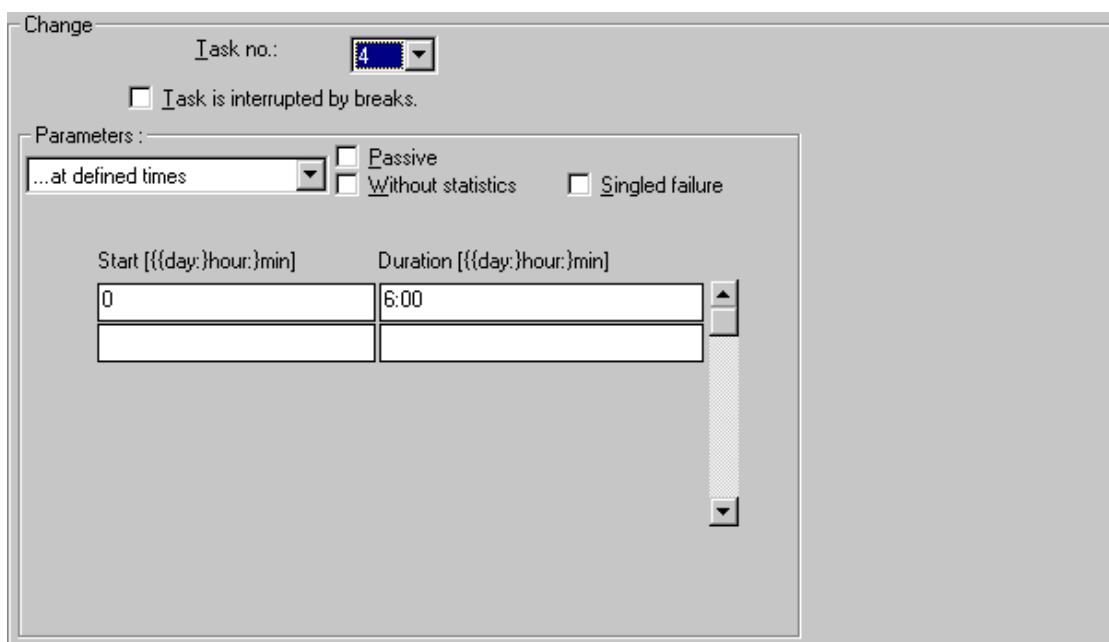


Figure 7.6: Parameterization of general tasks



7.2.3.6 *Supervising*

The supervising task takes place at the capacity-monitoring module. When the first object enters the area, for personnel employment, the appropriate number of workers as intended is requested for supervising. These can be used for all other activities freely. As long as supervising is necessary, the workers return back to the area to continue the activity. When the area is again empty, the task of supervising ends. The resulted time for this is statistically separately recorded. Thus, those activities can be recorded statistically, which need a presence of personnel for better control, without binding this worker however compellingly to the process.

The capacity-monitoring, which may cause the worker requirement, is to be selected in the fourth column of the task list. Only areas which are connected with the work area are offered.



7.3 Work Breaks

Breaks are defined analogue to failures and breakdowns. First, a list appears in which the breaks are managed. There are options to delete breaks completely from the list (Delete) or to add a new one (New).

Differences are made between fixed, random and periodical breaks, whose parameter assignment complies with the procedure described at failure level.

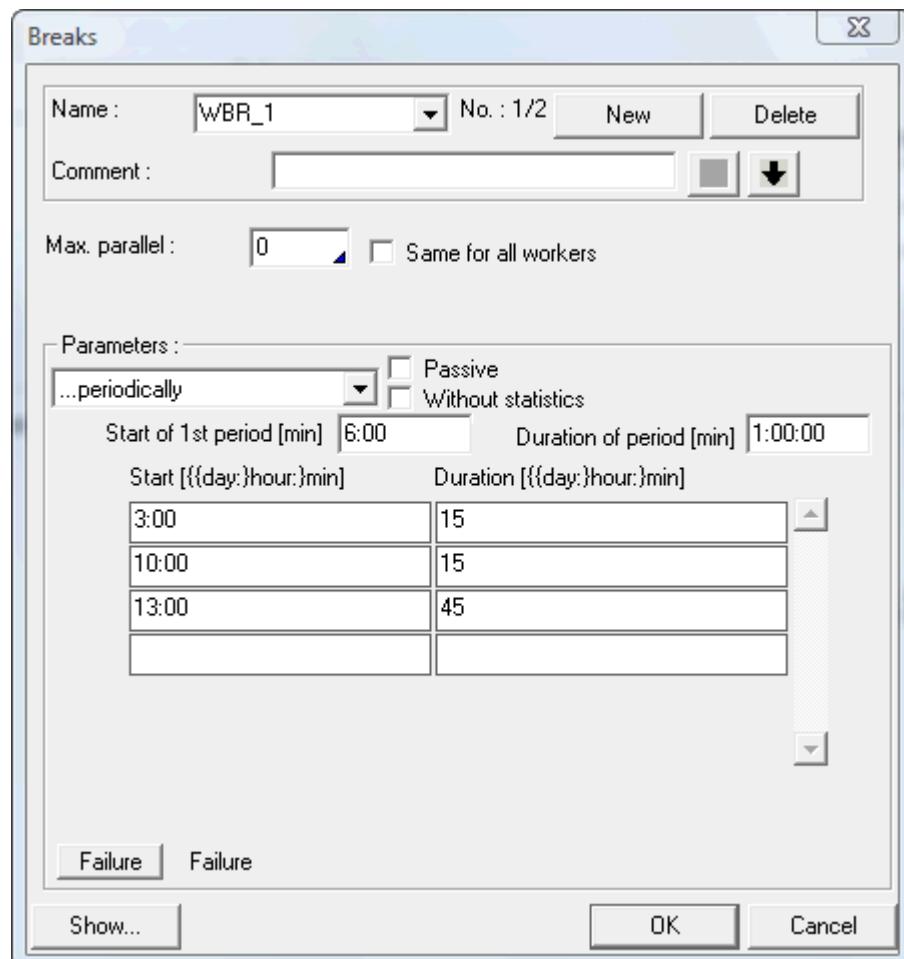


Figure 7.7: Menu Work breaks

Additionally it can be specified, how many workers will take a break at the same time. Additionally, in the field **Max. parallel**, the appropriate value is to be entered. Thus it can be prevented for example that all workers take their break at the same time.

The modeler can, in addition, pre-define whether the length of the break is the **Same for all workers**. Otherwise these would be individual for each worker. 'Individual' does not mean that here that a special break list must be drawn up for each worker, but that in the case of a task which cannot be interrupted, the break for the worker concerned can be postponed individually. The same break time for all workers may not however be chosen at random. After a break, the worker is either at his defined regular place of work, or, if he does not have a regular work place, at the site of his last task.

With the help of the arrows the breaks can be moved up or down.



The switches **Without statistics** and **Passive** have the same meaning as described for [failures](#). The button **Show** activates the visualization of the break. Thus the parameters can be more simply examined on the time bar. This is particularly meaningful for periodic shift models.

7.3.1 Show Work Breaks

The visualization of work breaks takes place in a separate dialog.

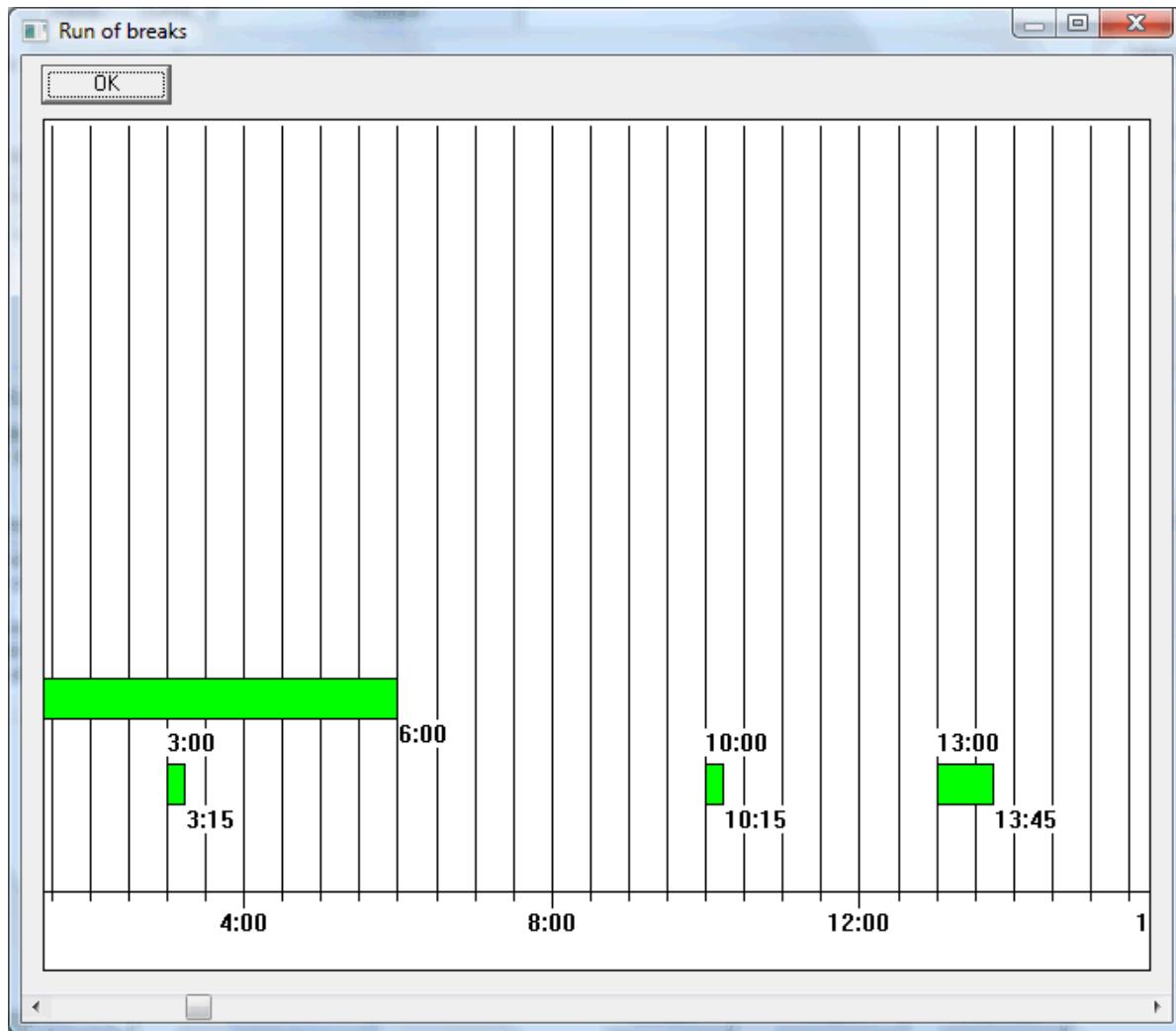


Figure 7.8: Show breaks

Here the placing as well as start and end-time can be examined more exactly. The range of the display can be steered by the cursor keys. With the combination SHIFT + PageUp, it is zoomed into the representation. Similarly with SHIFT + PageDown the display range is decreased. Starting from a certain zoom factor, start and the end-times of the breaks are shown in the display. Since all defined breaks are shown in the display, the tracing model can therefore be quite easily examined.



7.4 Groups of Workers

It often occurs that the same activities in different shifts are worked on by teams with different manning level. This can be realized by groups of workers. Each group of workers possesses its own list of workers with its qualifications as well as its own shift model. The possibility exists of **Deleting** groups of workers either completely or of adding **New** one again or of making and/or of changing the parameterization.

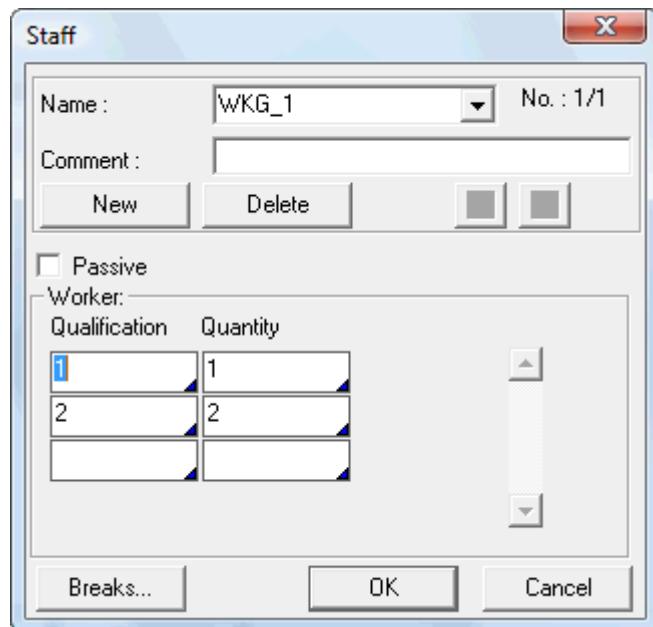


Figure 7.9: Parameters of Worker Group

The worker groups are parameterized in the same manner as list of workers and work breaks. Further information can be taken from those sections.

7.5 Route List

Each task is connected to a definite site because it has been defined by a module. In order to perform a sensible simulation, the system requires information concerning the routes between the individual work stations. These must not be neglected because they limit the availability of workers. Therefore, a route list must be defined for each work area.

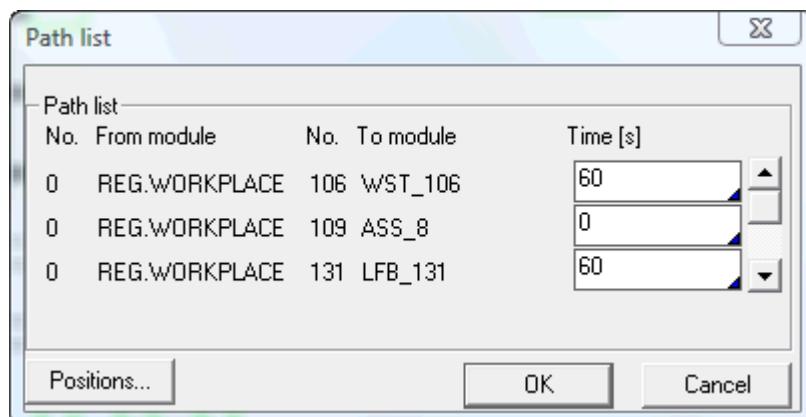


Figure 7.10: Process of route lists'



The main menu for processing work areas has a button **Distances....**. For all combinations of work places, including the regular place of work, the time a worker needs for a route must be defined in the relevant mask. These times are the same for all workers. The default is 0.

If the worker has to go to a work place in order to begin a task, the time is determined by means of this list which extends the stand-by time of the task to the worker. This route time is the standard second or third criteria after the priority number for deciding on the allocation of workers or tasks.

For the simplified input of the route times, the coordinate for each module can be indicated via the button **Positions**.

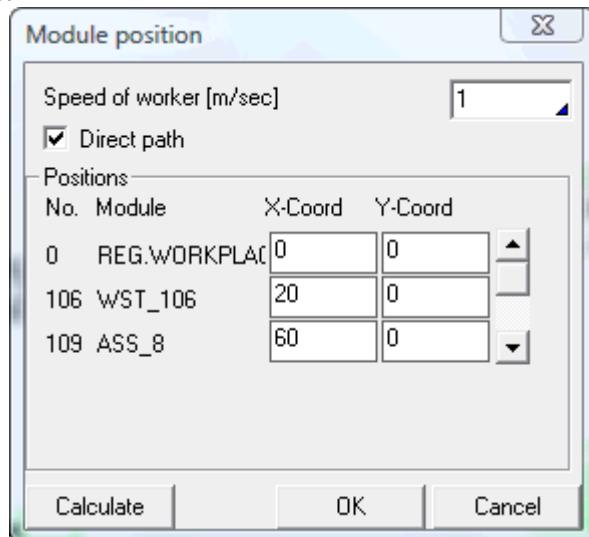


Figure 7.11: Parameters of module positions

With the help of the **Speed of worker**, the route time between all modules can be calculated. For this, the dialogue is to be left by clicking the button **Calculate**. With the computation, two alternatives are available. If the switch **Direct path** is selected, the direct distance between the two coordinates is put as distance. Otherwise the "right-angled" distance is used, which means that the worker goes first only into x-direction to the level of the module and then in y-direction toward the module.



7.6 State Cost

In the parameter dialog referring to the costs in a work area, a cost set can be indicated for each activity of the workers. This consists of the fixed costs, which is taken for each work and a time-dependent portion, which is calculated as a function of the duration of this work. These parameters can be individually indicated for each qualification. . Die level of qualification references the qualification of the worker and not the qualification, which is necessary for the task.

List of personnel costs			
State	Qualification	Fixed costs [mu/pce]	Variable [mu/s]
Standstill	0	30	120
Failure	0	30	120
Processing	0	20	100

Figure 7.12: Parameters of working costs

These costs are calculated to the assigned work area. Additionally those costs will be added to each object, which is processed. When an entry cannot be found for a qualification, the values of the next lower qualification level with the same state are taken. If no entry can be found, no costs are calculated for this state.



7.7 Result Graphic

7.7.1 Work Area Statistic

In the work area statistics, the status proportions of the individual workers are displayed. Besides the parameterization of standard results, choosing only average results in the average values of all workers in a work area being displayed.

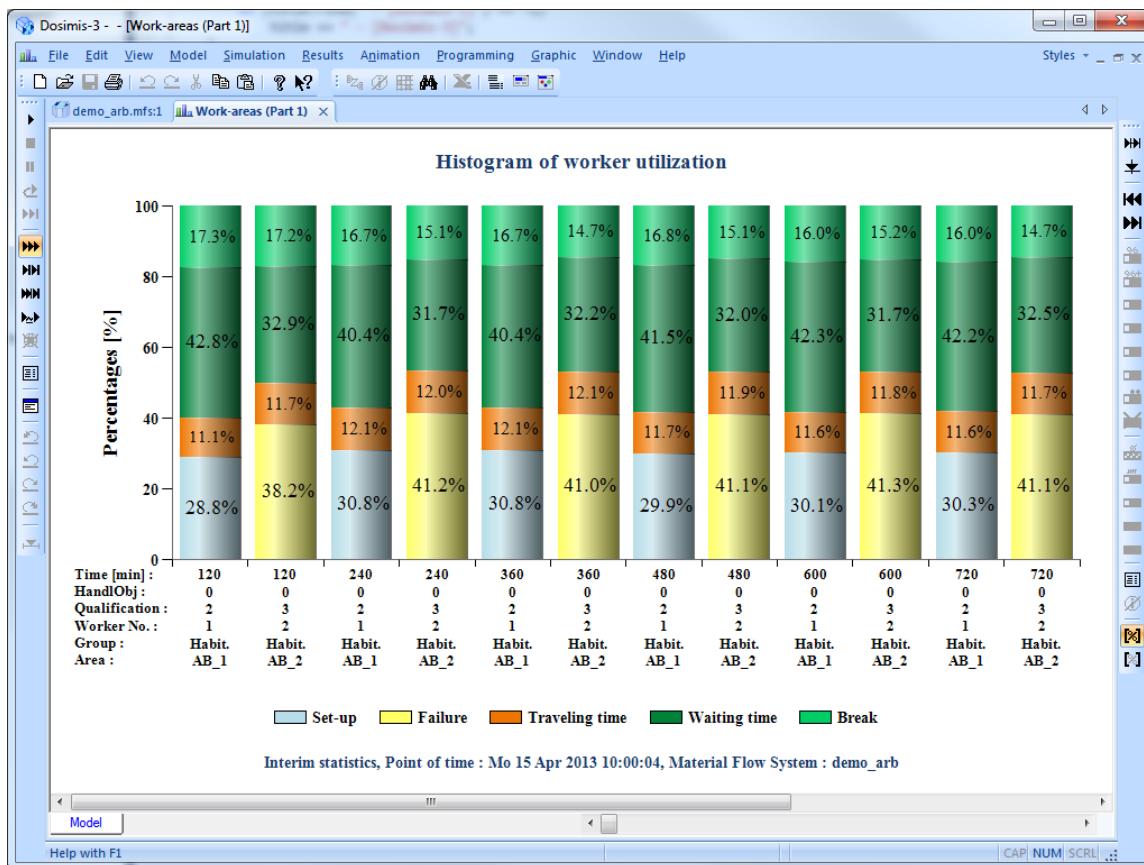


Figure 7.13: Histogram of work areas

The data are located in the *slg-file below the modules statistic.



7.7.2 Worker Employment Diagram

With the aid of the diagram of worker employment, it can be checked how many workers of a work area were actually employed. For that, the workers used for an operating range are represented similarly to the occupancy diagram.

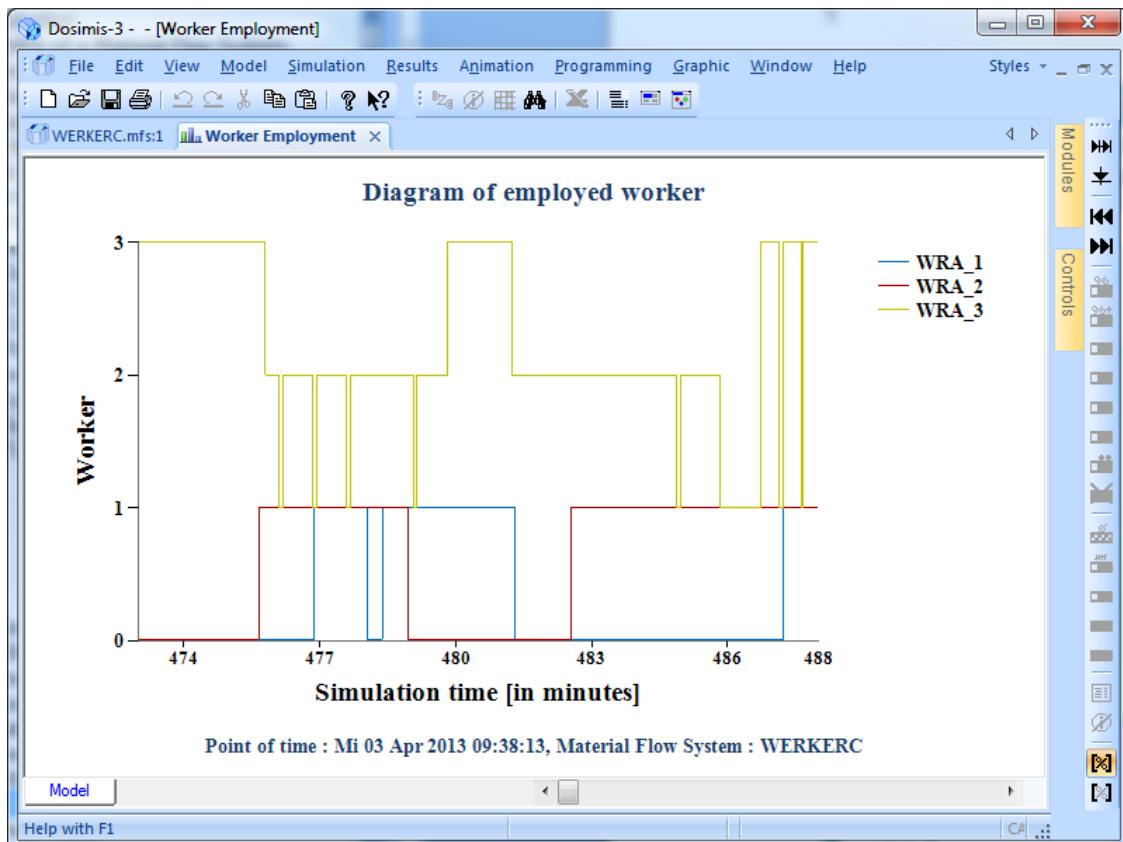


Figure 7.14: Worker employment of several work areas

7.8 Result Table

A worker can be in certain states during a statistic interval. For each state and for each worker the percentage-time portion is listed. All possible states are to be taken in the sample table.

Interim statistics		: Work areas (Part 1)					
Material flow system		: WERKERB					
Statistics for time		: 120.0					
No.	Work area name	Worker no.	Waiting-times	Object-proces.%	Set-up-times	Cleaning-times	Walk-times
							Processed objects
1	WRA_1	1	45.11	0.00	33.96	0.00	11.93 0
2	WRA_2	2	39.82	0.00	0.00	0.00	12.22 0
3	WRA_3	3	56.63	33.28	0.00	0.00	2.40 93
4	WRA_4	4	29.90	37.18	0.00	0.00	25.00 133
5	WRA_5	5	64.46	23.42	0.00	0.00	3.53 82
6	WRA_6	6	28.98	32.71	0.00	0.00	34.40 143



```
*****
Interim statistics          : Work areas (Part2)
Material flow system        : WERKERB
Statistics for time         : 120.0
*****
no. work area      Worker Other Super- Failure- mainten.- Break- Without
name           no.   action% vise % bance % times % times %
statistic%
-----
1 WRA_1            1    0.00  0.00  0.00  0.00  9.00  0.00
2 WRA_2            2    0.00  0.00  39.09  0.00  8.87  0.00
3 WRA_3            3    0.00  0.00  0.00  0.00  7.70  0.00
4 WRA_4            4    0.00  0.00  0.00  0.00  7.80  0.00
5 WRA_5            5    0.00  0.00  0.00  0.00  8.59  0.00
6 WRA_6            6    0.00  0.00  0.00  0.00  7.43  0.00
*****
```

Figure 7.15: Results output, work area statistics



8 Controls

8.1 Failure / Maintenance / Break

The performance of a system depends mainly on the availability of its modules, which can be restricted by **failure, maintenance or breaks**.

Failure means all events where certain modules are not accessible due to technical defects. Maintenances are firmly planned downtimes. Break means the time periods in which modules are switched off, for example, during the operator's coffee or lunch break, while the rest of the system is still running.

However, a difference is made between failure, maintenance and break only in the statistics. The system does not differ between whether a module is not accessible because of a failure or a break unless it is during the course of a simulation. The user can, therefore, model machine breakdowns as failures or breaks to distinguish this restricted accessibility of individual modules. Therefore in the following description we will describe only failures. The specifications are valid correspondingly also for breaks. If the processing of breaks shows differences, this is mentioned in the text explicitly.

Failures cannot only affect components that are transported by objects. There is also the possibility that failures affect controls. The following distinction is made.

- Work area: All workers go on the break, as soon as the failure is activated. If the failure is terminated, these return back to their work.
- Failure All modules, which are assigned to the failure, are disturbed. However the statistics of the disturbed failure for these modules is booked.
- Throughput-time Measurement: For the period, when the module is disturbed, the measurement is suspended. So downtimes of the plant can be figured out from the throughput time.



When defining failures and breaks the user must remember that the pre-switched areas of the disrupted module also have to be blocked especially when handling substitute strategies. If, for example, a distributor and two work stations are connected by buffer sections, the relevant buffer section BFS2 has also to be disrupted when the work station WST2 is defect, in order to avoid possible backlog.

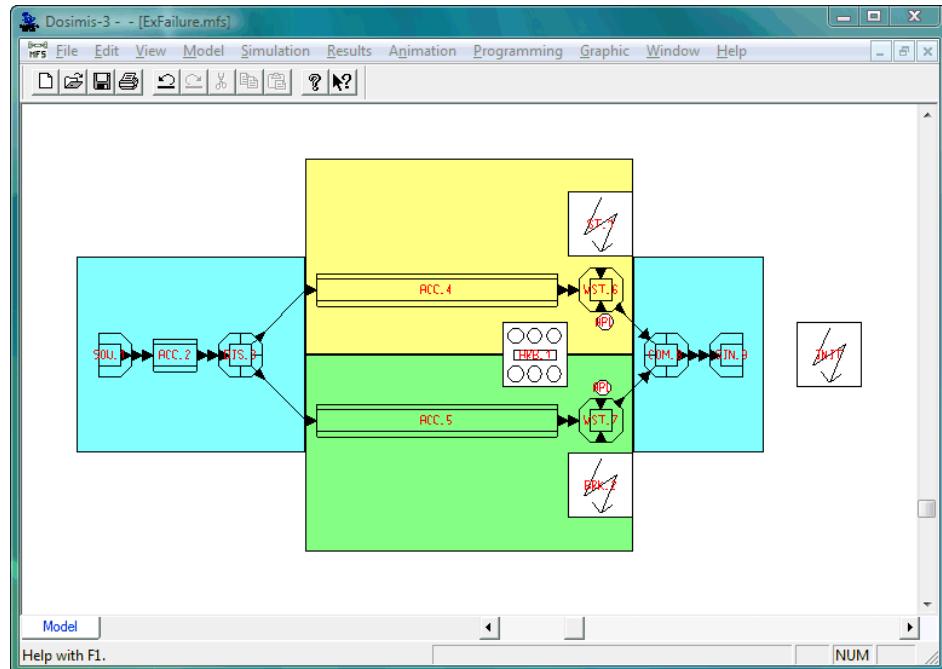


Figure 8.1: Example

If the user selects the element **Failure/Maintenance/Break** the following dialog appears:

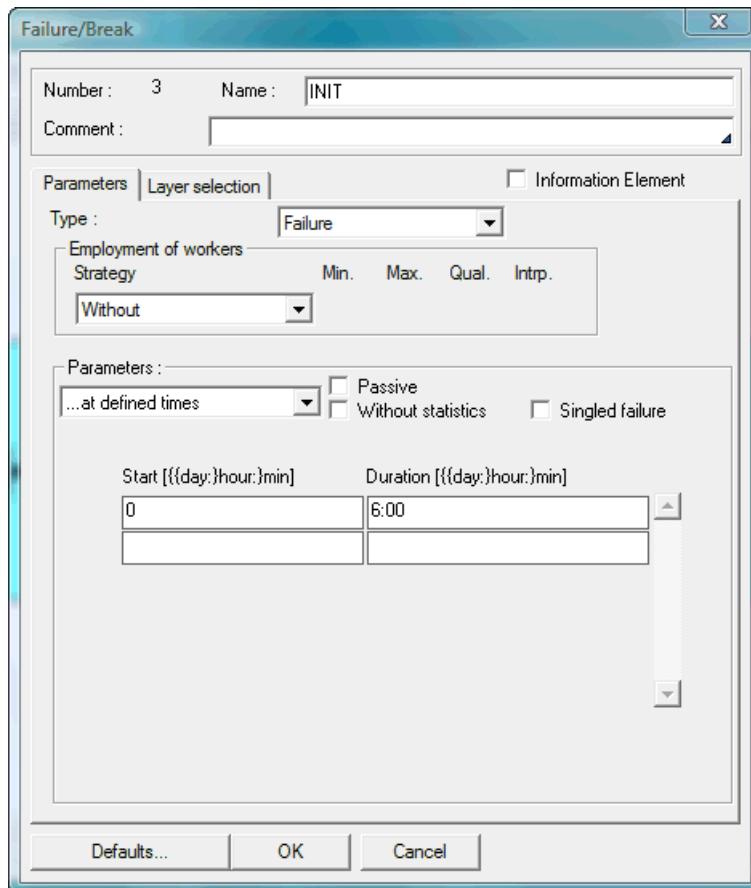


Figure 8.2: Main menu Failure „at defined times“

A failure is defined by selecting the type **Failure**, **Maintenance** or **Break**. Before the relevant input mask appears in which the failure or break can be parameterized a number for the failure is assigned. This number appears as **failure number** or **break number** in the input mask, e.g. the number **1** in Figure 8.2.

After selecting the Failure/Break icon, in the material flow system the break/failure can be deleted by using the menu **Edit/Delete**.

Failures can be disabled by the switch **Passive**.

When the switch **Without statistics** is activated for a failure, the statistics of the components this failure acts on are converted to the period of time at which this disturbance is not active.

In principle all elements, which are connected with the failure, are disturbed at the same time. If the switch **Singled failure** is active, however only one module from the list of modules connected with the failure is disturbed.

Times, which are at least given in minutes, can be entered according the format day:hour:minute as well. The display can be configured by the *properties* of the *user interface*.

A failure has two essential factors. On the one side, the duration and time of the failure, and on the other, the module that is affected by the failure.



In the mode *Linking is active* the user is able to select the affected modules. The selected modules then appear red. It could be that several failures are allocated to one module if the user wishes to distinguish between failures and breaks in the above-described mode or if different types of failures occur when a module is either directly disrupted or as in the above example indirectly due to the disruption of another module.

Finally, a decision can be made as to how the disruption can be recovered. If this is to be done manually, then the parameters can be set for the option **Employment of workers** where five parameters can be entered (see chapter work areas for a detailed description).

The parameters are (from left to right):

- Strategy: It can be selected whether no worker employment is planned (Without), or, if workers are needed, to which number of workers the breakdown duration refers (one worker, the minimum number of workers or the maximum number of workers).
- Minimum number of workers: the least numbers of workers needed to recover the failure.
- Maximum number of workers: the maximum number of workers needed to recover the failure.
- Qualification: the lowest level of qualification which the worker(s) need to have to recover the failure.
- Interrupting: It can be selected whether the current activity of a worker for the removal of the disturbances may be interrupted. For this the value is to be entered here, starting from whose priority the interruption may take place.

There is a pre-set automatic recovery, i.e. there are no entries. The condition for a correct recovery of a failure by a worker is the acceptance of the relevant module in a work area from where the worker is then requested.

When determining failure or break times the user has five alternatives at his disposal:

- defined failures random failures periodical failures failures dependent on throughput Reference Failure

The user concludes the definition of a failure with **OK**.

8.1.1 Defined Failure

The modeler must determine the time and duration of all failures and breaks if he wishes to define failures **at defined times**. These are sorted into a list after commencement in ascending order and then processed as follows from the beginning of the simulation. The failure process waits for the commencement of the next event, identifies the affected module, releases it after the duration of the failure and waits for the next failure. This ends when the list has been completely processed.



8.1.2 Random Failure

With **random failures**, two time periods must be established, the duration of a failure (failure time, MTTR, meantime to repair) and the time interval between two failures (function time, MTBF, mean time between failures) using the known distribution functions. The corresponding area of the mask then appears thus.

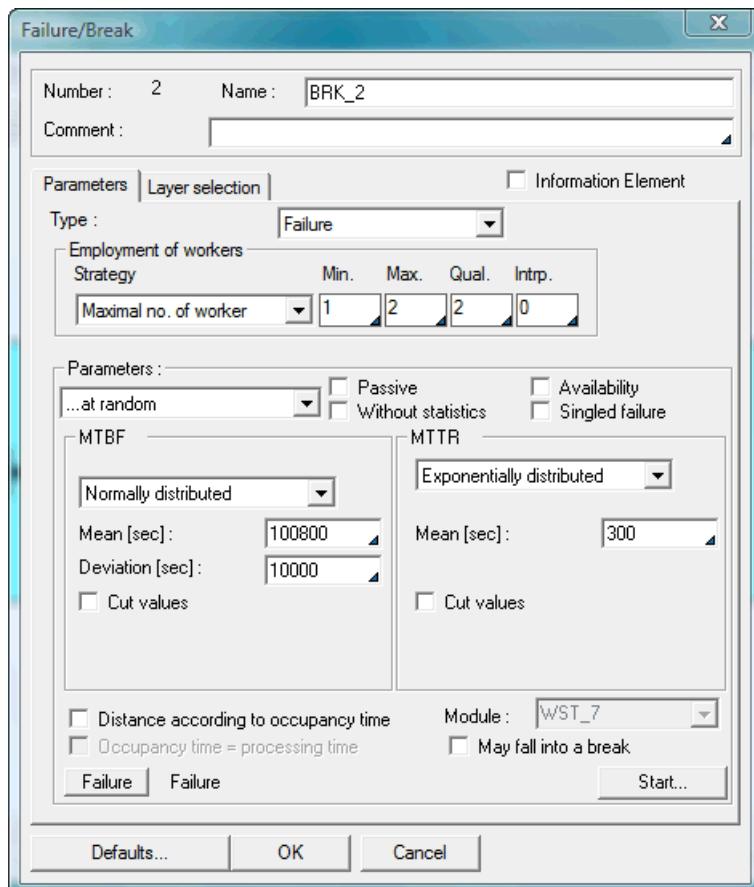


Figure 8.3: Parameters „... at random“ times

At the beginning of the simulation the first failure is defined with time and duration. If the entry in section **Start** is positive, the results are according those parameters. In the other case the point in time of the first failure will be calculated by the parameters of MTTR. The process waits for the start, identifies all affected modules and releases them after the failure time has ended (duration of break). Then, the time and duration of the next failure are determined.

Random disturbances can be parameterized by input of the availability instead of the input of the distance. For that, the **Availability** checkbox must be selected. This type of the parameterization is valid then for all disturbances in which random disturbance is defined.

If the switch **Distance according to occupancy time** is activated, the time between two distances is counted only if an object is in the module. For modules with processing on, this can be further reduced for the processing time. Then only the period is regarded, in which a set-up or work process takes place.



With the switch **May fall into a break**, it can be forced that random failures can become active also during a break. These switches exist for reasons of compatibility with old versions. This implicit rule (stochastic failures cannot occur during a break) can now be parameterized explicitly. The recommended proceeding is however to let a break affect this failure. See also for this the chapter Overlay of Failures and Breaks.

Remark:

If a model with a simulation time of 1000 minutes fails for 100 minutes, where the 100 minutes are divided into 10 single failures, this results in a total failure time of 100 minutes and a total function time of 900 minutes. The first failure begins after approximately 90 minutes and lasts for approximately 10 minutes. This is repeated 10 times. The times 90 or 10 minutes are average values dependent on the selected distribution function.

If the failure is supposed to be parameterized from availability ($V[\%]$) and mean duration (MTTR[sec]), the duration is calculated (MTBF[sec]) by: $MTBF = MTTR * V / (1-V)$. The availability of the module is 90%. Since MTTR is 10 min, this will result in a MTBF of $10 * 0,9/(1-0,9) \text{ min} = 90 \text{ min}$.

The availability can be checked by pressing the button **Failure**.

8.1.3 Periodical Failure

For periodical failures, the duration of the period and the beginning of the first period is established. The events are defined analogue to the fixed failures within a period and sorted into a list according to the sequence.

The time periods determined in the list always refer to one period, resulting in the following process sequence:

With the beginning of the first period, the list is processed analogue to fixed failures. The next failure begins at the time resulting from the addition of period begin and failure time (analogue to breaks). When the list of one period has been processed, the beginning of the next period in which the list will through again is awaited.



8.1.4 Dependent on Throughput

When a concerned module fails due to a failure dependent on throughput, the criterion is the object throughput in a counting module to be defined. To define a failure dependent on throughput, this count module, the throughput necessary for a failure, and the duration of the failure have to be established.

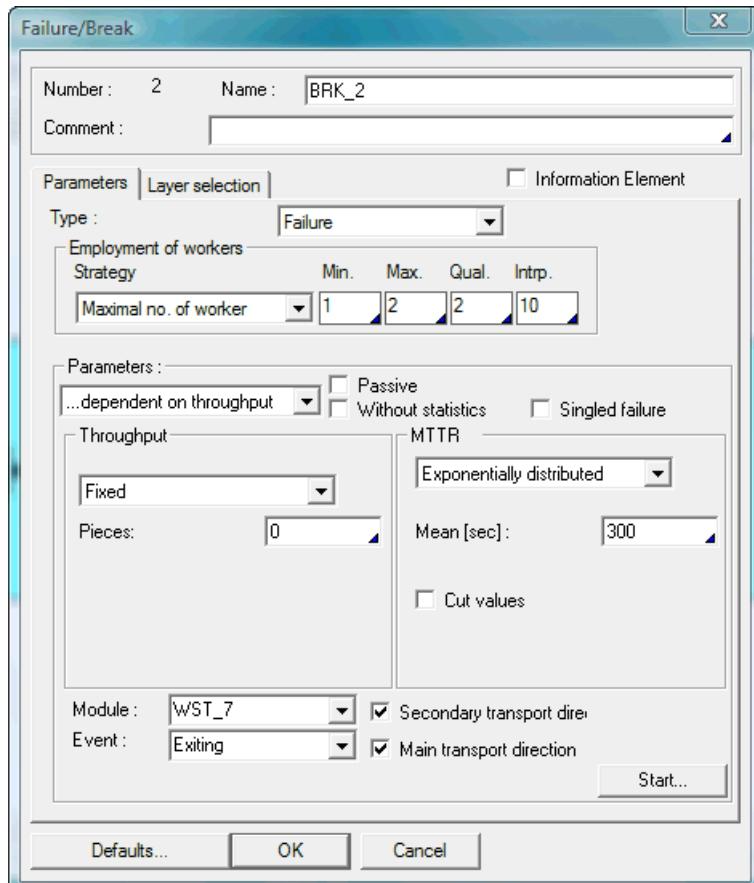


Figure 8.4: Parameters „Failure: dependent on throughput“

The count module whose throughput is responsible for the failure of the previously defined module must belong to the failure element. As soon as the given number of objects, defined in the area **Throughput**, has passed the count module, the connected modules are disturbed. This value can be determined by a stochastic process. The failure occurs as soon as the computed value of the amount of the interspersed object has reached or is exceeded. The count module does not continue with registering the throughput until the failure has ended; thus the next failure follows, when the correspondingly numerous of objects have passed the count module. The duration of a failure is established by means of the known distribution functions. With the button **Start**, an individual throughput can be determined, which can be specified once at the beginning of the simulation.

definition of the counting module occurs via the selection box in the left field. In the event field, the **Event** which is responsible for the counting can be defined. The possibilities are **Entering**, **Exiting**, **Start work** and **End work**.

Apart from the events *Entering* and *Exiting* it can be defined, whether this is to be counted in the **Main transport direction** or **Secondary transport direction**. This makes only sense



with *Entering* in combination with an assembly or a loading station as well as with *Exiting* in combination with disassembly and unloading station.

The terms are to be described here again by the example loading station. With the event *Entering*, it can be differentiated whether the vehicle is to be counted (main *direction*) or the load (secondary *direction*). So both the number of loaded vehicles and the number of charges (several batches per vehicle are possible) as well as the total of both values can be counted. When a vehicle enters the loading station, which will load there, then the event *start work* reacts once; if the vehicle does not load there, then it accordingly doesn't. The event *start work* is counted exactly once per vehicle, regardless of how many charges are to be taken up. This occurs after entering the module, when it is determined that a load process takes place and before the accommodation of the first charge. Accordingly also the event *end work* is only counted once at finish of the load processes. When the vehicle leaves the station, again the event *Exiting* occurs once. This occurs independently of loading the vehicle in the station or not. In that case only the parameter *main direction* makes sense for a loading station, since a *secondary direction* does not exist when driving out.

8.1.5 Reference Failure

a failure is defined by **parameter of a reference failure**, the failure, from which the parameters are to be taken, is to be selected. When a random failure is selected, the activation takes place with an own stochastic process, so that both failures have the same availability, but however a different time performance.

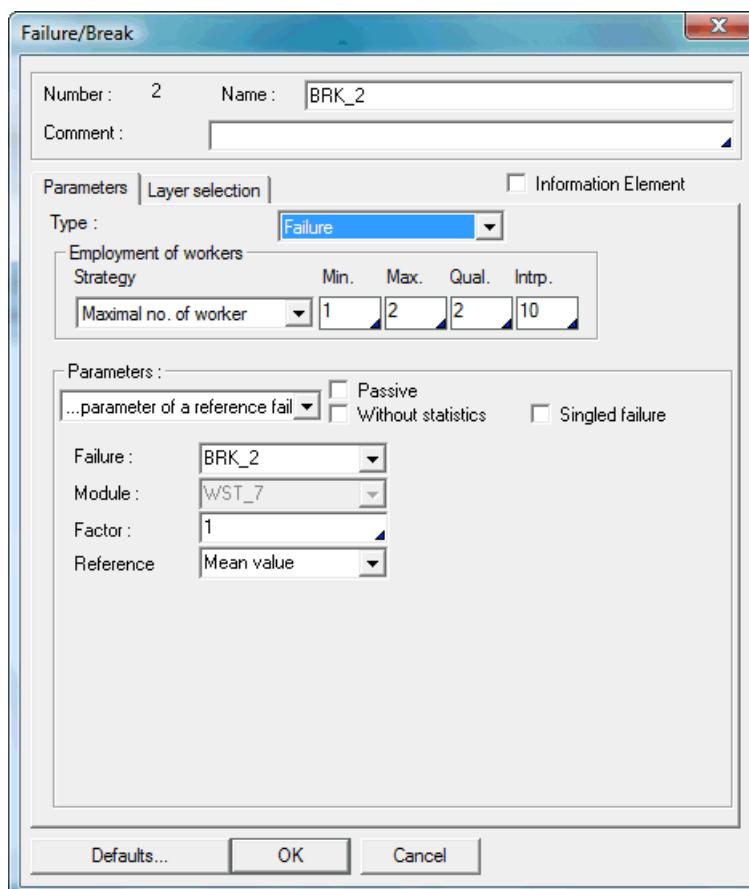


Figure 8.5: Parameters of „Reference failure“



If the reference failure is throughput-dependent, additionally a module must be determined, on which the throughput is based. Same applies to random failure, if the MTBF is to refer only to the holding time of a module.

In the field **Factor** can be indicated, how often the referenced failure is to be referred for the computation of the availability. So very simply e.g. robots - areas with different number of robots can be disturbed. If the factor is not equal to one, several restrictions to the reference failure apply:

- The kind of failure must be random.
- The definition of the average value is not made by a model constant.
- The distance may be histogram-distributed.
- In case that the distance is uniformly distributed, the lower bound must be 1.

If the reference failure is throughput-dependent, additionally a **module** must be determined, on which the throughput is to be based. Same applies to stochastic failures, if the mean time between failures is to refer only to the occupancy time of a module.

If the factor is larger than 1, **Type of reference** defines how the MTBF is to be calculated:

- Average value: The availability of the failure is calculated by the availability of the referenced failure powered with the factor. The breakdown duration remains. The mean time between failures is reduced, so that the calculated availability is reached.
- Interval: The availability of the failure is calculated by the availability of the referenced failure powered with the factor. The breakdown interval (sum of mean time between failure and mean time to repair) remains. Mean time between failure and breakdown duration are adjusted accordingly, so that the calculated availability results.
- Copy: Several independent breakdown processes are started, each disturbing the connected module independently.

The third alternative corresponds to the theoretical behavior of a failure, which is defined in such a way. Instead of this single failure, several (number: factor) independent failures have to be placed in the layout.

8.1.6 Overlay of Failures and Breaks

Basically it is possible in DOSIMIS-3 that two failures and/or breaks affect the same element at the same point in time. Here the following special features are to be considered.

If a random failure falls into a break, the starting time of this failure is shifted by the duration of the break. When then the failure becomes active, the duration and the next point of starting time are produced again in accordance with the parameterization. This circumstance is to be considered with the definition of availabilities from machines too.

In order to avoid this implicit regulation, failures can affect random and throughput-dependent failures. Such failures should be called main failures. For this the failure which is to be disturbed is to be connected with the main failures while pressing the SHIFT key in the linking mode.

The random failure is suspended for the appropriate period and takes place accordingly later. If the failure is already active, the duration extends accordingly. If a main failure is to affect



throughput-dependent failures, the counting module must be situated also in the module list of the main failure.

If personnel are needed for a failure, their request takes place at beginning of the failure. This meant that for random failures, which should start during a break, this happens after shifting, whereas with random breaks (maintenance), this always happens at the start of the break.

Breakdown times belong to the utilization of the module.

The statistics status of a module results according to the following table. When an active failure that affects the module fulfills the given criteria, the statistics are created for the given state. The first criterion, which applies to an active failure, determines the statistics, into which the period falls.

Failure-status statistics	Status of the module
Failure, without statistics	Without statistics (disturbed)
Break, without statistics	Without statistics (break)
Fixed or periodical failure, without workers or workers present	Failure
Fixed or periodical maintenance, without workers or workers present	Maintenance
Fixed or periodical break without workers or workers present	Break
Failure or break with worker, workers not present	Waiting for worker
Failure without workers or workers present	Failure
Maintenance without workers or workers present	Maintenance
Break without workers or workers present	Break

This means, for example, that the module is in the failure status, if at the same time for failure, a break is active.



8.2 Capacity Monitoring

The option capacity monitoring is to prevent system deadlocks within a simulation model. Deadlocks can also arise if too many objects remain within certain areas. If a certain number of objects stay within an area with the risk of a deadlock, all entrance links of the whole area will be blocked. The closure does not affect entrance links of modules within the area itself. If objects now leave the closed area and thus the number remains under the user-defined threshold value, the entrance links will be de-blocked. This is done in a certain order depending on the stand-by time of the objects wanting to enter this area, but which are forced to wait in front of the blocked links. The link where the longest waiting object is positioned will be de-blocked first. Then one link after the other is de-blocked, observing the order of stand-by until all are de-blocked.

After selecting the CPM-icon (**Capacity monitoring**) in the control palette, the element is to be positioned in the material flow system. After double-clicking the icon, the following parameter dialog appears:

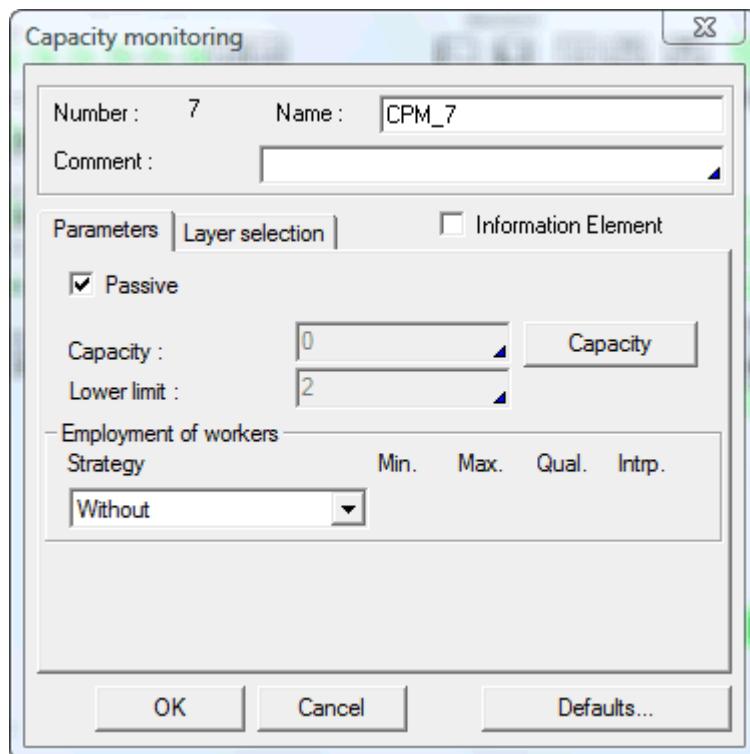


Figure 8.6: Parameters of „Capacity monitoring“

Under **Number** is the unique value, by which this item is to be found in the system.

The modeler determines the size of the individual areas, by switching into the mode **Linking is active**. Subsequently, those modules to be taken up to the area are to be clicked. Taken-up modules appear red. Taken-up modules are removed from the area by repeated clicking (compare to chapter [Linking of controls with modules](#)).

The actual deadlock control depends on the **Capacity** of the area which can be defined. During the simulation, the actual allocation of the area is constantly checked. When it is smaller than the indicated capacity, the input nodes remain unlocked. If the capacity is achieved, all input nodes become closed. If the capacity is set to zero, no object will enter the controlled area. If the number of the actual allocation drops under the **Lower limit**, then the



nodes in the order mentioned above are again unlocked. When the value of the lower limit is smaller than the upper limit, the entry into the area is only enabled when the occupancy of the area has fallen below this value. Thus the permanent release does not take place after each object withdrawal, but only if the situation within the area is a little stabilized.

With the button **Capacity**, the sum of the capacity of all modules can be transferred into the input field capacity.

For the deadlock control, the following special features are to be considered. When assembling and loading stations, access is not considered over the assembly/load input. This node does not become closed, even if the maximum capacity is achieved. Thus, for example, vehicles can be further loaded, since the charge does not contribute to the total capacity.

Furthermore, modules of the type assembly and disassembly enter only with a maximum allocation of one total allocation. Therefore, for example, the combination of a disassembly with the follow-up module at the disassembly output is critical during the composition of the modules.

The following is also to be considered when inputting the value of the capacity. It is not checked whether the input value is larger than the actual possible allocation of the area. If one inputs now a value, which is larger than the possible actual allocation, then no error message takes place. In this case, however, the area check cannot operate meaningfully. The consequence from it is a possible deadlock.

If **passive** is selected the deadlock control is used only as a statistics module. A check of the entries does not take place. During the results display, the deadlock control behaves in such a way, as if a capacity was inputted, which corresponds to the total of all individual capacities of the modules.

Within a capacity monitoring a **worker employment** can be specified. Thus with entrance of the first object, a task of the type supervising is created in the list of the tasks which can be processed. If personnel are available, they are assigned and this work is booked accordingly. The presence of personnel serves only the statistic analysis. It does not have an effect on the processes within the area. When the last object has left this area, so that this is again empty, the task is again released. (For the explanation of the terms see chapters of work areas).



8.2.1 Travel Control

The capacity monitoring can be used to control the existence of objects between two areas.

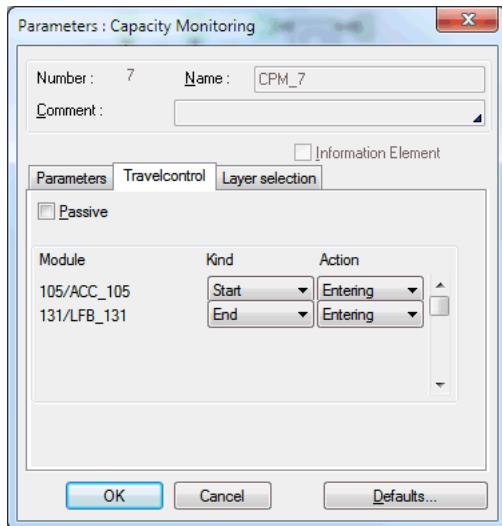


Figure 8.7: Dialog Travel Control

For this purpose must be defined on which modules the region begins and where it ends. The dialogue takes place in the sub dialog **Travel control**. Now, if the stock between the start - modules and the end- modules reaches the **capacity** of the capacity monitoring, a further feeding of objects is prevented. If then fewer objects are in the area, as indicated in the **lower limit**, the feeding of objects is released. To define the travel control, not all components of the area are connected but only the modules on the border of the area. When the switch **Passive** is set, there is no control.



8.2.2 Result Graphic

8.2.2.1 Capacity Monitoring – Minimum, Maximum

In the diagram of the capacity monitoring the minimum, average and maximum occupancy of the selected capacity monitoring are shown.

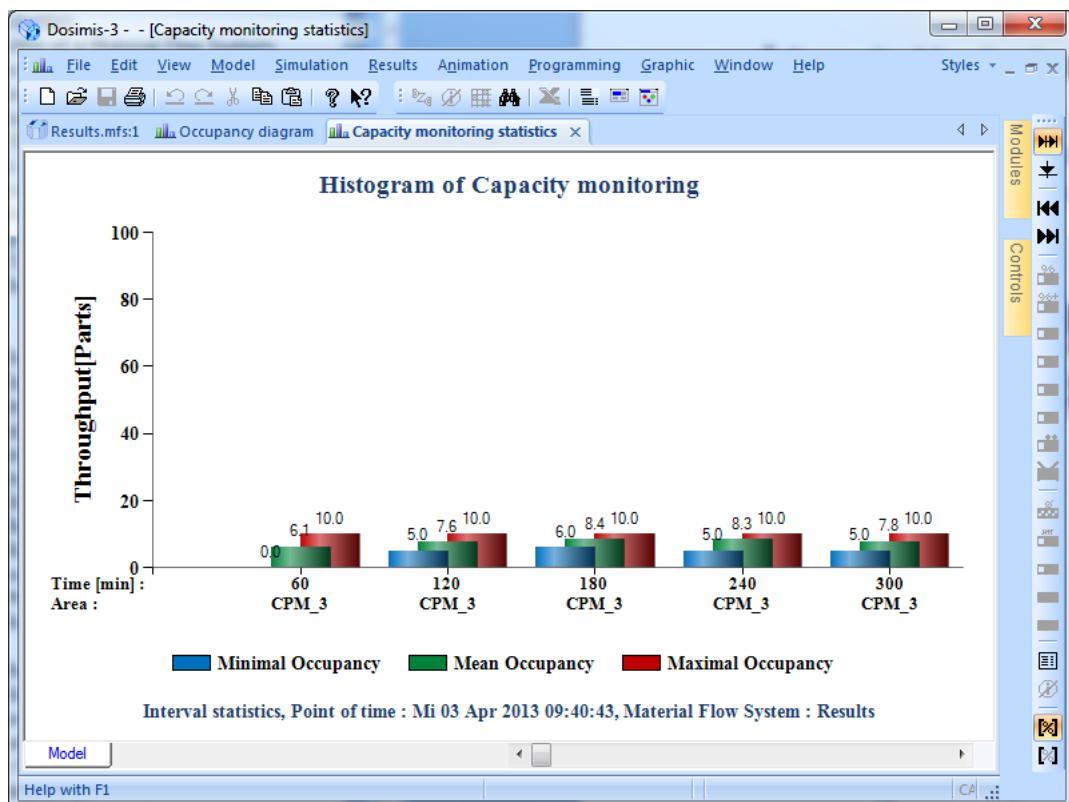


Figure 8.8: Histogram of Capacity Monitoring



8.2.2.2 Capacity Monitoring

The allocation diagram of the capacity monitoring displays the allocation within the area over the time. There are two versions, absolute and relative allocation in analogy to the allocation diagram of the elements. These can also be smoothed.

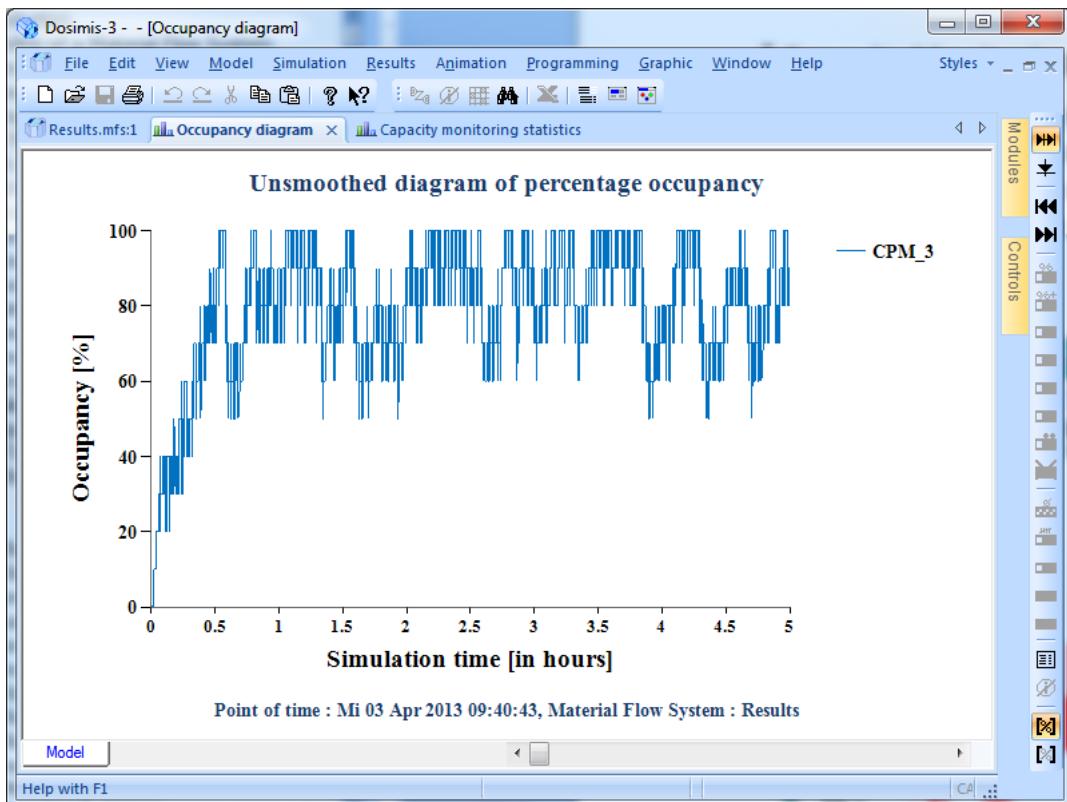


Figure 8.9: Capacity Monitoring

8.2.3 Result Table

In the capacity monitoring statistics for each area, actual, minimum, average and maximum contents are logged.

```
*****
Interval statistics          : Capacity monitoring statistics
Material flow system        : Results
Statistics for time interval : 60.0 - 120.0
*****
No   Capacity             Act. Max. Min. Aver. Percent.
monitoring      con. con. con. con.    cont.
-----
3   CPM_3                6   10   4   7.92   79.23
*****
```

Figure 8.10: Results output, Capacity monitoring

8.2.4 Statistic

For the deadlock control, following statistics and logs are carried out. At each statistical point in time, the minimum, average and maximum allocation is recorded. For this, the deadlock control must be taken up to the statistics list. Additionally each movement can be logged



within the deadlock control. From this then an allocation diagram can be led across all modules, which are situated in the deadlock control.



8.3 Connector

The connector is used to combine parts of the model. Instead of deleting source and sink and joining the modules again, a connector can be put in. This is linked to the source and sink, which it is supposed to connect (see linking of controls).

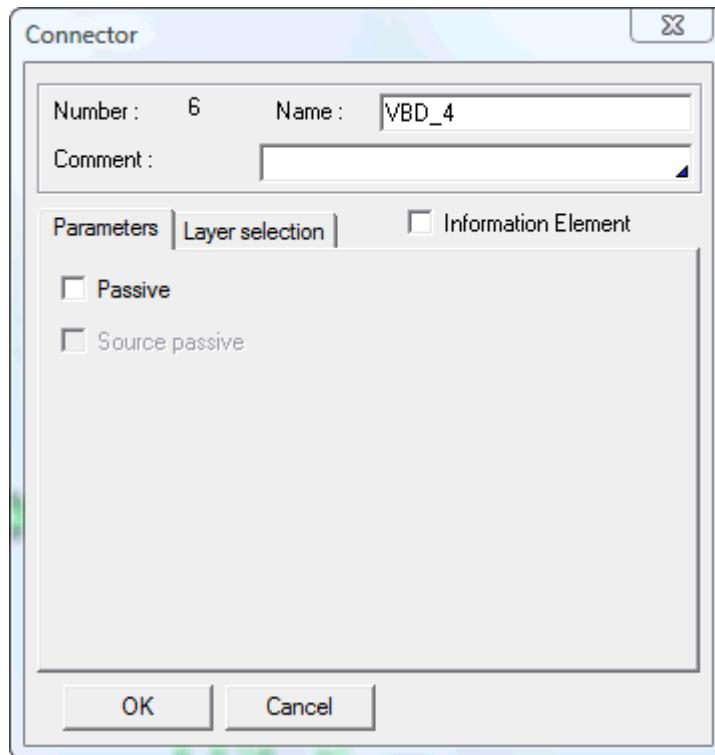


Figure 8.11: Parameters of the connector

The connector has only 2 parameters:

Passive: Connector is not active. Source and sink are not bypassed.

Source passive: The source of the following part of the model is set as passive (only reasonable, if active is not selected). Only the first part of the model is simulated (simulation duration).



8.4 Shuttle Control

The passing on of objects between several modules can be controlled by a shuttle control. Thus, a passing on can be carried out at all modules only if the first object is ready to be transported or the component is empty (emptying cycle).

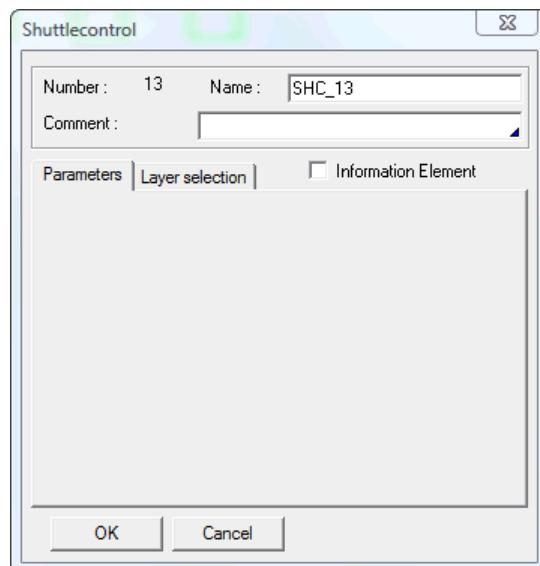


Figure 8.12: Parameters of Shuttle control



8.5 Storage Control

The storage control supervises the interaction of a storage and the components prior to it. This is essentially the I-Point (Indicator point), at which one decides, into which storage aisle an object is to be stored. A storage can have several I-Points.

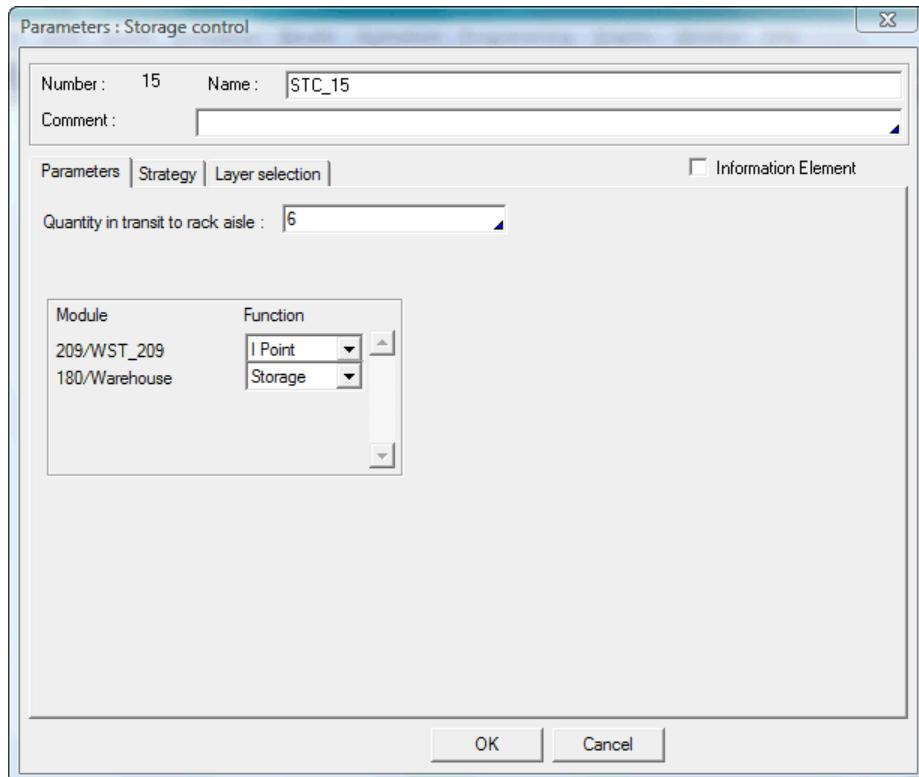


Figure 8.13: Parameters of Storage Control

The parameter **Quantity in transit to rack aisle** determines, how many objects may be on the way between the I-Point and a rack aisle. Thus it is to be prevented that in the incident more objects on a storage pass want, than there has place. If the quantity in transit for a rack aisle is reached, then this aisle will not be considered with the further scheduling. If the quantity in transit for all rack aisles is reached, the exit junctions of all I-Points will be blocked. These are again released, as soon as an object from the storage was again released from stock.

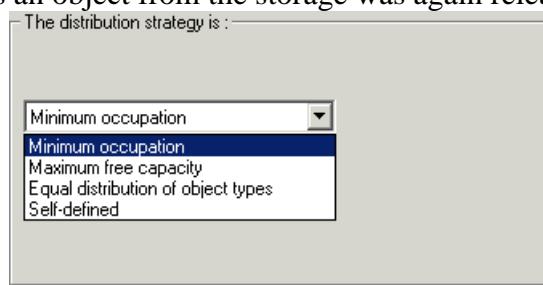


Figure 8.14: Parameters of storage strategy

With the help of the distribution strategy one determines the rack aisle where an object should be stored. For this several alternatives are available.

In the case of **Minimum occupancy** the rack aisle with the smallest occupancy is selected. With **Maximum free capacity** the rack aisle, which has most free storage bins, will be



selected. During **Equal distribution of object types** the rack aisle is selected, which possesses the smallest allocation of objects with the same object type as the type of the arrived object.

If the decision is not clear, with the first two strategies the first found stock area is selected. With the *Equal distribution of object types* the area with the smallest total allocation is selected. In all three cases the objects, which are between I-Point and storage, are taken in account. The exact storage location will be made if necessary when the object is taken over by the storage.

As last alternative a **Self-defined** strategy is available. Here the area can be determined with the help of a decision table.

Following the evaluation the destination parameter of the object is set on the number of the storage area, to which the object must be navigated. Therefore the distribution of the objects toward the storage must be made by the destination oriented distribution (*with destination: dest. Para.*). It is to be ensured, that only the objects, which possess also the appropriate destination, reach the input area of the storage. Furthermore it is to be considered with the parameterizing of the model that the destination parameter of the object is not put back after placement into the storage.

8.6 Throughput Time Measurement

The module *throughput-time measurement* serves to measure the throughput time of the objects between any measuring points. For this, at least two modules must be connected with the element. One is where the measurement begins and a second, where it ends. However, several modules can also be inserted, which must be more exactly specified by the parameters.

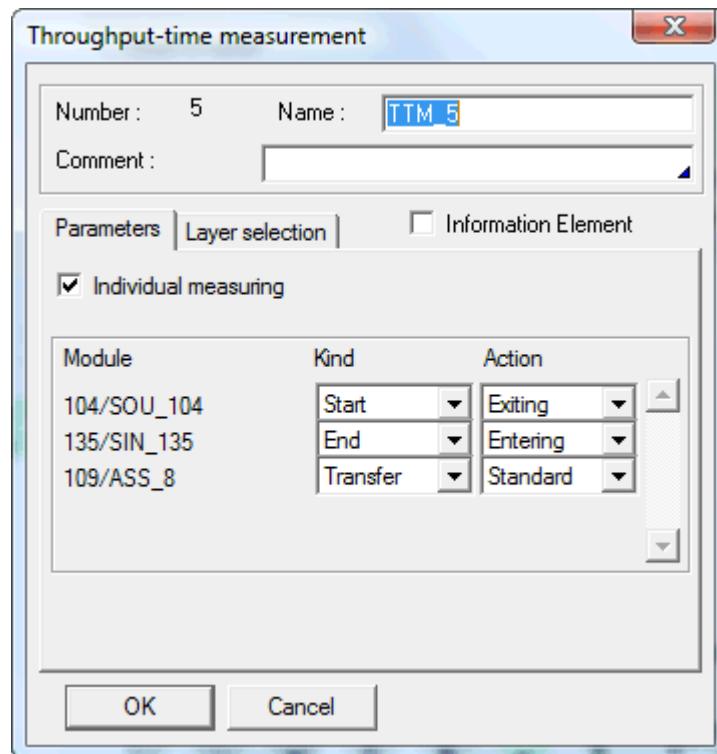


Figure 8.15: Throughput-time measurement



Each module possesses two further parameters. On the one hand, it must be determined, whether the module is a start module, i.e. whether the measurement is to be begun there. Otherwise the module is a final module, which means that the measurement is to be made there. Furthermore the event in the module can be selected, with which the measurement is to be begun or terminated. Here the alternatives *entering* and *exiting* are available.

The type *transfer* represents a special case. Thus the measured time of the basic object and/or the assembled objects can be transferred when assembling or disassembling. This corresponds to the switch *throughput-time transfer* of the two types of modules. With this situation, two possibilities exist. In the standard case, the times are transferred as described. The special case means that when assembling, only the times of the assembly objects (entrance 1) will be transferred. For disassembly, the value will be transferred to the disassembled objects. The measurements of the basic object (exit = 1) will be deactivated.

In the statistics for each object, which reaches a target module, the throughput time is measured. It is reported for the type of object, which the object possesses at the point in time of the measurement of the final module.

For sources, the action *entering*, with sinks the action *exiting* is not permitted.

If certain failures from this throughput time are to be disregarded (for example work breaks), the module is to be connected to this failure. With end of the failure for all objects, the breakdown duration is taken off from the throughput time. For good reason therefore, the counting modules should be also located in the failure.

The module is passive. This means it does not influence the flows in the simulator.

8.6.1 Statistics

For each throughput-time module, the statistics can be switched off. This takes place by means of the fact that the module from the statistics list is delivered.

If the **individual measuring** is activated, an internal message log will be carried out for each module. This supplies a result diagram, in which each individual measured value is represented.

Additionally the costs between the measuring modules will be calculated and drawn in a separate statistics for Cost Measurement or Cost Clouds.



8.6.2 Result Graphic

With the help of the diagram each individual measurement is visualized. Thus it can be analyzed more exactly, at which point in time particularly large throughput times occur or how, for example, disturbances affect the behavior.

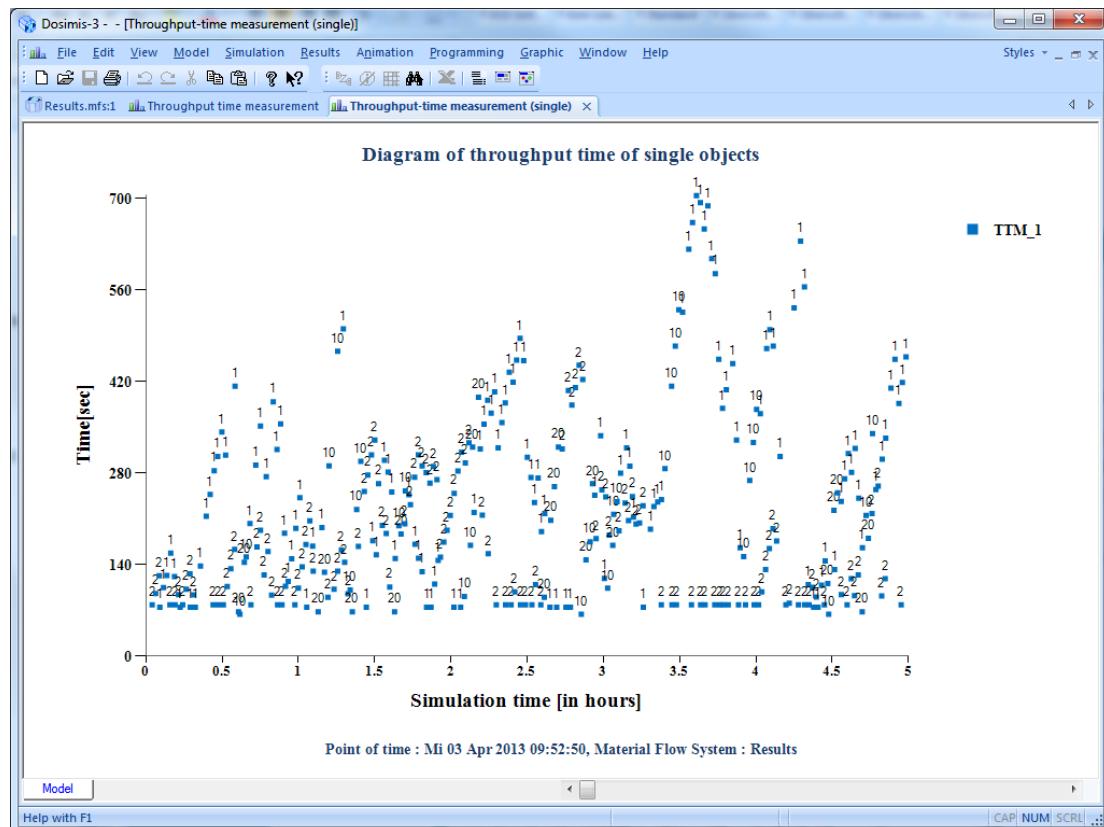


Figure 8.16: Throughput-time measurement (single)

On the x axis the point in time of the measurement is represented, while in y-direction the throughput time of the measured object is held.



8.6.3 Result Table

In throughput-time measurement, similar to the general throughput time measurement, the maximum, minimum and average throughput time between the defined measuring elements are logged.

```
*****
Interval statistics          : Throughput time measurement
Material flow system        : Results
Statistics for time interval : 60.0 - 120.0
*****
Nr. Modul-
name      Object-
type       Number of   Average    Minimal   Maximal
               objects     rtt       rtt       rtt
-----
1 TTM_1           1          21  200.703  74.929  566.217
                  2          29  314.465  78.500  627.557
                  10         2   91.279  64.214  118.343
                  20         6  213.636  95.931  341.400
2 TTM_2           1          20  282.309 132.053  735.643
                  10         3  362.782 190.709  624.479
*****
```

Figure 8.17: Results output, Throughput-time measurement

Additionally the costs measured in this section are logged. These data are located in a further table following the measured times.

```
*****
Interim statistics          : Costs measurment statistics (data in [mu])
Material flow system        : Results
Statistics for time          : 60.0
*****
No. Modul-
name      Object-
type       Number of   Average    Minimal   Maximal
               objects     costs     costs     costs
-----
1 TTM_1           1          26      0.00      0.00      0.00
                  2          26      0.00      0.00      0.00
                  10         2      0.00      0.00      0.00
                  20         2      0.00      0.00      0.00
2 TTM_2           1          23  20844.70 15187.87 26521.77
                  10         4  13545.52 10690.01 15393.33
*****
```

Figure 8.18: Results output, Costs measurement



8.7 Measuring Element

The measuring element serves to measure characteristics between arbitrary measuring points. For this, the modules where measurement should take place must be connected with the element. The measuring element covers the functionality of the throughput-time measuring element, is however to be more flexibly parameterized. In particular, it permits user-defined statistics by the use of attributes.

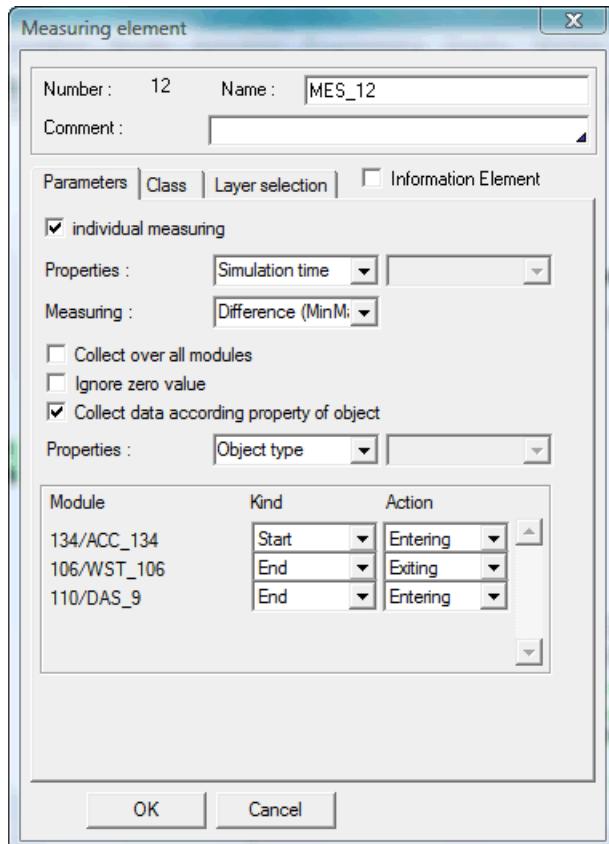


Figure 8.19: Dialog of a measuring element

The following characteristics can be measured:

- Piece: Each object increases the measured value by one (e.g. throughput measurement in combination with accumulation).
- Simulation time: Each object increases the measured value by the simulation time (e.g. throughput-time measurement in combination with difference formation)
- Costs: Each object increases the measured value by the current object costs (e.g. change of value in combination with difference formation)
- Float attribute: Each object increases the measured value by the current value of the selected float attribute (e.g. change of value in combination with difference formation)

The measured characteristics can be statistically processed differently

- Sum: The sum of all characteristics is measured



- Value (min max): The value of the characteristic is measured. From this, a minimum - maximum statistics is made, similarly for the object throughput time
- Difference (min max): The change of the characteristic is measured. For this, it must be specified, at which modules the value is set and at which module the difference to this value is analyzed. Here the same explanations apply, as those made for the throughput-time measuring element. The above parameterization corresponds to a throughput-time measurement.

Furthermore it can be specified whether the measurement is to take place individually for each module or is to be summarized over all modules (e.g. throughput measurement over parallel areas).

Additionally measured characteristics can be evaluated for each object characteristic:

- Type of object: For each type of object, which was recorded in the measurement, a separate statistics is carried out.
- Integer attribute: For each object, which was recorded in the measurement, the statistics are summarized in accordance with the value of an integer attribute at the time of the measurement. This can be specified more exactly. During difference measurement, this is then the value in the final module.

If certain disturbances are to be disregarded from the difference measurement (for example work breaks), these disturbances have to be connected with this module. Then with the end of the disturbance of all objects, the breakdown duration is taken off from the difference. Naturally therefore, the counting module should be located also in the disturbance. This functions only during the difference measurement the simulation time.

The module is passive. This means it does not affect the processes in the simulator.



8.7.1 Measuring Classes

Often the number of measured values is very large. Therefore it is quite problematic to analyze these values individually. Often it is however sufficient if values, which lie near together, are summarized. Therefore these can be grouped into classes. The measuring classes work only in combination with the differences statistics.

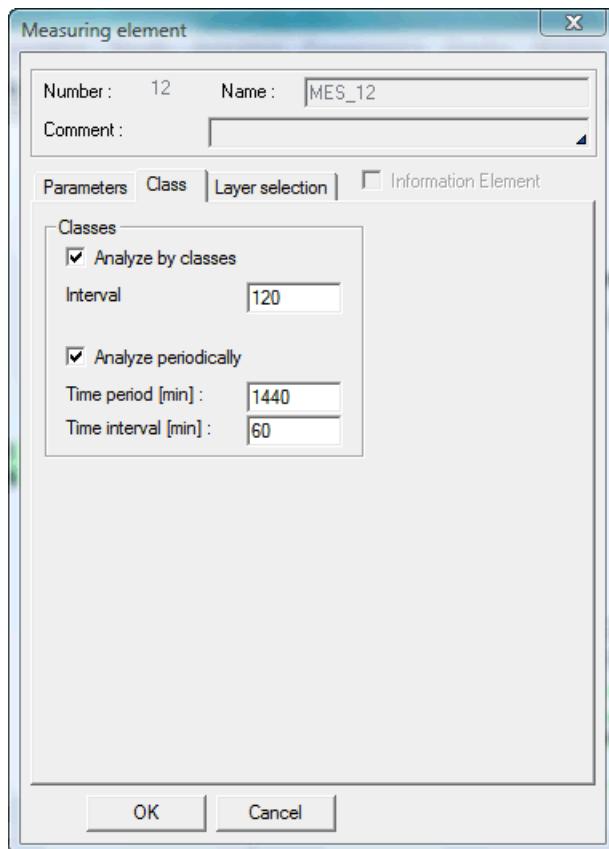


Figure 8.20: Dialog of measuring module classes

The interval length indicates which values fall into the same class. All values between the last interval end and the current interval end are summarized. The statistics consist then of many values, which result from maximum measured value divided by interval length. Each value contains the number of the measurements, which have fallen into this interval. This proceeding is frequently suffices in order to determine, for example, how high the proportional portion of objects is, which exceed a demanded throughput time.

Sometimes it is helpful to consider the time of the measurement during the class formation. Thus, it can be meaningful with a daily returning load to determine whether the behavior is independent of the daily hour. Therefore, the measuring point of time can additionally be consulted during the class formation. In addition, the time periods (e.g. 1440 minutes/1 day) and the time interval (e.g. every 60 minutes/1 hour) are to be indicated. Thus the behavior in this example can be individually examined for each hour. If the time period and time interval are entered as 0, this allocation does not take place.



8.7.2 Result Graphic

If in the measuring element a class measurement is parameterized, this can be indicated in a diagram. On the x-axis the time of the measurement is represented, while in y-direction the values of the object measured there are shown.

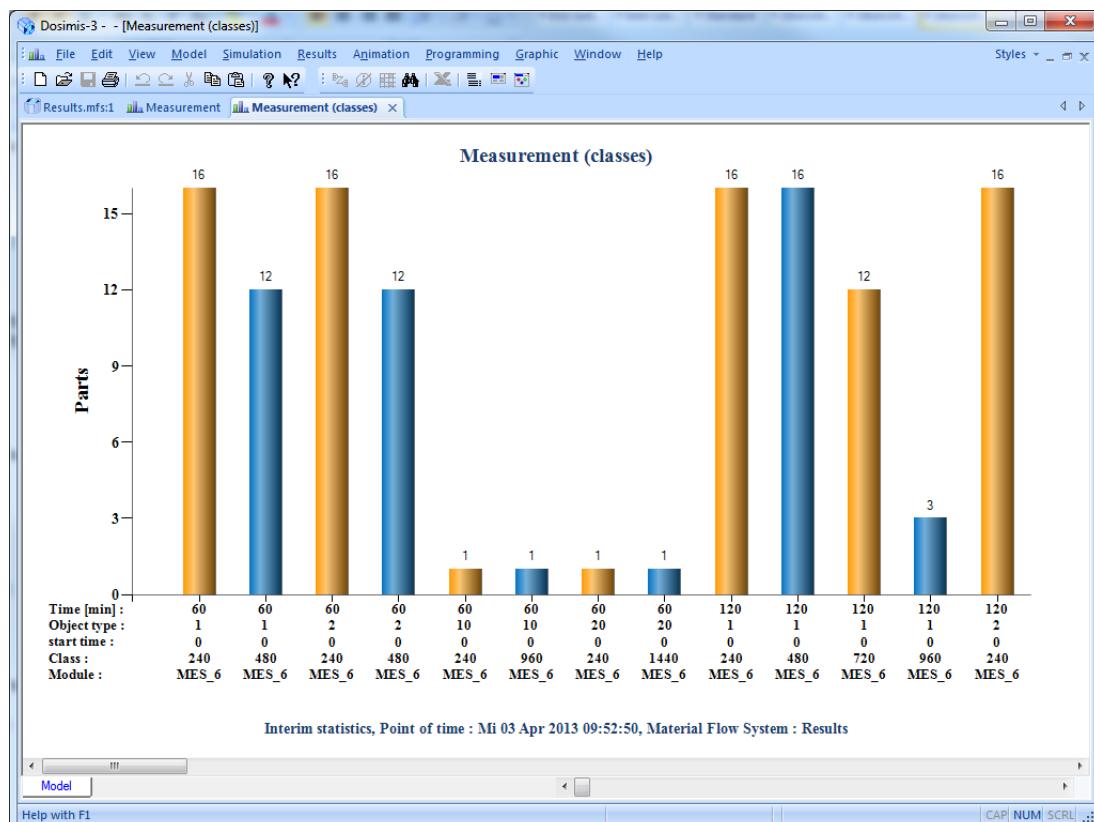


Figure 8.21: *Measured values*



If the classes do not distribute themselves over many values, these can be summarized in a compact diagram.

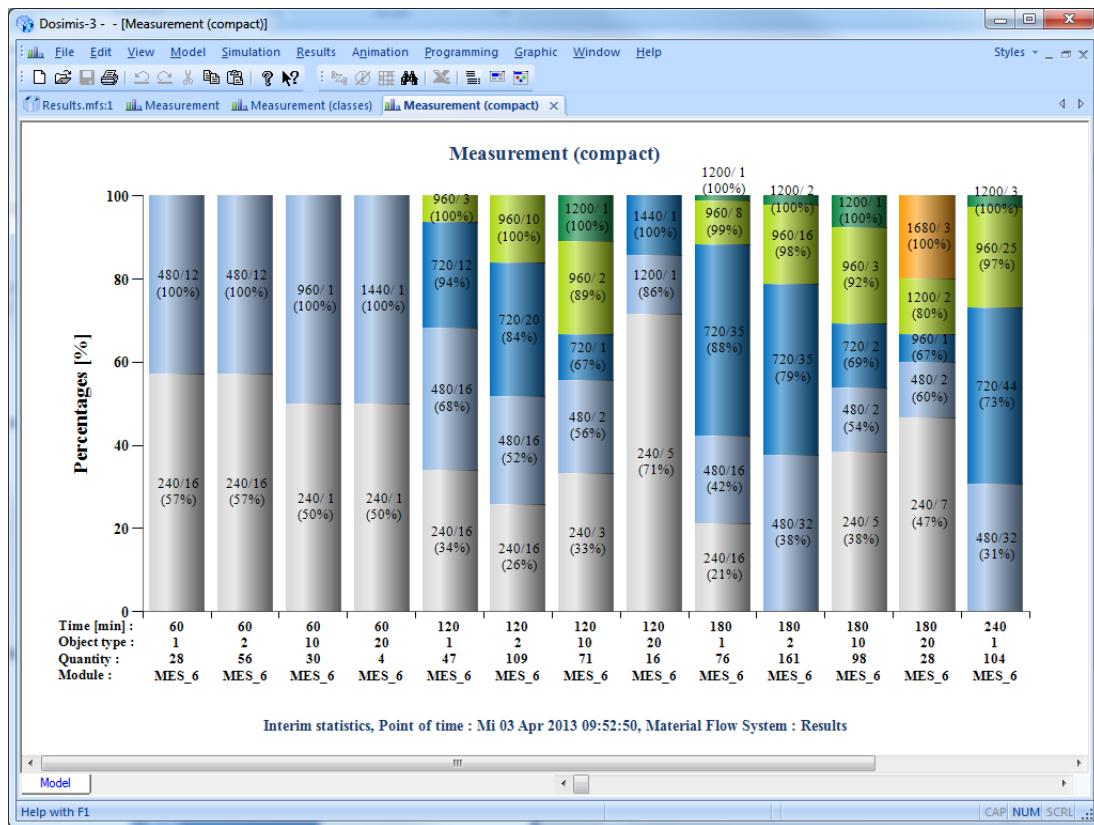


Figure 8.22: Measuring element (compact)



8.7.2.1 Measuring (single)

With the help of the diagram, each individual measurement is visualized. Thus can be analyzed more exactly, at which time particularly large measured values were adjusted, or how, for example, disturbances affect the behavior.

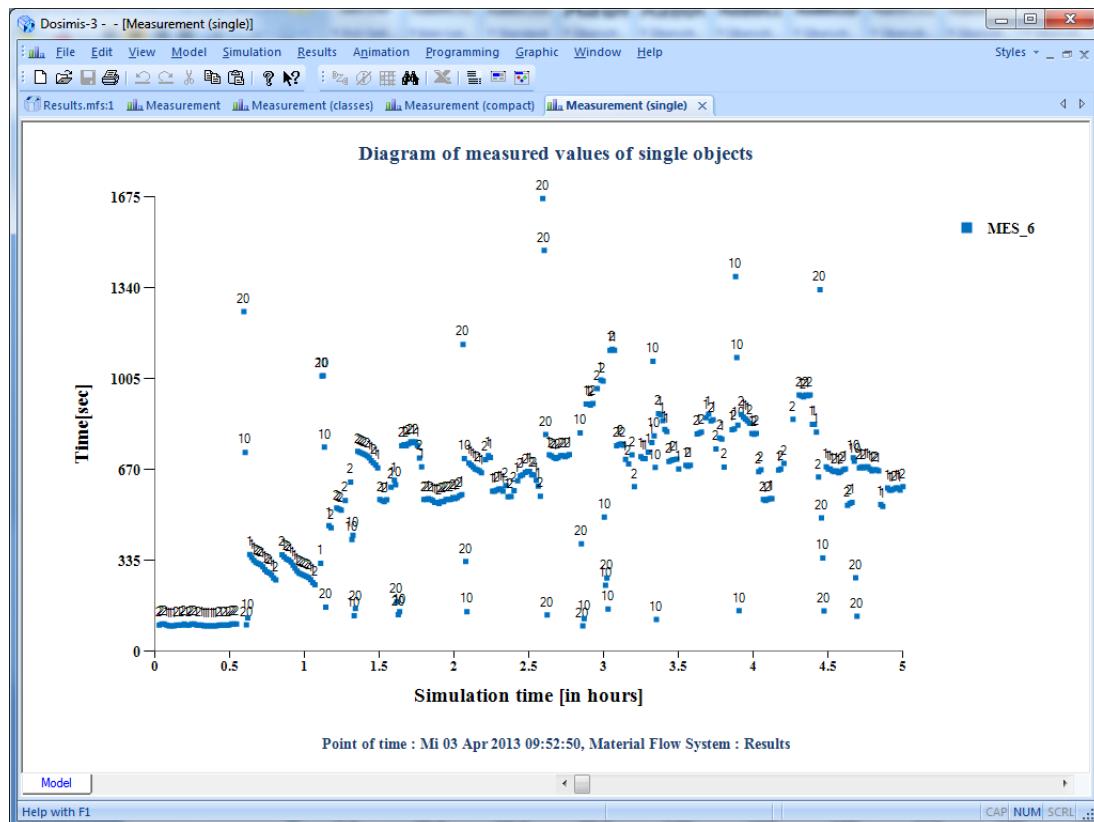


Figure 8.23: *Measuring module (single)*

On the x-axis, the time of the measurement is represented, while in y-direction, the values of the object measured there are shown.



8.7.3 Result Table

In the measuring element, the module, the type of object and the number of measurements are indicated as a function of the selected measuring method. If minimum/maximum statistics is selected, the average, minimum and maximum measured values are logged.

```
*****
Final statistics : Measurement
Material flow system : Results
Statistics for time : 300.0
*****
No. Modul- No. Module- Object- Count Mean Minimal Maximum
name name value Meas. measure measure measure
-----
5 MES_5 15 ACC_15 1 126 0 0 0
          2 128 0 0 0
9 ACC_9 1 123 0 0 0
          10 27 0 0 0
19 BUL_19 10 27 0 0 0
          20 21 0 0 0
7 ACC_7 2 127 0 0 0
          20 21 0 0 0
6 MES_6 0 <all> 1 126 613.47 93.86 1049.29
          2 128 617.68 97.43 1045.72
          10 27 576.37 107.72 1282.63
          20 21 574.90 98.35 1726.19
*****
```

Figure 8.24: Results output, Measuring module

The measuring classes are available only with differences-statistics. If no periodic interval is indicated, the initial value is 0. Otherwise the time of the measurement is found below the column of initial value within one period.

```
*****
Final statistics : Classes of measurement
Material flow system : Results
Statistics for time : 300.0
*****
No. Module- Object Classes Count
name type Startvalue Measuring
-----
6 MES_6 10 0 1440 1
          1 0 1200 5
          20 0 1200 1
          1 0 960 38
          20 0 960 1
          10 0 1200 2
          2 0 1200 6
          1 0 720 53
          10 0 960 8
          20 0 720 1
          2 0 960 39
*****
```

Figure 8.25: Result outputs, Measuring classes

8.7.4 Statistic

For each measuring element, the statistics can be switched off. This takes place by means of the fact that the module is deselected from the statistics list.



If the **individual measuring** is activated, an action log will be maintained for each module. This supplies a result diagram, in which each individual measured value is represented.

8.8 Monitoring

The monitor module offers an easy way to follow changes of values from simulation.

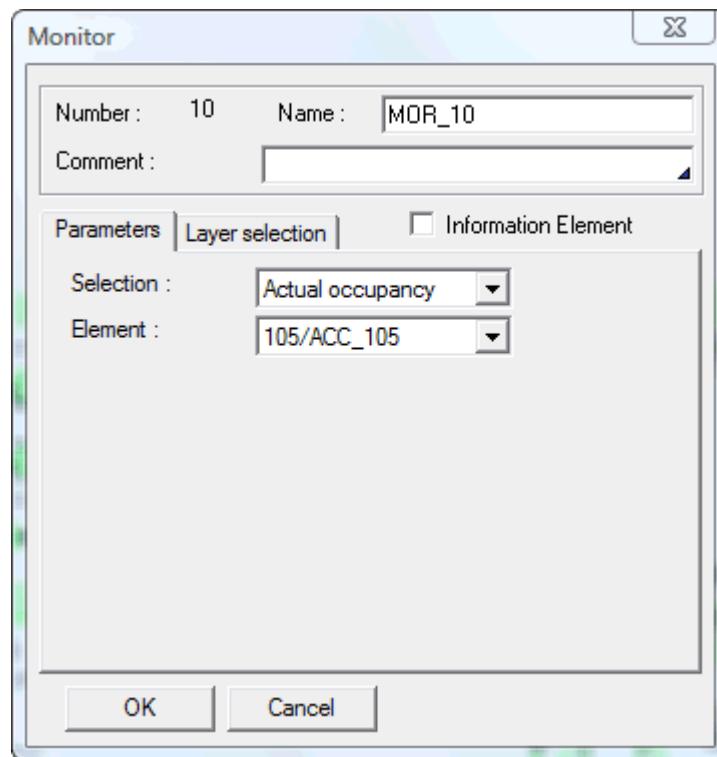


Figure 8.26: Parameters of monitor module

For this, the appropriate value is to be selected. Additionally, the element is to be specified more exactly in the field element. The logging happens then during the animation in the protocol window. Every time the value changes an entry is made. From the module statistics, the current occupancy and the throughput can be selected. The further alternatives serve for the analysis of the decision tables and are described here. **Evaluation**



The evaluation module serves to parameterize result evaluations properly so that this can be displayed faster. The elements to be evaluated are to be connected with the evaluation module in the connecting mode. From the selection, one of the possible result representations is to be selected.

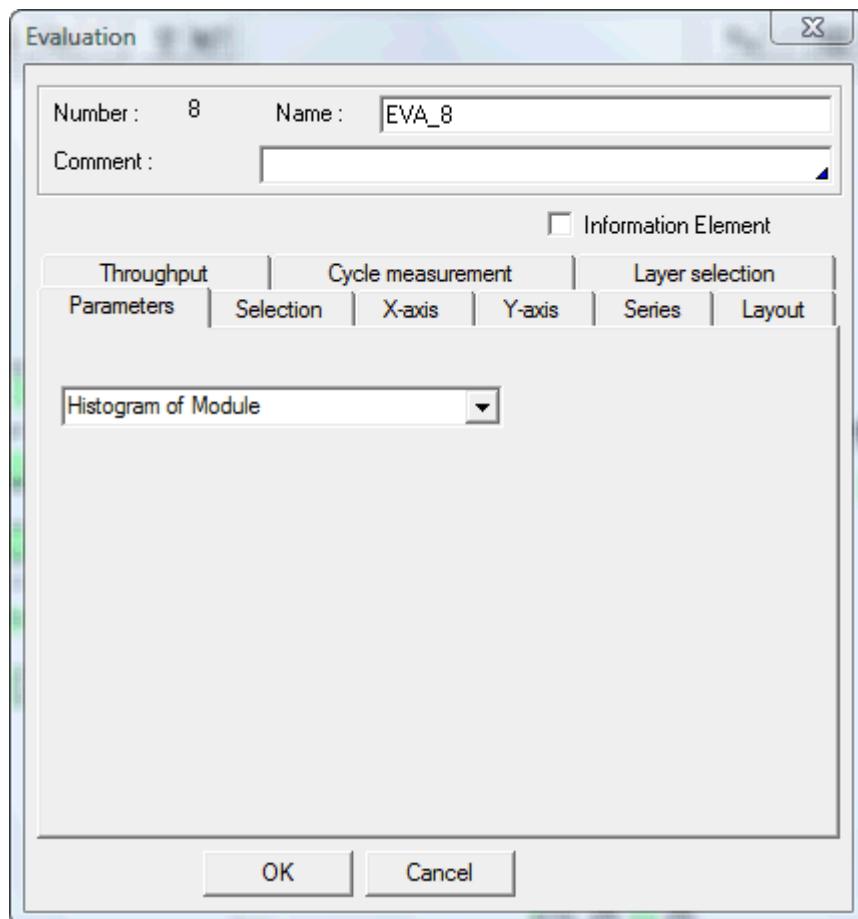


Figure 8.27: Parameters of evaluation

After the execution of the evaluations, all connected elements are selected and the selected result diagram is displayed. This takes place by the context menu (right mouse button) or with double-click on the element with pressed SHIFT key. The sub-dialog selection, series, x-axis and y-axis correspond to those of the result parameters. When executing the evaluation, the parameters will be transferred. Further information is to be inferred from the chapter [result parameters](#).

The evaluation module is marked as inconsistent, if the display parameters do not fit the simulation parameters. This can be caused, for example, by change of the pre-run time or the statistics interval. The display parameters are to be adapted then accordingly. The simulation of the model is, however, further possible.

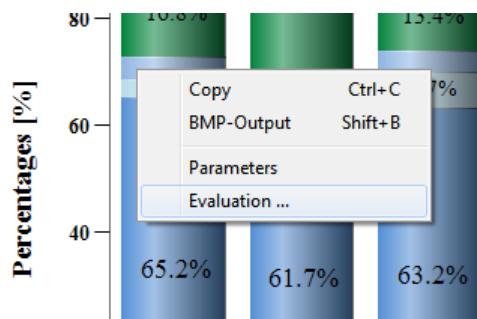


Figure 8.28: Definition with the context menu

The evaluation module can be defined also from the context menu of the results display. Exactly one evaluation module is to be selected besides the elements which are displayed. When the entry **Evaluation...** is selected from the context menu, the parameters of the representation are transferred to the selected evaluation.

The evaluation can be executed also by the COM interface. Further information is to be inferred from the chapter [COM-Server](#).



9 Text Editing / Drawing Polygons

These elements have been eliminated from the modules palette. Please use text and polygon from the graphics palette





10 Simulation Run

The menu **Simulation** contains the following submenus:

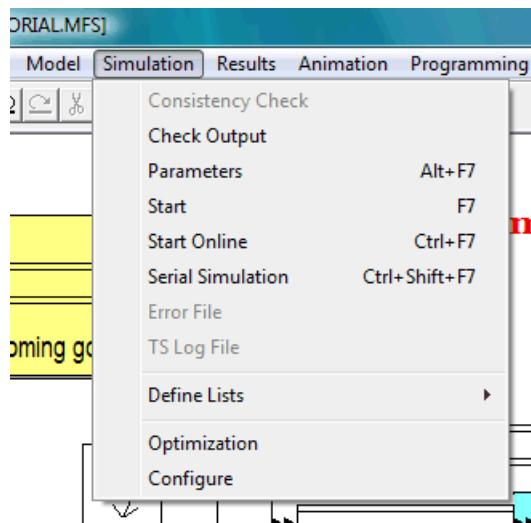


Figure 10.1: Submenus „Simulation“

The **Consistency check** checks whether all parameters as well as linking of modules are correct. This inspection takes place usually after every change of the model parameters. It can be started in the case of an inconsistent model here still once explicitly. For the case that in the MFS not all specifications have been made, DOSIMIS-3 announces *system contains errors*, otherwise the message *system is error-free* appears. All inconsistent modules are selected after an explicit consistency check. These can be changed afterwards by using the menu *Edit/Parameter*. If the model is not consistent, the option *Simulation|Start* is disabled.

Check **output** displays the error file (*.chk), so that the user is able to see, which errors are listed in the check file.

For defining the simulation **parameters** (submenu Parameter) the user must enter the simulation time, preparation time and the protocol interval.

Shortcut

Symbol:	
Toolbar:	<i>Modeling</i>
Keyboard:	Alt + F7

Start processes the simulation run. For more details see chapter [Start of Simulation](#).**Shortcut**

Symbol:	
Toolbar:	<i>Modeling</i>
Keyboard:	F7



Start Online processes the simulation run in online mode. For more details, see below.

Shortcut

Symbol:



Toolbar:

Modeling

Keyboard:

Ctrl + F7

With **Serial simulation**, offline simulation is started for each opened model.

Shortcut

Keyboard:

Ctrl + Shift + F7

With **Error file** and **TS-Log file** the corresponding files (Extension *.err* or *.log*) can be reviewed after the simulation run.

With the help of **optimization**, series of simulations to automatically optimize the model can be defined.

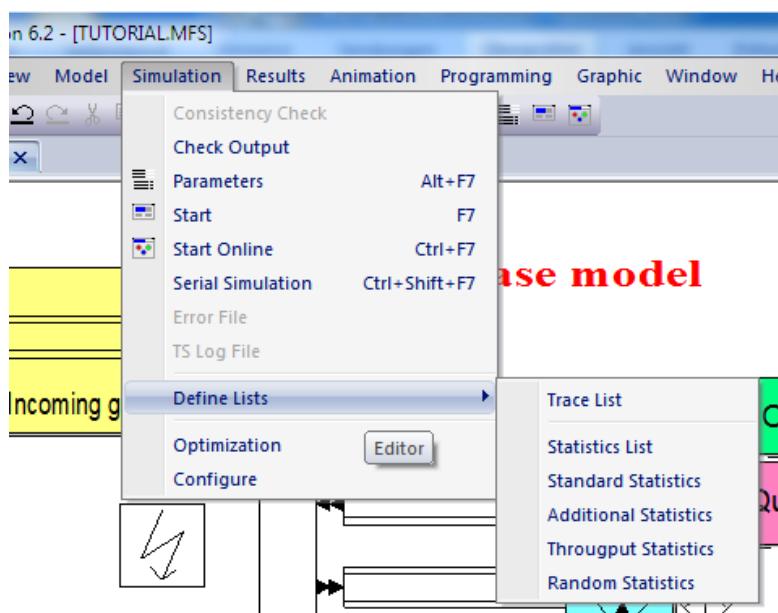


Figure 10.2: Submenu „Define lists“

Define **Lists** takes the currently selected modules into the statistics list, trace list or additional statistics. By default, all possible modules are taken over. If this action is performed without any module selected, the corresponding list will be empty afterwards. Selecting only the modules under consideration reduces the time of the simulation run and the size of the statistics and trace files. Selecting one of these menus without any selection means that the corresponding statistics will not be made for any element. For selecting the old list use **Model>Select/Statistics List**.

With the help of **Optimization**, a series of simulations can be defined, from which an optimization of the current model can be made automatically and in a destination-orientated manner.

The option **Configure** opens the property page of the simulation.



10.1 Simulation Parameters

When defining the simulation parameters, the user must enter the simulation time, pre-run time and the statistics interval. These three times have the following meaning:

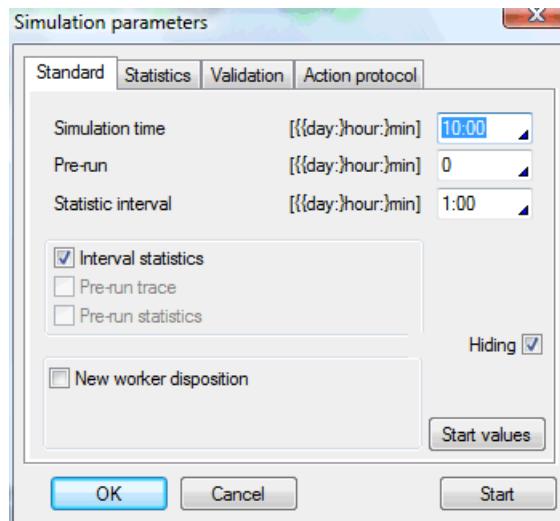


Figure 10.3: Simulation parameter standard

The simulation begins at time 0. The system is empty except for possible buffer initializing. The duration of the simulation in minutes is given by **simulation time**. The **pre-run** time is part of the simulation time and starts at time 0 too. The pre-run time serves to illustrate the transient phase of the system. It is thereby prevented that the values are falsified by transient behavior at the beginning of simulation. Finally, the **statistics interval** states the time intervals, in which interim statistics are to be made after the preparatory phase. In these interim statistics, the data are logged at the real simulation time. The inter-relationship can be seen in the following figure.

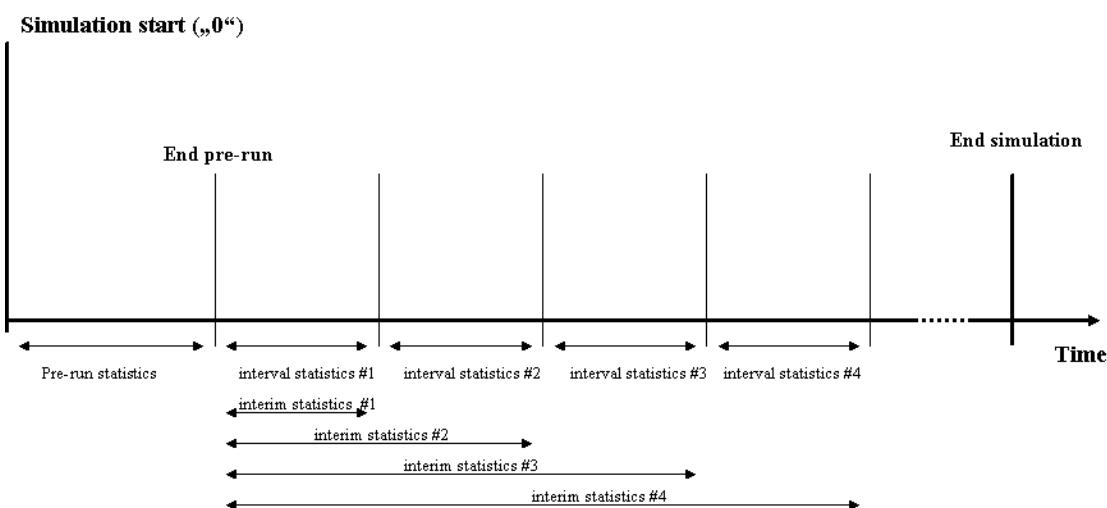


Figure 10.4: Statistics times

This statistics can also be requested for the preparation time by switching on the corresponding switch. The modules, which are included into statistics and animation are defined in the corresponding lists (see submenu **Simulation/Define lists**).



If **Interval statistics** is selected, then interval statistics, only considering the intervals, are computed in addition the interim statistics.

If a time for pre-run is given, then the user can log the statistics by **With pre-run statistics** and **With pre-run trace** during the preparatory time and record the action protocol. Otherwise both start first from the preparatory moment. The recording of the statistics always occurs at the preparatory moment and from there on always in the statistic interval.

With **New worker disposition**, an optimized scheduling algorithm can be activated. For reasons of compatibility the old disposition algorithm keeps on being available.

By **Hiding** it can be selected whether the user interface is visible during simulation or not. During the simulation the interface does not react to any action.

The initial value for random number can be changed with the button **Start values**.

To begin a simulation, the button **Start** is to be pressed. Alternatively, the menu **Simulation/Start** can be used. In the top left-hand area of the screen, a window is opened showing the real simulation time. The simulation can be interrupted by clicking the **Cancel** button. The results which had been calculated up to the moment of interruption are stored in the protocol and can be accessed.

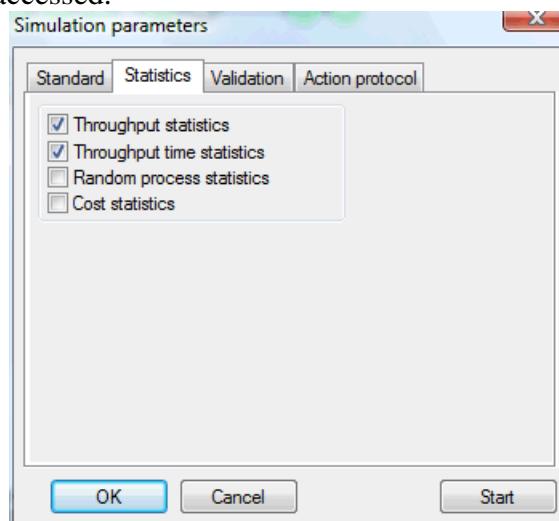


Figure 10.5: Simulation parameter statistics

With the switches **Throughput statistics** and **Throughput-time statistics**, the corresponding statistics can be selected for the entire model. The throughput statistics for a module are recorded only if this module is in the statistics list and if **With throughput statistics** is activated.

By the entries **Random process statistics** and/or **Cost statistics**, the corresponding statistics can be enabled or disabled for the whole model.

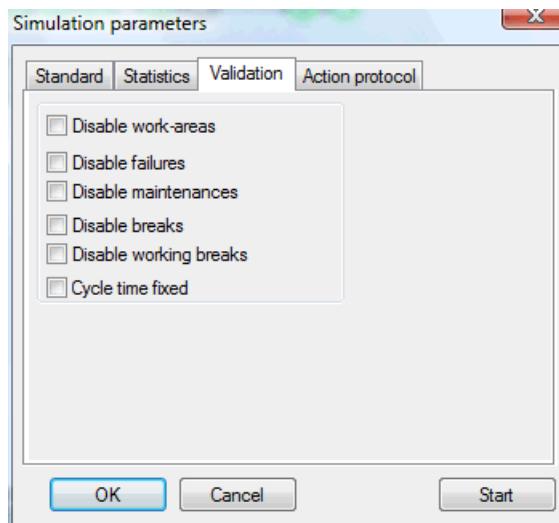


Figure 10.6: Simulation parameter validation

If **Disable work areas** is selected, then the manual activities are automatically executed, when the maximum number of workers is available.

With **Disable failures** all failures of the model are passivated. All **maintenances**, **breaks** and **working breaks** can be passivated in the same manner. If the status *indifferent* is selected here, then this is interpreted as test attitude.

With **Cycle time fixed**, all random processes in the simulation will work with the corresponding mean value.

These points are used basically for the validation of the model. If one of the two points is activated or the work areas are deactivated, this is marked in the simulation by the color of the progress bar.

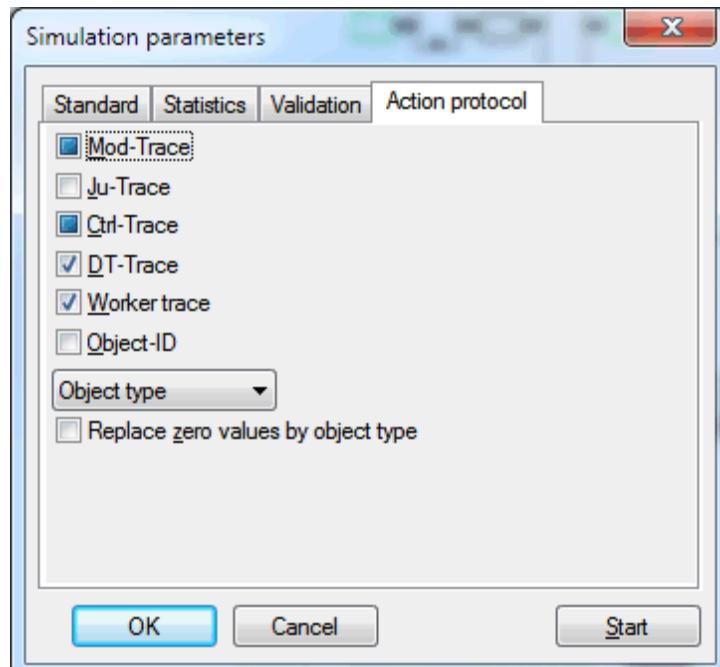


Figure 10.7: Simulation parameter action protocol



The **El-trace** switch has three positions. If it is activated, the action protocol is carried out for all components. If it is passivated, the action protocol is not carried out for any element. In this case, no result graphics based on the trace file are available. In the status *indifferent*, the action record is written according to the trace list.

With activated **Ju-trace**, the transfers of the objects at the junctions are recorded more exactly. These are highlighted then during the animation in color.

Über den Schalter **Steuerung-Protokoll** kann das Aktionsprotokoll für Steuerungen (z.B. Bereichskontrolle) beeinflusst werden.

If **DT-Trace** is not selected, then outputs (both the text- and anitext output and the debug output of individual decision tables) of decision table are suppressed.

The switch **Worker trace** enables recording of an action protocol for all work areas so that the behavior of the workers can be animated as well.

When the switch **Object ID** is switched on, with each action recorded in the task protocol, a unique ID is placed, from which the object can be identified.

The switch has three positions. If it is activated, action protocol is made for all workers. If it is passivated, action protocol is disabled for all work areas. In this case also, no result diagrams based on action protocol are available. In the status *indifferent*, action protocols are written in accordance with trace list.

Instead of the object type, a different object attribute can be used. In the selection box it is possible, to change the information, which is used as output, from object type to the destination parameter or each integer attribute.

10.2 Random Numbers

All random processes in DOSIMIS-3 have an own random seed value to create a reproducible order of random numbers. In order to initialize this initial value, every material flow system has a random number which is used for this. This number is changed after each generation of a new random process, so that each process gets another initial value. A module can have several random processes. The workstation has at least one random process for the determination of work time and a further one for the production of the new object type.

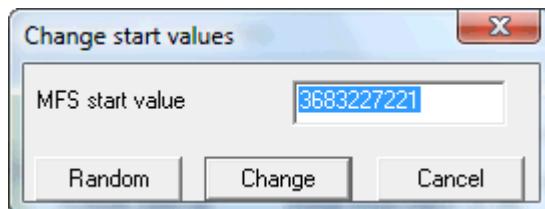


Figure 10.8: Random number start value

The user can enter the initial value directly. This can also be initiated again by the system with the button **Random**. If the dialog is exited via **Change**, all random processes are re-initialized with the shown initial value. While exiting the dialog via **Cancel**, the entered initial value is taken over, so that all new random processes are initialized with this initial value.



10.3 Start of Simulation

The preceding chapters can be seen as a preparation for the main task of DOSIMIS-3, simulation of the dynamic behavior of a material flow system, especially determining concrete results concerning the behavior of the system under certain conditions. These results can be presented to the user in three different ways; as statistics in a tabular format with simulation data, as graphics showing certain data and by using the animation program as a presentation of object movements in the system.

Before starting a simulation run, the user must carry out a consistency check and define the simulation parameters. After selecting **Simulation** in the main menu, the system can be checked for complete parameterization via the item **Consistency check**. The result of the check is written in the status bar. Logical errors are not checked. They may still occur during the simulation.

The simulation can only be started if the consistency-check result is OK and all simulation parameters are set.

After selecting **Simulation/Start** the simulation dialog appears. The name of the simulated material flow system is in the title. The blue bar shows the simulation progress, which is also indicated as time. Messages that may occur during the simulation appear ahead of the blue bar. These can refer, for example, to errors (errors in the simulation). In this special case further information is to be inferred from the *.slg-file or the *.err-file.

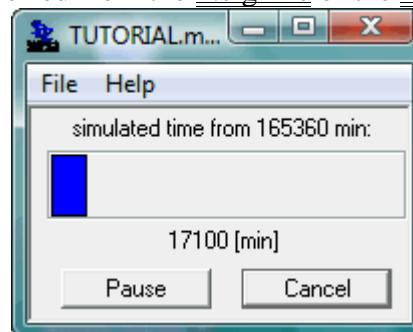


Figure 10.9: Simulation dialog

With the left button (Pause) the simulation can be interrupted. Until the next click on this button (Continue) no calculation will be done and the computer performance is available for other programs.

If a current simulation should be interrupted, this is achieved by pressing the button **Cancel**.

The progress bar will be drawn in green during the test mode. A test mode is one if:

- Disturbance, tracing, maintenance work breaks or work areas is switched off. (switch condition indifferent)
- An element is passivated (switch condition indifferent).
- *Cycle time fixed* is active
- A parameter of an element is in the test mode

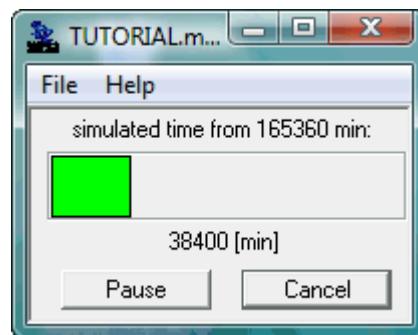


Figure 10.10: Simulation dialog (test mode)

The submenu **Help** shows a dialog that supplies further information on the calculation program.



Figure 10.11: Information about the simulator

As already mentioned, two data files can be set up during a simulation, ***.slg** and ***.tra**. The ***.tra** file contains data of the movement protocol which is necessary for the setting up of the occupancy diagram and for animation, whereas the ***.slg** file lists certain results in tabular form.



10.4 Online Simulation

The simulation of DOSIMIS-3 is usually conducted in an offline mode. The calculation takes place in an external program. That means that all information which is calculated during the simulation is exchanged by a file with the user interface. Since this is not often sufficient during the validation, an online mode is available when required. This runs directly under control of the user interface. In this debug mode all data of the simulation can be accessed.

At start of the online simulation, the animation is started automatically. All actions are animated as far as possible. The operation can occur with the aid of the animation bar. Here the following functions are enabled: *Stop*, *Pause*, *Fast* and *Single step*. A special feature is that the drawing operations can be disabled. This happens with the button *Hide*.

Attention: With the start button of the animation bar, the animation is activated. The start button of the online simulation is in the modeling bar.

The online mode is marked by a blue frame in the animation bar. Further ahead, the blue simulation bar runs at the top edge of the work area.

With activated **Digital display** the simulation time is displayed. In the other case, this occurs in the right area of the status bar.

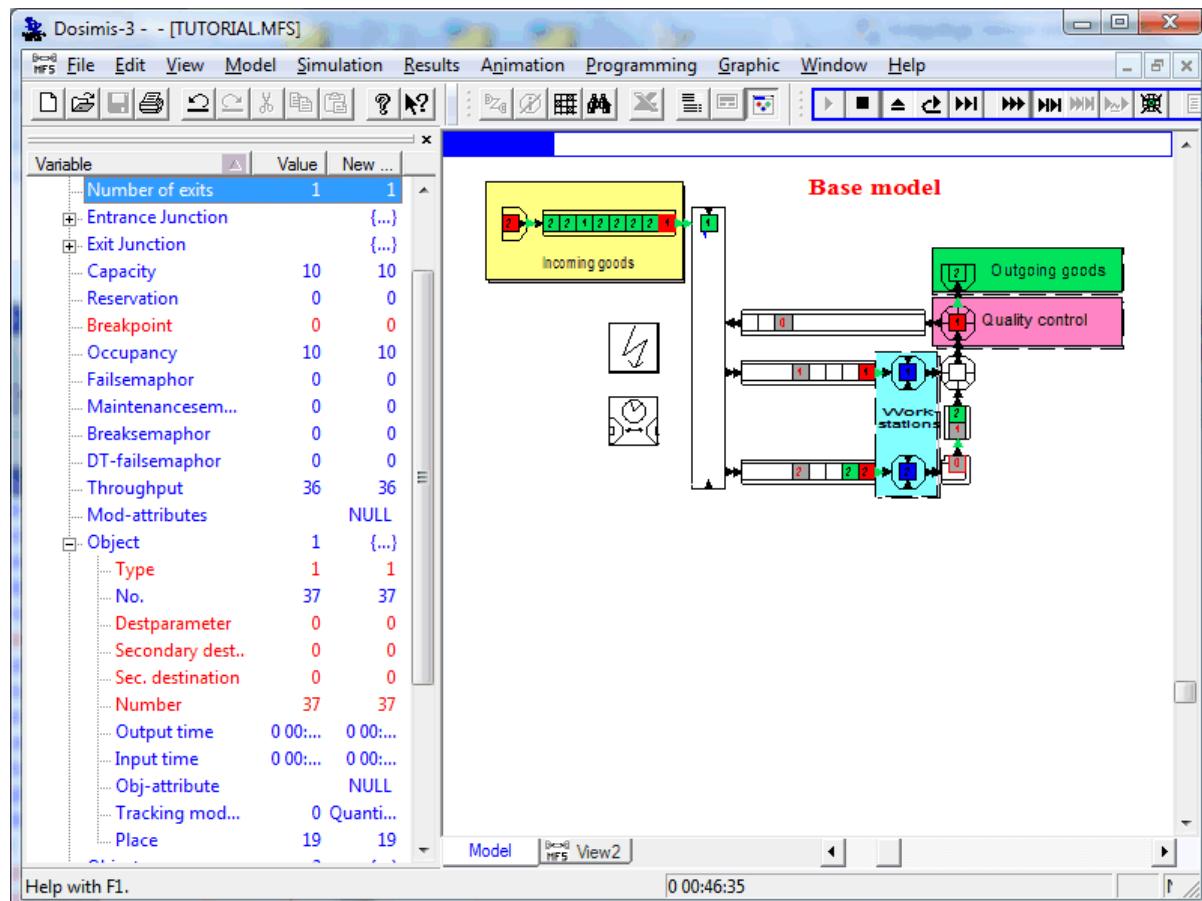


Figure 10.12: Online Simulation



In the variable window, the parameters of the elements can be seen. In the digital display, the simulation time is shown. Both windows can be activated when by clicking the action bar with the right mouse button and selecting the corresponding windows.

The current simulation can be suspended by pressing of the spacebar. This can be achieved also through the definition of breakpoints. This happens via the context menu of the component.

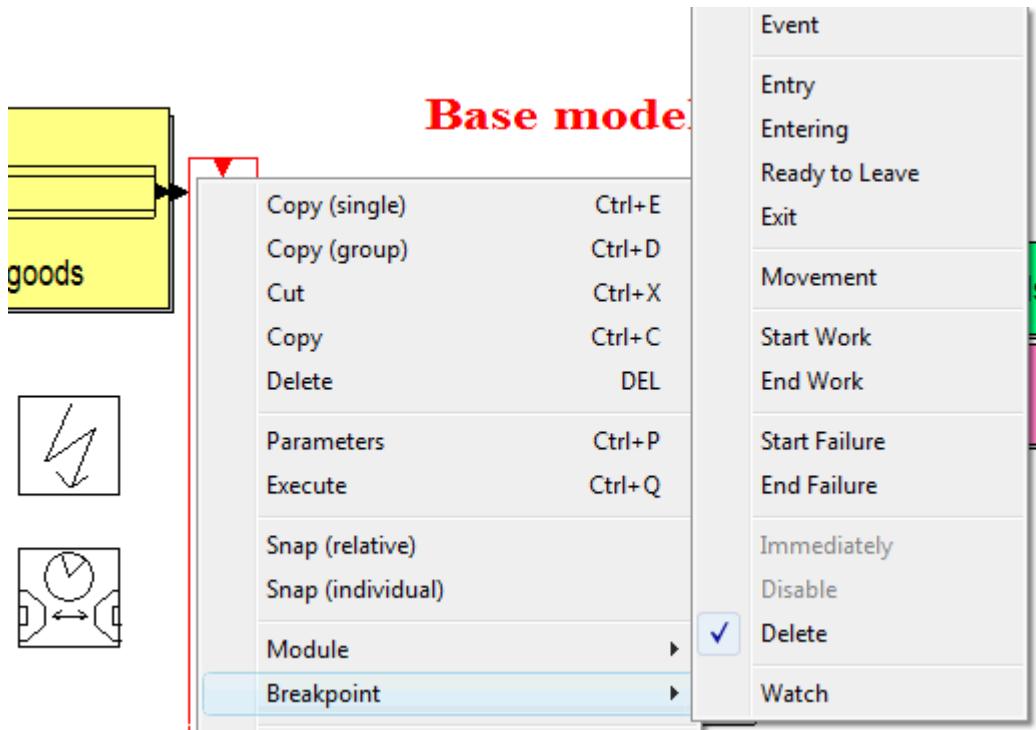


Figure 10.13: Breakpoints

Different event types are available for selection. If **Event** was chosen, the simulation is stopped at every event of the element. The definition of a breakpoint is only possible if exactly one element is selected. Since individual events often lie within an order of events and the simulation cannot be suspended beforehand according to such an order, all elements in the variable window that are shared at such an order are indicated. This is, for example, an object transfer since the exit from an element directly forces the entry into the succession element. If both events have occurred, the simulation is suspended. The abbreviations for these events are shown at the beginning of the display. They correspond to the abbreviations in the trace file. Components, for which a breakpoint is defined, are marked at the upper left corner with colored squares.

If the simulation is intermittent, every component can be indicated via the context menu item **Watch** in the variable window.



If nevertheless the simulation is supposed to be suspended within such a mandatory order, the switch is in addition **immediately** activated. The simulation is stopped then and the parameters of the activating component can be seen in a dialog.

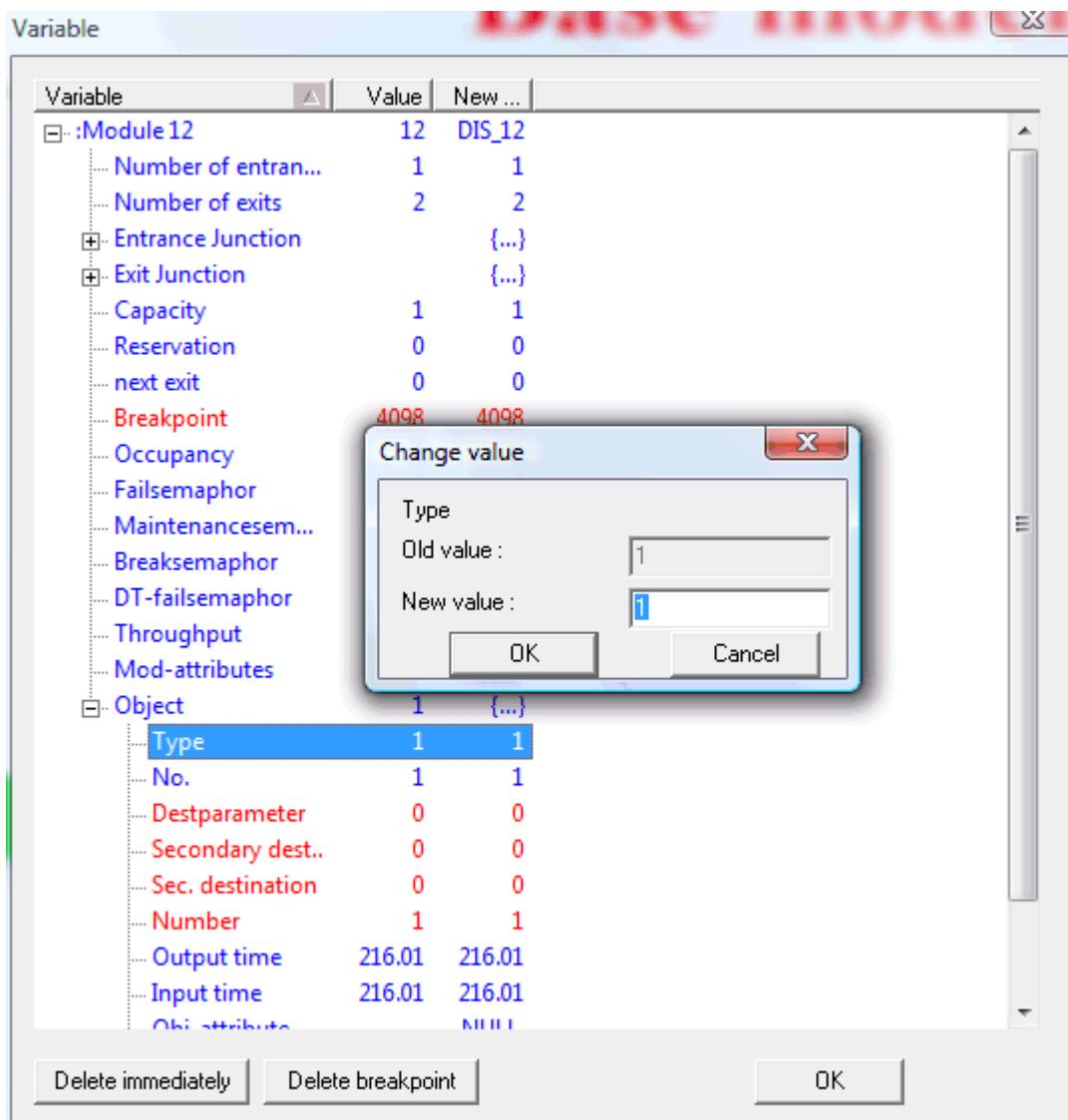


Figure 10.14: Variables during simulation

The values marked in blue are for information. The values in red can be changed. This happens by double-clicking the corresponding line. After the change, the new value is immediately fitted into the calculations of the simulator.

Active breakpoints can be **disabled**. The module is marked further on, but animation will not stop when reaching the selected event

With the button **Delete immediately**, the switch can be immediately deactivated. The simulation is then suspended at the end of the event sequence. If this is not wanted either, the breakpoint can also completely be removed.

The methods of the breakpoints are available with Global Controls and are connected to decision tables. For further information, see *decision tables*.



10.5 Optimization

Via a sub-menu in **Simulation**, the optimization module can be activated by the corresponding menu item. Several optimizations can be parameterized for a model.

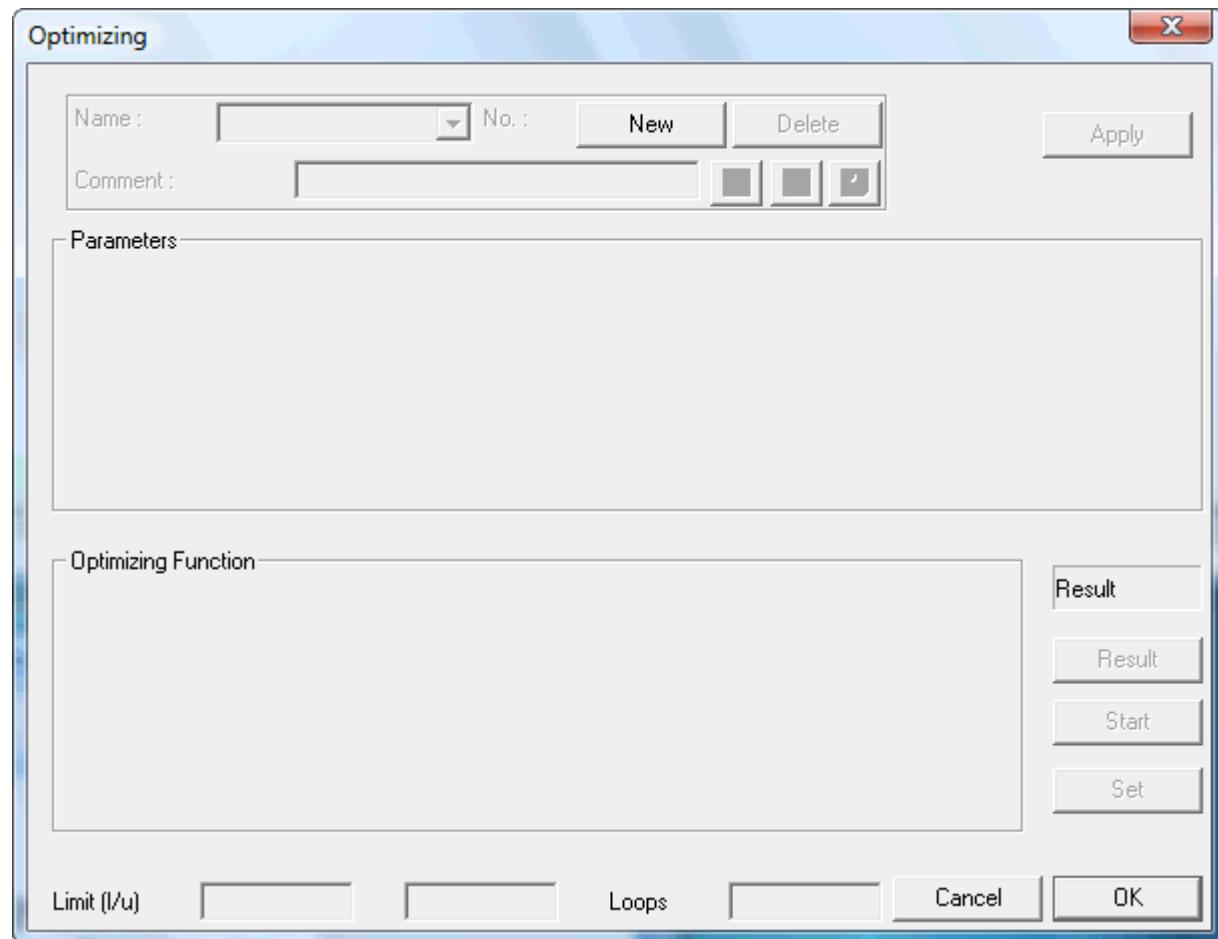


Figure 10.15: Empty optimizing dialog



After the selection, an empty dialog will appear.

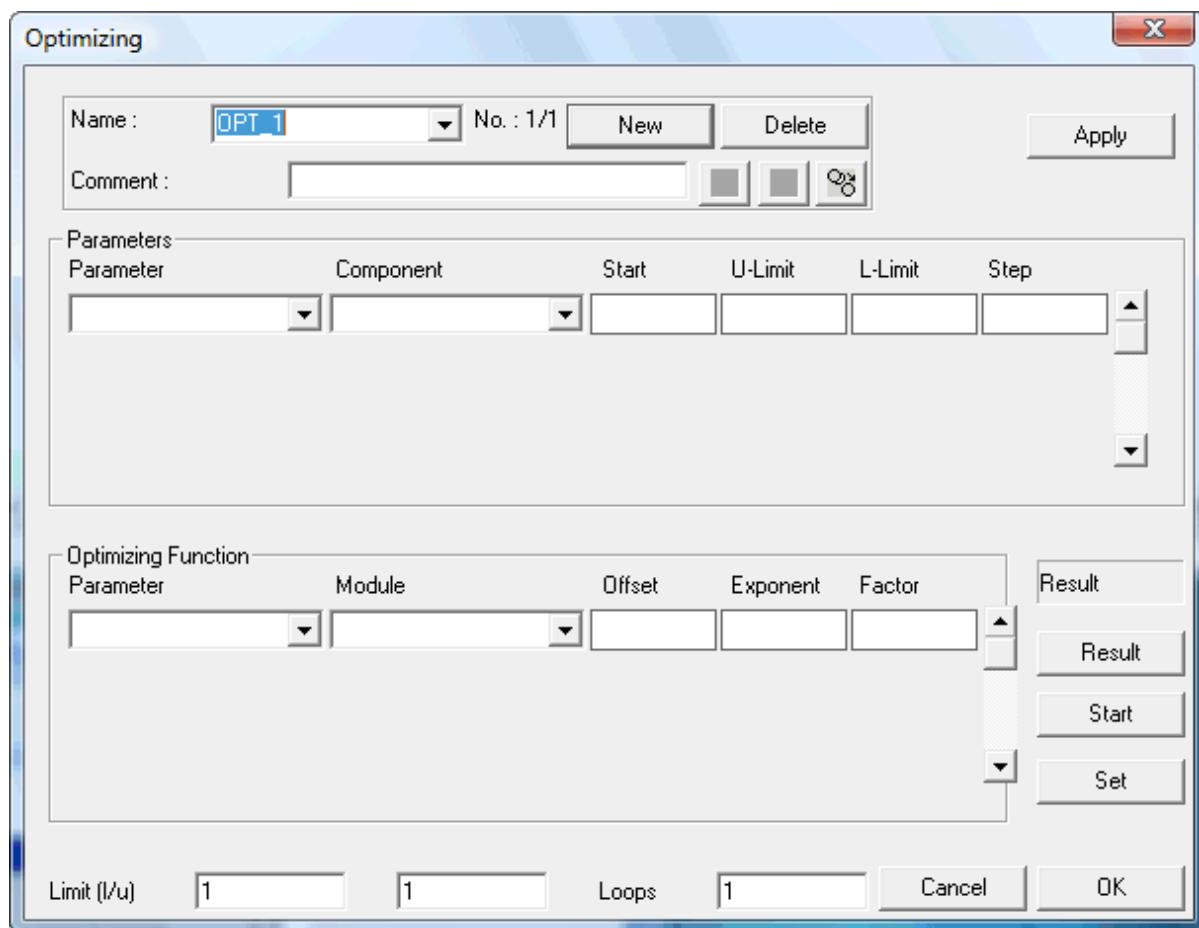


Figure 10.16: Default – Optimizing dialog

After the selection of **New**, the fields in the parameter mask and optimizing functions are ready for input. This optimization can be assigned a name different from what is shown by default.

An optimization is defined basically by two components. Firstly, the parameters which are supposed to be varied and secondly, the optimizing function, which an optimum is supposed to achieve, are to be defined.

To access modules, these must be announced with the help of the button **Overtake**. After this, the selected modules are prepared for selection.



10.5.1 Parameters

Parameter	Component	Start	U-Limit	L-Limit	Step
Capacity	ACC_N_1	1	10	8	1
Capacity	ACC_N_2	1	10	8	1

Figure 10.17: Parameters

With the parameters element, characteristics can be varied. All integer and float constants are available. With these, all parameters of the elements can be varied. For compatibility, the capacity and the initialization can be accessed directly. These are initialized in this case at the beginning of the optimization with the initial value. This happens independently of the current value in the simulation model. The optimization now changes this value by the increment in every iteration step until the upper limit is reached. The lower limit is used in the second optimization step, in which the parameter value is reduced by the increment from the current value in every iteration step until the lower limit is reached.

10.5.2 Objective Function

Optimizing Function	Module	Offset	Factor
~Throughput	SINK	0	1

Figure 10.18: Objective function

The objective function is defined similarly. Statistical module properties are available here as well, which are taken from the statistics file after completion of the simulation. To put different results in a comparison, the individual parameters can be weighted with the help of the offset and the factor. First, the offset is added to the parameter value and then multiplied with the factor. The optimizing function turns out then as the sum of the individual target values.

$$\sum_{i=1}^n (off_i + value_i) * fac_i$$

10.5.3 Target Value

Limit (l/u)	37000	38000	Loops	10
-------------	-------	-------	-------	----

Figure 10.19: Target values



The parameters in the lower line are used now to delimit the number of simulation runs. In the first step, the simulation runs until the **upper limit** (medial field) is reached. In the second step the parameters are reduced until the **lower limit** is achieved. However there only so many **loops** that are gone through, as indicated in the right field.

10.5.4 Optimization Run



Figure 10.20: Optimization run

The optimization operation is started by **Start**. After termination, the reached target value stands in the field above the result button. A detailed statistic of the optimization operation is viewable by **Result** button. After termination of the dialog, all parameters of the model are reset. When the button **Set** is pressed, the parameters are taken over into the model.

CAUTION! The saved *.mfs file has the values of the last optimization operation at this time. The user is asked after completion of the optimization, whether the model is to be saved.



10.5.5 Algorithm

Strictly speaking, the implemented algorithm works through an all-possibilities test. Every iterative loop exists from just as many steps, as there are defined parameters. In every step, one parameter is changed; the target value resulting in this case is noted and then reset again. After one loop, the parameter that induces the largest (smallest in the second step) change in the objective function is taken.

If the upper limit of a parameter is reached after a number of steps, this remains constant and is not varied in the subsequent loops anymore.

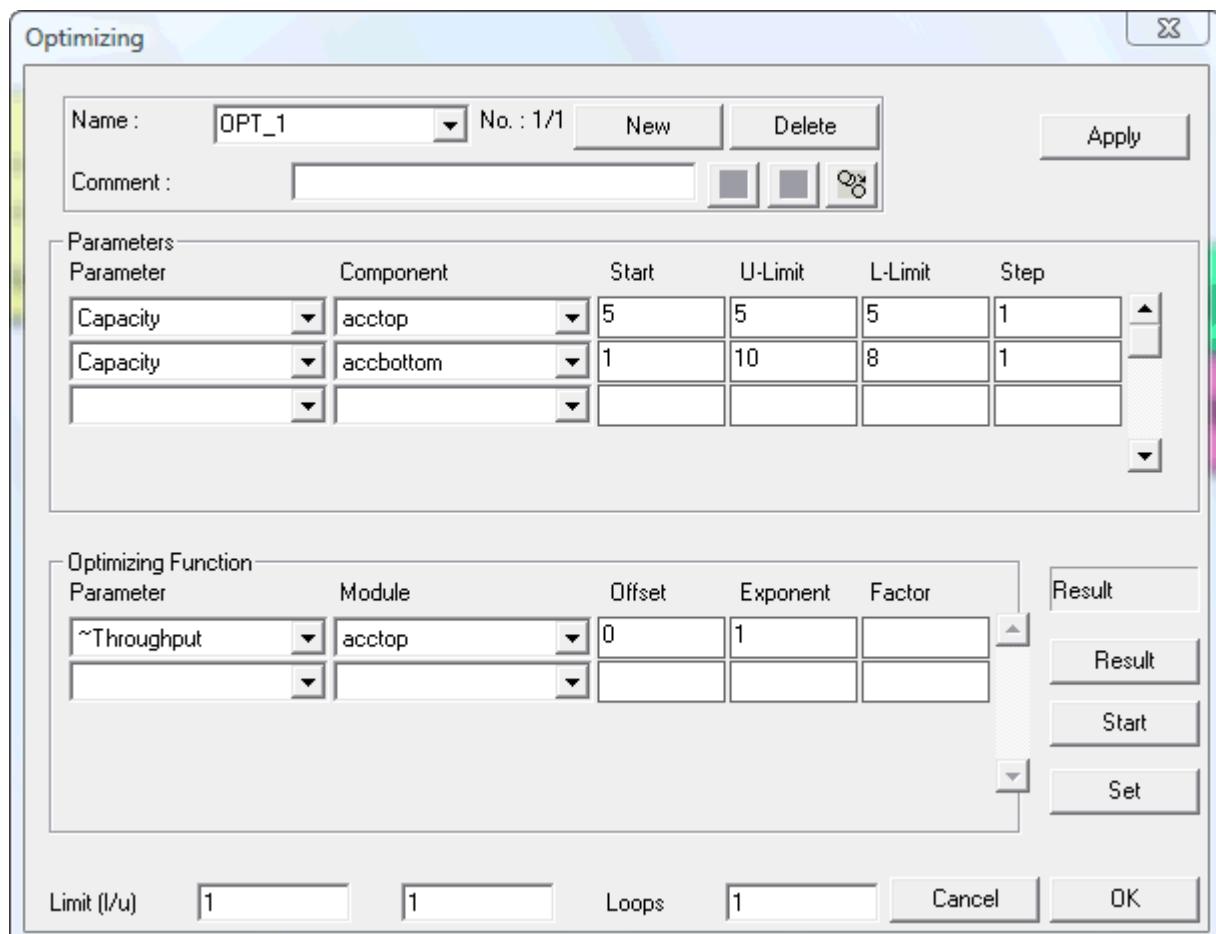


Figure 10.21: Example of dialog

The optimization consists of two flows. In the first, the upper limit is approximated by the objective function from below, and in the second, the lower one from the top. The basic idea is that the parameters with the greatest effect first ensure that the top limit is achieved, and that in the second step, the values which do not show almost any effect are reduced. This procedure becomes clear if the user performs buffer dimensioning for several buffers and wants to optimize the throughput. First, a few buffer positions are added until the top limit is reached, and then as many as possible buffer positions are deleted again until the lower limit is achieved.



10.5.6 Result Graphic

With the help of optimization diagrams, the different experiment loops can be compared. For each optimization flow, the target value and the parameter, which causes the largest modification of the optimizing function, are represented. The results of the two optimization flows are colored separately.



Figure 10.22: Optimization

10.5.7 Result Table

In the first lines, the parameters of the optimization are repeated once again; first, the parameters to be varied and then the parameters of the optimization loops.

```
KAP SST_N_1           1.00 10.00  8.00  1.00    1   10
KAP SST_N_2           1.00 10.00  8.00  1.00    1   10
Start 10 29219.00     38000   37000
```

In the second section, the results of the loop are shown. First the number of the loop, then whether the first or second optimization step is carried out. Behind that are for every step, the value of the objective function and the change of the objective function compared with the last loop. If all parameters were iterated, the specification of the maximum target value as well as the component and the value of the parameter which led to this target value are shown. The line is concluded with date and time of the loop.



```

LOOP 0 -> 30145.0/ 926.0 31800.0/2581.0 _ 31800.00 - MAX 3.00 SST_N_2 Wed Jul 19 13:22:15 2000
LOOP 1 -> 33607.0/3462.0 34224.0/2424.0 _ 34224.00 - MAX 5.00 SST_N_2 Wed Jul 19 13:23:10 2000
LOOP 2 -> 35366.0/1759.0 35955.0/1731.0 _ 35955.00 - MAX 7.00 SST_N_2 Wed Jul 19 13:24:04 2000
LOOP 3 -> 36676.0/1310.0 36953.0/ 998.0 _ 36953.00 - MAX 9.00 SST_N_2 Wed Jul 19 13:24:58 2000
LOOP 4 -> 37554.0/ 878.0 37779.0/ 826.0 _ 37779.00 - MAX 10.00 SST_N_2 Wed Jul 19 13:25:54 2000
LOOP 5 -> 38111.0/ 557.0 37779.0/-999.0 _ 38111.00 - MAX 8.00 SST_N_1 Wed Jul 19 13:26:21 2000

```

The statistic is similarly made for the second step.

```

KAP SST_N_1 1.00 10.00 8.00 1.00 8 10
KAP SST_N_2 1.00 10.00 8.00 1.00 10 10
LOOP 0 <- 38111.0/-999.0 38080.0/ 301.0 _ 38080.00 - MAX 9.00 SST_N_2 Wed Jul 19 13:26:49 2000
LOOP 1 <- 38111.0/-999.0 37833.0/-247.0 _ 37833.00 - MAX 8.00 SST_N_2 Wed Jul 19 13:27:17 2000
LOOP 2 <- 38111.0/-999.0 37833.0/-999.0 _ 0.00 - LIMIT FUER ALLE PARAMETER ERREICHT Wed Jul 19
13:27:17 2000

```

At the end of the file the values of the parameter that led to the reaching of the lower limit are shown.

```

KAP SST_N_1 1.00 10.00 8.00 1.00 8 10
KAP SST_N_2 1.00 10.00 8.00 1.00 8 10

```

The results can be displayed in a result diagram. This is provided by the sub-menu Optimization of the menu results.



11 Simulation Results / Output

As already mentioned at the beginning of the previous chapter, at the end of a simulation run, the user has various options of presenting and evaluating the results. The following alternatives are provided from the menu **Result**:

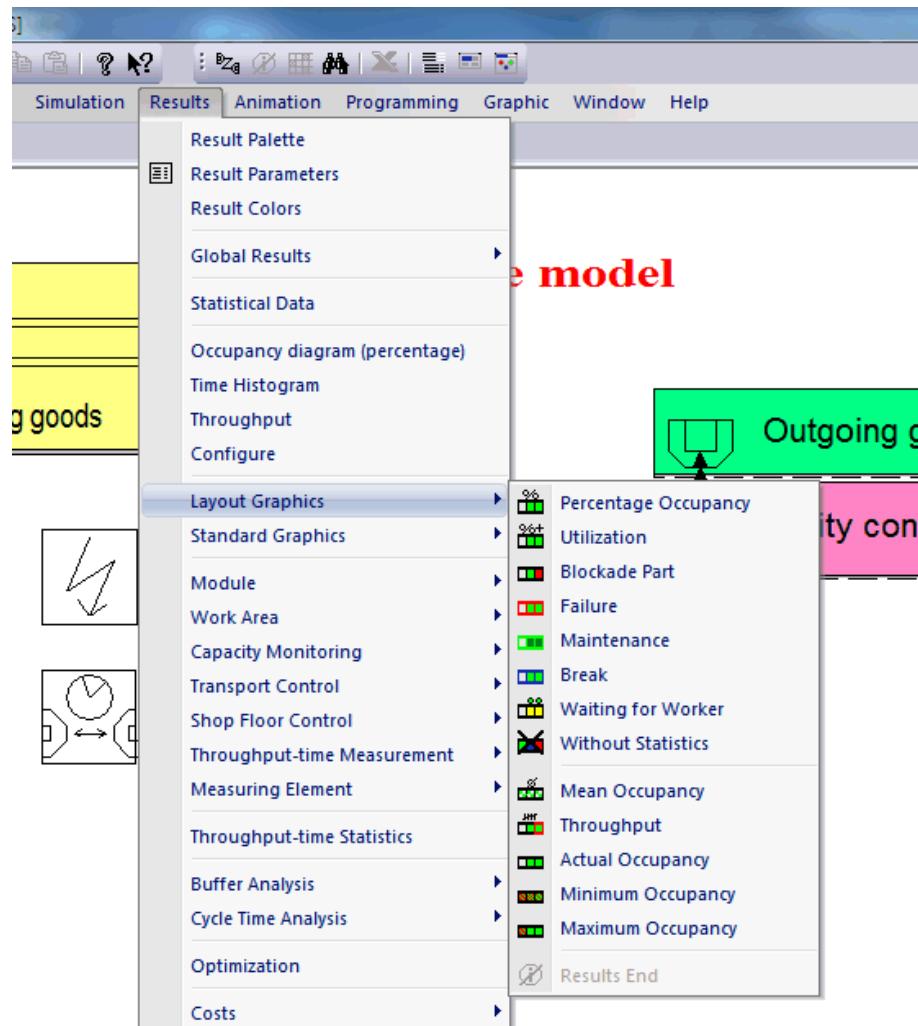


Figure 11.1: Menu „Results“

Several possibilities are available to analyze the results. **Statistical data** opens the statistics file. As these statistics data are already large volumes of data in the case of medium-size models, the further possibility exists if filtering these data by graphic tools. This occurs by using of **module graphics art** directly in the layout or in continuous or discrete-time diagrams, which are described further below. Below the entry statistics data are the top three of the result representation. These can be configured in the properties of the result representation.

All diagrams can be configured with the aid of **Result parameters** and **Results colors**. Results, which are represented directly in the layout, can be called directly with the result palette.



11.1.1 Comparing Results Graphics

If several models are opened, the selected modules of the different models can be represented in the same diagram. First, the result diagram is to be selected. Then this diagram has to be expanded to all models with the help of the menu option **Comparison (global)**. The global result representation possesses an own dialog for **Result parameters (global)** and **Result colors (global)**.

For the global-result comparisons of *time-discrete* diagrams the models should be parameterized with same statistics times. The displayed times are determined by the firstly opened model.

There are global result comparisons for all kinds of allocation diagrams, component histograms, time histograms, work area statistics, transportation statistics and throughput statistics.

11.2 Result Parameters

For better and compact input of the result parameters, the data are to be defined in grouped sub-dialogs.

11.2.1 Selection

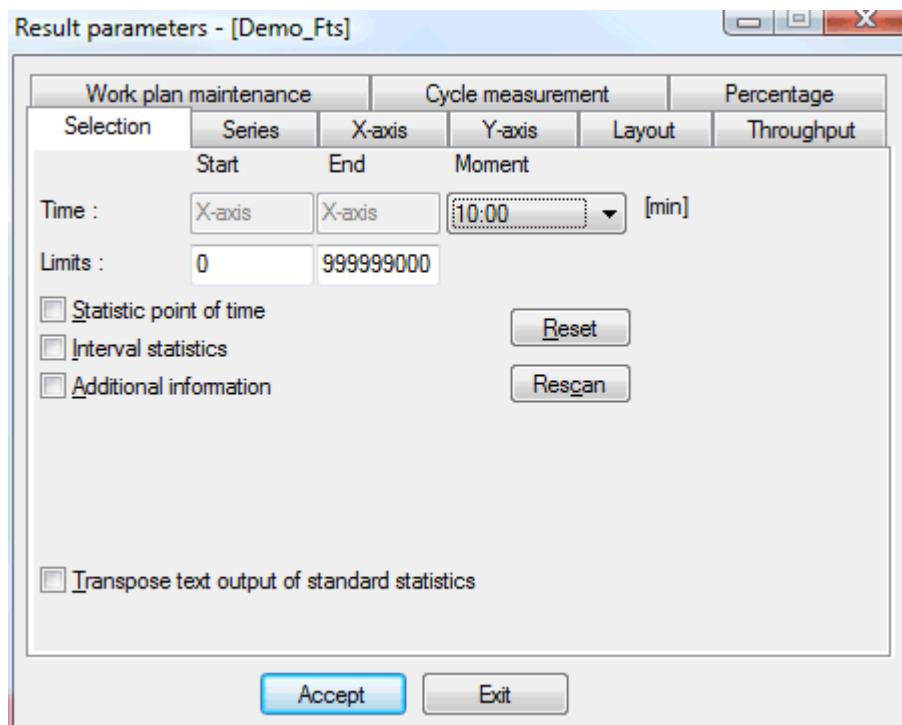


Figure 11.2: Result parameters of Selection

Time	Delimitation of the display period with diagrams is made by the parameters of the x-axis
Time Interval	Limitation of the period of time for charts which represent information over several points of time, e.g. occupancy diagram or time diagram. To view data of only one point of time, the limits should include this interval. Alternative Statistic point of



	time can be selected.
Moment of View	Determination of the time moment for diagrams which only represent data of a moment in time (e.g. module histogram or Point of time activated).).
Limits	Limitation of the displayed results. For example the object types in throughput statistics or occupancy diagram. Here it is to be noted that changes of the type of object between entrance into the module and exit from the module cannot be recognized.
Statistic Point of Time	The display occurs only for the display time, not for the entire interval.
Interval statistics	Selection of displayed statistic data (interval statistics, if the switch is set, interim statistics otherwise).
Additional Information	Draw additional information in the diagram, e.g. gridlines and marking of pre-run.
Rescan	Repeat reading of the statistical data (e.g. after an external start of simulation).
Reset	Calculate point of time for start and end of simulation parameters.

11.2.2 Series

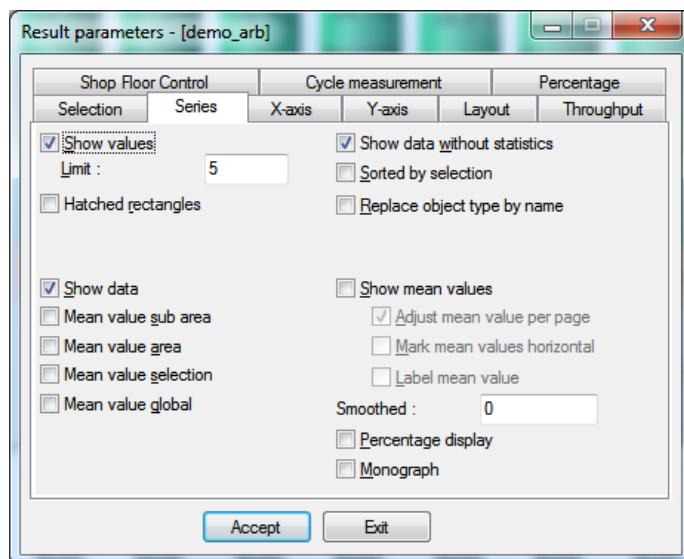


Figure 11.3: Result parameters of Series

Show values	The values, which are the basis for the display, are displayed in the diagram.
Limit	If <i>Show values</i> is active, only values greater than <i>Limit</i> are shown.
Hatched rectangles	The rectangles are hatched. Thus, the color boundaries can be better detected in black-and-white printouts. Since the rectangles with same or similar colors possess different hatches, in case of doubts, the allocation of color to statistical values can be made better.
Show data without statistics	Data values, which do not contain any statistic entry during the whole statistic interval, can be excluded from drawing.
Replace object type by	At places where object types are used, these will be replaced by



name	the name of the object. The name results in the first name found in the object type constants which agree with the object type.
Sorted by selection	The sequence of elements is according to their selection, not by number.
Show data	Indicates the individual statistics data. If only the average value is to be indicated, this switch must be deactivated.
Show mean values	With some statistics (time histogram, work area statistics) an average value can additionally be displayed. This results from the statistics of the selected elements at a point of time.
Mean value	With some time-discrete statistics (time histogram, work area statistics) an average value can additionally be indicated.
	Sub area
	Area
	Selection
	Global
Show mean values	With some continuous diagrams (allocation diagram) the average value of the represented data can be shown in the right area.
Adjust mean values per page	The average value refers to the visible interval, not to the entire simulation period.
Smoothed	Smoothing of the occupancy diagrams. The simulation interval is split into the pre-set number of intervals and the mean value of the individual periods of time is displayed.
Percentage representation	The display of the allocation diagram is not made according to the absolute allocation but relative to the capacity of the element.
Monograph	The picture of an occupancy diagram contains the occupancy for each object type in the module.



11.2.3 X-Axis

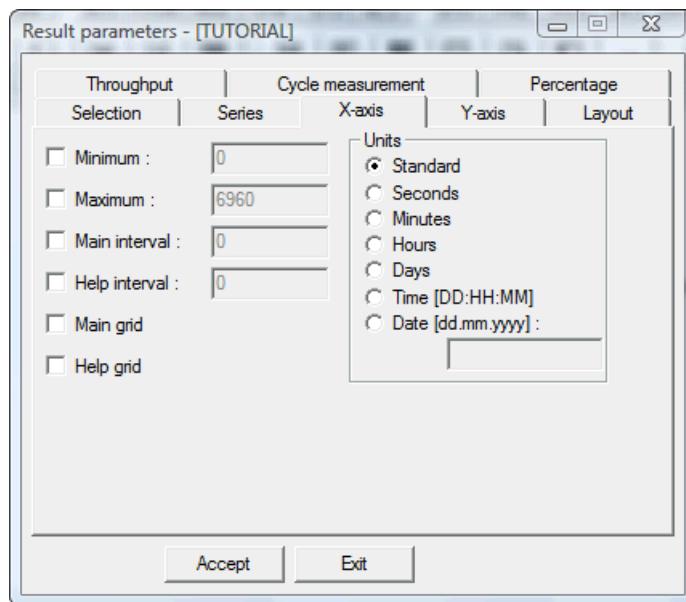


Figure 11.4: Result parameters of X-Axis

Standard	Scaling the x-axis in seconds, minutes or hours, depending on the visible interval.
Seconds, ..., Days	Scaling the x-axis according the selected format.
Time [DD:HH:MM]	Draw time at x-axis with format day:hour:minute.
Date [DD:HH:MM]	According to input of a date in the format day:month:year, the representation of the time in the x-axis occurs relative to this date.
Minimum, Maximum, Step	If this switch is activated, the represented cutout and the scaling of the x-axis can be altered.
Minimum, Maximum, Main interval, Help interval	The limitation of the display period of diagrams, which represents information from several times (e.g. allocation diagram or time diagram), is made by the parameters of the x-axis. For the display of the data only once, the borders are to be selected in such a way that only the desired time lies in the interval.
Main grid, Help grid	If this switch is activated, the represented cutout and the scaling of the x-axis can be altered. For this then, the desired value can be entered in the input fields.
Units	In the layout, vertical lines are drawn at the appropriate interval limits.

Title of the x-axis:	
Standard	Representation of the time in the x-axis in the format seconds, minutes or hours depending on the display interval.
Seconds, ..., Days	Display of the time in the x-axis in the selected format.
Time [DD:HH:MM]	Display of the time in the x-axis in the format Day:Hour:Minute instead of



Date [DD:HH:MM]	minutes. After input of a date in the format day:month:year takes place, the display of the time in the x-axis is relative to this date.
-----------------	---

11.2.4 Y-Axis

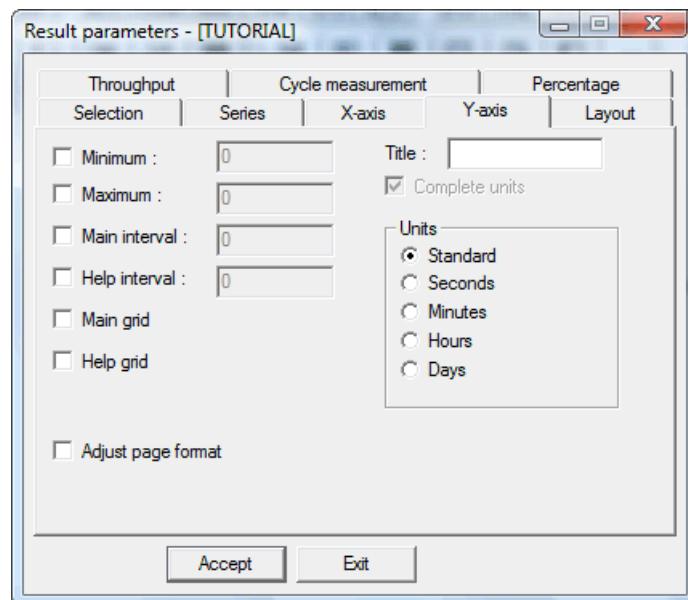


Figure 11.5: Result parameters of Y-Axis

Title	Name of the title caption. When the entry is left empty, the standard entry is used.
Units	Extension of the y-axis title with the selected unit. This is considered also for the caption of the y -axis.
Minimum, Maximum, Step	If this switch is activated, the represented cutout and the scaling of the y-axis can be affected.
Minimum, Maximum, Main interval, Help interval	If this switch is activated, the represented cutout and the scaling of the y-axis can be affected. For this then, the desired value can be entered in the input fields.
Main grid, Help grid	In the layout, horizontal lines are drawn at the appropriate interval limits.
Title	Name of the title of the inscription. With an empty field, the standard entry is used
Units	Addition of the selected unit to the y-axis title. This is considered also for the title of the y-axis
Indiv. every page	Scaling of the y axis only concerning the maximum value in the representation range, not during the statistics period. The average values of the continuous representation refer only to the data within the visible range



11.2.5 Layout

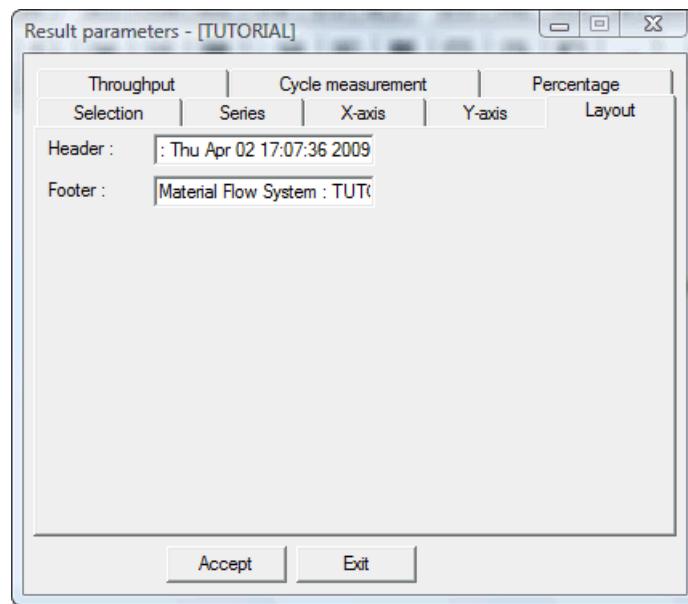


Figure 11.6: Result parameters of Layout

Header / Footer

These texts build the header and footer of the displayed result diagram.

Macros can be used in each line. These start with a \$-sign. The following macros are available:

\$Date	Date when drawing the results
\$Moment	Time of day when drawing the results
\$Modelname	Name of simulation model
\$Simulationdate	Date of statistic file
\$Simulationmoment	Time of day of statistic file



11.2.6 Throughput

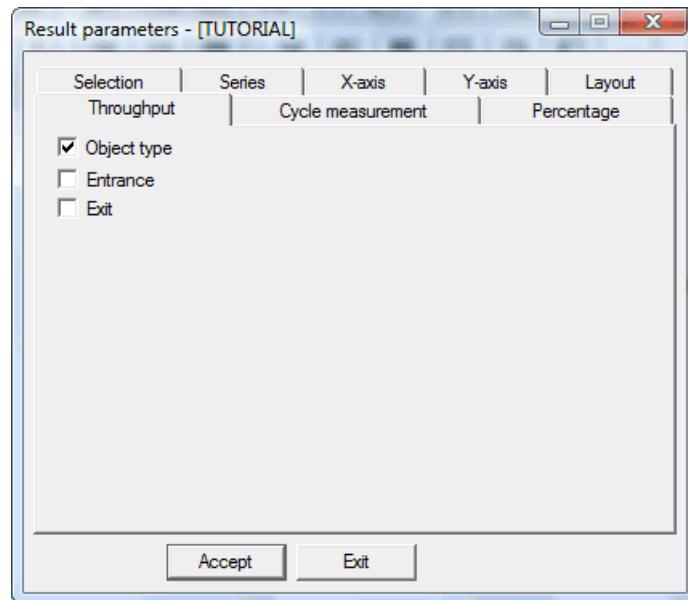


Figure 11.7: Result parameters of Throughput

- | | |
|--------------------|--|
| Object Type | Considers the object type for the representation of the throughput diagram. Otherwise the values are collected independent of the object type. |
| Entrance | Considers the entrance number for the display of the throughput diagram. Otherwise the values are collected independent of the entrance. |
| Exit | Considers the exit number for the display of the throughput diagram. Otherwise the values are collected independent of the exit. |



11.2.7 Percentage

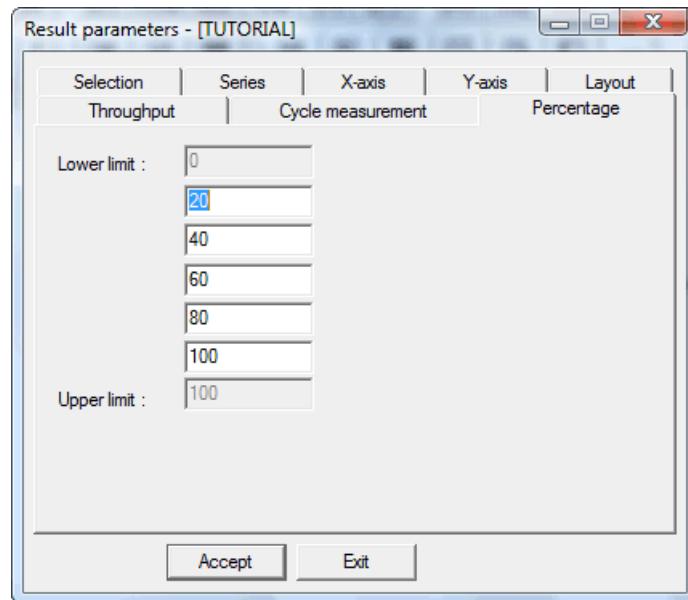


Figure 11.8: Result parameters of Percentage

Values

Ascending values between 0 and 100. These values are considered during the colored display of results in the layout.

11.2.8 Cycle Time

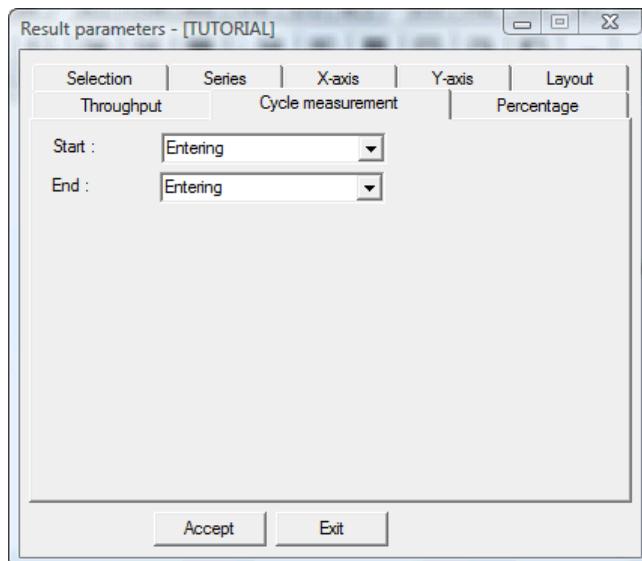


Figure 11.9: Result parameters for cycle time analysis

Start

Event with beginning of the measurement

End

Event at end of the measurement and at the same time of the display time.



11.2.9 Shop Floor Control

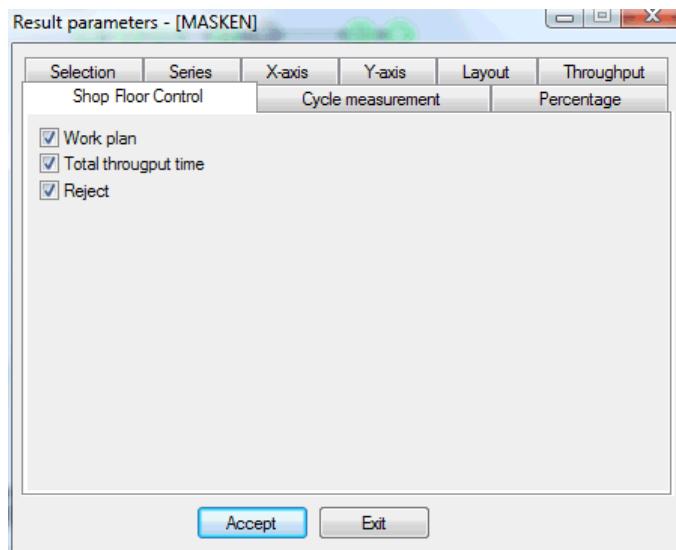


Figure 11.10: Result parameter of the shop floor control

Work plan

The statistics of the work plans will be shown. In the other case the statistic is summarized according the order.

Total throughput time

In the throughput time diagram the total time will be shown. In the other case the time from Start of first operation until end of last operation is drawn.

Reject

The information according rejects can be faded out.

11.3 Result Color

The colors of the result diagrams can be customized.

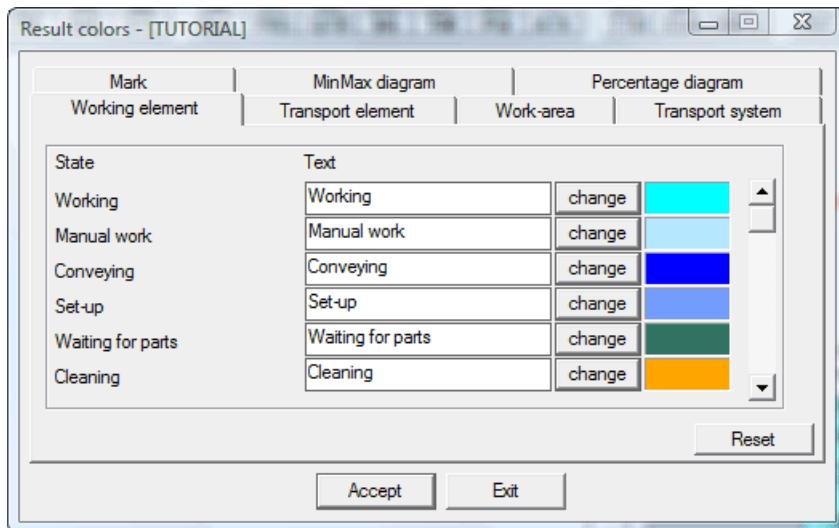


Figure 11.11: Configuration of colors of the result diagrams

For each diagram group, the colors can be customized. This configuration is valid only for the current session.



11.4 Statistics File

DOSIMIS-3 sub-divides the statistics into different types:

Interim Statistics	Statistics since beginning of simulation and/or end of pre-run, if a pre-run moment was defined in the simulation parameters.
Final Statistics	Statistics at the end of the simulation; correspond to interim statistics at end-time.
Abort Statistics	If the simulation is interrupted, abort statistics is written at the moment of interrupt. This corresponds to an interim statistics to the moment of interruption.
Interval Statistic	Statistics since the last statistics output. In the simulation parameters the switch <i>Interval statistics</i> must be activated.
Pre-run Statistics	Statistics before or to the pre-run time moment. This can be both interval statistics and interim statistics.

The moments of time, when these statistics are recorded, result from the values *simulation time*, *pre-run time* and *statistic interval* as well as the selection *Interval statistics* of the simulation parameters.

11.4.1 Total/Standard Statistic

Activating the menu item **Results/Statistical data** displays the previously mentioned ***.slg-file** in the view area. Scrollbars provide scrolling of the view port. Of course, the user can also view this file outside of the DOSIMIS-3 program with the appropriate editor functions. The construction of the ***.slg-file** can be influenced by the user in the usual way regarding pre-run, interval and interim statistics (simulation parameter). Data of these three statistic types are only collected for modules selected in **Simulation>List defining/Statistics list**.

- Module number and module name
- Number of objects put through
- Number of objects in the module at the statistic point of time
- Maximum number of objects in the module during statistic period
- Average occupancy of module
- Utilization of the module in percent
- Blocking time of the module in percent



The following statistic resulted in the case of the example given in chapter *Simulation Run*:

```
*****
Simulation results of the material flow system TUTORIAL
*****
Interval statistics          : Values of elements performance
Material flow system        : TUTORIAL
Statistics for time interval : 0.0 -      60.0
*****
No. Module-
    name     Throu. Act. Max. Min. Aver. Percent. Utili-
                  put   con. con. con. con. cont.% zation %
-----
```

No.	Module-name	Throu. put	Act. con.	Max. con.	Min. con.	Aver. con.	Percent. cont.%	Utilization %	Blocking-times%
1	SOU_1	60	0	1	0	0.00	0.00	0.00	0.00
6	ACC_6	29	0	1	0	0.04	4.03	7.96	0.00
8	WST_8	29	1	1	0	0.78	78.33	78.33	0.00
10	WST_10	25	1	1	0	0.76	75.99	75.99	9.56
11	COM_11	53	0	1	0	0.31	30.67	30.67	23.31
12	DIS_12	53	0	1	0	0.46	46.00	46.00	38.64
13	SIN_13	47	0	1	0	0.65	64.94	64.94	0.00
17	ACC_17	28	1	2	0	0.32	7.92	7.92	13.58
19	receipt	51	9	10	0	5.19	51.90	51.90	87.74
20	SHU_18	56	1	1	0	0.50	50.38	98.61	0.00
21	acctop	26	0	2	0	0.69	13.77	13.77	42.22
22	accbottom	30	0	4	0	0.97	19.40	19.40	39.82
23	ACC_23	6	0	1	0	0.11	3.67	3.67	8.51

Figure 11.12: Results output (Statistic data)

At this point, it should be mentioned again that, for interim statistics, the statistic period extends from the end of the pre-run to that moment of the protocol interval but, for interval statistics, refers only to the period within a protocol interval.

Module name and module number correspond to the model definition of the user.

The **Throughput** of a module is increased, as soon as an object leaves it.

The **Actual contents** of the module are the current content when the statistic is written.

When an object enters an element, it is checked, whether the old **Maximum contents** are exceeded and the value will be increased.

The **Minimum contents** are analog to the maximum contents, the minimum count of objects, which was in the element during the statistic period.

For each object passing a module the residence time in the module is measured. The sum of these residence times together with the already performed residence times of objects still in the module equal a certain time value for the statistic period. The ratio of this value to the whole of the observed statistic period gives the average number of objects (average **contents** of the module).

The **Percentage contents** of a module is obtained as a ratio of average allocation to module capacity.



The **Utilization** contains on the one hand the percentage contents of the element. Additionally, the strategic times when the element is empty, such as empty run of a shuttle or time for exiting, when forward control is active, belong to this utilization. Traveling to the basic position (Exception: The module has only one entrance. The shuttle travels always to entrance 1, because the next objects will be taken from there.), downtimes of failures, maintenance or breaks and waiting for workers for repair do not belong to the percentage utilization of an element. The utilization refers only to the non-failure period. When a workstation, for example, was down for 15 minutes during one hour and work was active for 27 minutes of the remaining 45 minutes, the utilization of that station is 60% (27 of 45 minutes).

The **Blocking times** are obtained from the sum of the times in which an object stands by at the exit of a module.

Depending on the type of module still further conditions are evaluated as blockings:

Working station, Assembly, Disassembly, multifunctional Workstation:

When the object cannot be processed, because the old object did not completely leave the component yet and the length > 0 is indicated, this is assigned to blocking time.

Disassembly, Unloading station:

The object, which should be disassembled / unloaded, cannot be delivered, because the module, which should take the object, cannot accept it (failure).

Storage If no object waits at the entrances and a lining up order for outsourcing cannot be executed, because of a back pressure at the follow-up modules, this is proven as blockade.

11.4.2 Utilization Statistics

The utilization statistics indicates the proportional portions of that utilization, which all components possess together. Despite this context, the statistics are placed before the component-type-specific statistics in the result file.

```
*****
Interval statistics          : Utilization statistics
                               for modules of type Workstation
Material flow system        : Results
Statistics for time interval : 0.0 -      60.0
*****
No. Module-      Wait for   Blocking- Failure- Mainten. Breaks- Transport- Waiting- Without
      name       worker%    times%     times%    times%   times%   times%   times%   stat%
-----
8 WST_8           0.00      0.00      0.00      0.00      0.00      3.89      31.78      0.00
10 WST_10          0.00     15.68      0.00      0.00      0.00      3.89     13.69      0.00
*****
```

Figure 11.13: Utilization Statistics

Waiting **for worker** is the period, when the elements waits for personnel. This results from way times and missing capacities. This may be according to object processing or elimination of failures.

For the sake of clarity the value of the **blocking time** from the standard statistics is repeated here again.

Downtimes are caused by module **failures, maintenance or breaks**.



The percentage component of the period, where failures with *no statistics* are active for that element, will be recorded by means of **Without statistics**.

The value of **transport times** indicates the time, when an object enters the module. This value may be zero, if transportation is examined more detailed, like in the shuttle or turning table.

Waiting **times** are the times in which a module is empty and is waiting for an object.

After a simulation run this data is kept in the form of **Final statistics**.

11.4.3 Additional Statistic Dependent on Modules

For certain modules additional statistics can be created. This is mainly data concerning type-dependent module status for which the percentage time portion is given for the respective state of the module. For this purpose, the elements must be selected by the menu option **Simulation/Define Lists/Additional Statistics**. These statistics are available for the module types: workstation, assembly, disassembly, loading station, unloading station, turning table, shuttle and paired shuttle.

11.4.3.1 Statistics for Module with Work

Under these groups the elements in which processing can take place are summarized. These are workstation (WST), assembly station (ASS), disassembly station (DAS), loading station (LOA) and unloading station (UNL). Under classification of the times of workstations, the following values are listed. Most values apply to all element types specified above. Values, which are element-type dependent, are separately marked.

Module number	See <i>total statistic</i>	
Module name	See <i>total statistic</i>	
Processing times	Working time without personnel	
Manual processing	Working time with personnel	
Driving times	Entering the module. With components with (implicit) forward control the exiting of the object belongs to it too.	
Waiting times	Waiting while no object in element	
Set-up times	Working time according to set-up table	Only WST and MFW
Cleaning times	Working time according cleaning processes	Only MFW
Waiting for transfer (parts)	Time when no material was available at the assembly entrance	Only ASS, MFW
Waiting for unloading	Time when a disassembled object enters the successor	Only DIS, MFW
Without statistics	Duration, when failures <i>without statistics</i> are active for this station.	
Pieces handled	Count of objects processed at this station.	
Count fault	Count of failures, which have been active for this station.	



Interval statistics	:	Classification of utilisation for modules of type Workstation					
Material flow system	:	Results					
Statistics for time interval	:	0.0 - 60.0					

No.	Module-name	Set-up times%	Waiting times%	Proces. times%	Manual proces.%	Driving times%	Pieces handled Count fault
8	WST_8	3.33	0.00	0.00	60.99	0.00	27 0
10	WST_10	3.33	0.00	0.00	63.41	0.00	27 0

Figure 11.14: Results of output, additional statistics of workstations

Waiting **times** are the times in which a module is empty and is waiting for an object. This value is always zero according to a new organization of the statistic tables. The waiting time can be found in the utilization statistic.

Set-up **times** are only shown in the result output for workstations, if the station has to be reset to process a new type.

For the processing part, the **Processing times** of all object types completely processed in the statistic period are added.

For the manual processing of parts, the **Manual Processing** of all object types completely processed in the statistic period are added.

Driving **times** are added in the same way. In both cases, the corresponding time portion of the object just being processed is also added.

The **Waiting times for assembly** are times, when the list of assembly is not yet ready and an object is waiting at the assembly entrance (no. of entrance > 1).

11.4.3.2 Statistics of Modules with Transportation

Under these groups the elements in which a complex feed takes place are summarized. These are the shuttle (SHU), the paired shuttle (PSH), as well as the turning table (TUT).

Number of module	See <i>total statistics</i>	
Name of module	See <i>total statistics</i>	
Transport unit		Only SHU
Transport runs	Transport of an object	
Charge times	Loading of an object	
Discharge times	Unloading of an object	
Empty runs	Travel to an entrance for loading an object	
Initial position runs	Change to basic position, since after unloading no further order is present.	
Waiting for transfer	Waiting for further object at an entrance	Only SHU with capacity > 1
Connecting run	Travel to an entrance, which was determined by the decision table	Only SHU with decision table
Waiting times	Waiting since no object is in the element or no travel job is present (empty run).	



Single run	Only PSH							
Double run	Only PSH							
Lifting time	Only PSH							
<hr/>								
Interval statistics	: Classification of transport times for modules of type Shuttle							
Material flow system	: Results							
Statistics for time interval	: 0.0 - 60.0							
<hr/>								
No. Module- name	Trans- portunit	Wait- times%	Transport- runs%	Charge- times%	Discha.- times%	Empty- runs%	Inipos.- runs%	Trans- ports
-----	-----	-----	-----	-----	-----	-----	-----	-----
18 SHU 18	0	0.00	33.17	9.17	9.01	33.23	0.00	59
<hr/>								

Figure 11.15:Result output, additional statistics of shuttles

If an object reports for acceptance at the entry to a shuttle, the **Loading time** is calculated from the time necessary for the object to travel completely on to the vehicle table. The parameters *Loading route* and *Loading speed* of the shuttle are decisive here.

A **transport run** consists of the run of a loaded vehicle till the point of delivery.

The **discharging time** is determined by two components. The first one is the time required to pass the unloading route which was defined by the parameters of the vehicle. It is influenced by the unloading speed which at this point has also been defined. The second one is influenced by the parameters of the corresponding successor module and lasts from the exiting of an object which has been reported to a successor module until it enters this module completely.

Empty **runs** occur when the empty vehicle must drive from its current position to another entrance in order to pick up an object there.

If the empty vehicle is waiting for a transport order, the **Waiting time** portion is increased. Blocking and downtimes are determined as in the total statistics.

If a basic position is entered for the shuttle, these runs are registered under **Initial position run**.

For the paired shuttle, generally the same applies for transport, loading and unloading times as for the shuttle. The times for run with one object and run with two objects are determined by the times, in which the shuttle is loaded with one or two objects. Additionally it must be observed which times are dependent on which module parameters. Compared with other module types, object pick-up and object drop-off can be overlaid for paired shuttles. Therefore this process is broken down separately for each connection. For each entrance the respective number of double cycles type 1, type 2 and single cycles are entered, analogue to them the double cycles 1 and 3 and single cycles at the exits.



11.4.4 Throughput Statistics

If throughput statistics are selected, a table is created for all modules selected by **Simulation/Define lists/Additional statistics** containing the number of objects delivered (taken at sinks), separated by types. In case of a sink, the objects taken are counted. Under the heading of the module name, the type of object and per object type, the number of the achieved objects, their total and the modules' throughput results are indicated. The throughput is differentiated by the start-destination relation the objects have taken passing the module. The throughput statistics can be switched off in simulation parameters for all modules.

Interval statistics	: Throughput statistics For modules of type shuttles				
Material flow system	: Results				
Statistics for time interval	: 0.0 - 60.0				

No.	Module-name	Object-type	from entrance	to exit	Number of objects

18	SHU_18	1	1	1	28
		2	1	2	25
		10	2	1	4
		20	2	2	4

Figure 11.16: Results output, throughput statistics

11.4.5 Throughput Time Statistics

In the throughput time, the maximum, minimum and average conveyance times are recorded, as well as the number of objects of the relevant type recorded in the observed period. The throughput-time statistics can be switched off in the simulation parameters.

Interval statistics	: Throughput-time statistics			
Material flow system	: TUTORIAL			
Statistics for time interval:	0.0 - 60.0			

Object-type	Number of objects	Average rtt	Minimal rtt	Maximal rtt

1	19	664.596	222.433	1123.891
2	21	909.564	270.015	1380.492

Figure 11.17: Results output, throughput-time statistics

By the passage of an object through the system, the period between the generation of an object and its leaving the system is meant.

An object can be allocated a new type in the workstation, assembly and disassembly modules, a fact which is taken into account in the throughput-times statistics as follows:

As soon as an object is generated in the system it is given a time mark which is registered and recorded in the statistics as soon as the object leaves the system. If the object type changes in this period, e.g. by processing in a work station, the time mark of the object is rewritten. In the statistics, the complete throughput time is recorded under the object type with which the object was last assigned.



Furthermore, it must be remembered that objects are not only generated in the source but also in disassembly modules. There, those objects which have been disassembled from a basic object are treated as new objects and are even given a new number of their own. In principle, the same thing happens with a basic object type from the disassembly module as in the workstations, i.e. the time mark of an object to be disassembled is passed on to the new basic object type (cf. chapter: Disassembly Station).

Analogue to this, objects do not only leave the system in sinks but also in assembly modules as long as they are assembled on to another object. In this case, the time mark of the assembled object is passed on to the statistics, as opposed to the newly-generated object type which receives the time mark of the former basic object type and continues its process through the system(cf. chapter: Assembly Station).

The throughput times can be transferred in the assembly or disassembly from/onto the basic object. Therefore both modules offer a checkbox in their dialog to do so.

The processing time statistics are automatically set up by the system, as was the case for work area statistics, for each statistic interval selected by the user. The objects are then only given a time mark when generated and are not recorded in the statistics until they leave the system (see above). This is of significance for the interval statistics.

11.4.6 Cost Statistics

11.4.6.1 Module Costs

In the module cost statistics on each component, in which state costs are defined, a data record is created. This contains the costs results of each state.

Interim statistics						: Module costs statistics (data in [mu])
Material flow system						: Results
Statistics for time						: 60.0

No.	Module-name	Failure time	Maintanace time	Down-time	Working	Set-up Without statistic
-----	-----	-----	-----	-----	-----	-----
1	SOU_1	0.0	0.0	432000.0	0.0	0.0
10	WST_10	0.0	0.0	0.0	237110.1	0.0

Figure 11.18: Results output, Module Costs



11.4.6.2 Object Costs

In the object cost statistics for each type of object, which leaves the system, the assigned costs are provided. This is done via the indication of the minimum, middle and maximum collected costs.

```
*****
Interim statistics : Object costs statistics (data in [mu])
Material flow system : Results
Statistics for time : 60.0
*****
Object- Number of Average Minimal Maximal
type objects cost cost cost
-----
1 23 64378.11 25766.86 129621.18
2 24 45304.03 12753.06 136975.05
*****
```

Figure 11.19: Results output, Object Costs

11.4.6.3 Worker Costs

In the chart of worker costs, the state costs for each worker can be found.

```
*****
Interim statistics : Worker cost statistics (data in [mu])
Material flow system : Results
Statistics for time : 60.0
*****
No. Work area Worker Stand- Working Super- Without
name no. still vising statistic
-----
1 WRA_1 1 44569.8 259981.1 0.0 0.0
1 WRA_1 2 42859.8 277441.4 0.0 0.0
*****
```

Figure 11.20: Results output, Worker Costs

11.4.7 Distribution of Random Time

During some random processes, a special statistics is carried out, from which it can be read, how well the adjusted mean value was achieved during simulation. This can, among other things, be used to determine the duration of the pre-run.

11.4.7.1 Distribution of Processing Time

```
*****
Interval statistics : Distribution of work time
Material flow system : TUTORIAL
Statistics for time interval: 0.0 - 60.0
*****
No. Module- Object Number of Minimum Maximum Average Dispersion
name type values value value value (relative)
-----
2 23 70.99 90.22 81.54 5.81
20 7 72.82 89.80 80.71 7.36
8 WST_8 total 30 70.99 90.22 81.34 6.22
10 WST_10 1 20 72.84 90.22 80.26 5.61
10 3 74.24 86.99 79.13 7.09
10 WST_10 total 23 72.84 90.22 80.11 5.84
*****
```

Figure 11.21: Results output, Distribution of processing time



11.4.7.2 Distribution of Failure Time

```
*****
Interval statistics      : Distribution of failures and breaks
Material flow system    : WERKERB
Statistics for time interval : 0.0 - 120.0
*****
No.       Name      Type      Definitiontype
......
Distribution      Number of   Minimum   Maximum   Average   Dispersion
of          values     value     value     value     (relative)
-----
1 sobli        disturbance random
distance        13      330.77   560.35   445.62   14.19
duration       13      39.73    88.54    58.57    24.25
5 sunli        disturbance random
distance        17      251.85   348.07   299.36   9.81
duration       17      22.43    37.28    29.70   12.12
6 sobre         disturbance random
distance        8       678.61   834.79   759.29   7.46
duration       8       119.55   173.12   146.84   12.60
7 sunre         disturbance random
distance        11      441.88   644.98   529.84   11.72
duration       11      49.82    67.46    60.62    7.78
*****
```

Figure 11.22: Results output, Distribution of failure time

11.4.8 Further Statistics

Dosimis-3 supports more statistics, which are adapted specially to controls. These can be found in the description of the control

Work Area
Transport System
Shop Floor Control
Capacity Monitoring
Throughput Time Measurement
Measuring Element
Optimization

11.5 Results Output in the Layout

The results output in the layout visualize the data of standard statistics (performance measures of modules). For the result display, there is a toolbar that facilitates selection for the display in the layout. To the left there are three buttons available, with which navigation is possible through the individual statistic time periods. With the first the change between interval statistics and interim statistic is supported. With the next two buttons, stepping through statistic time periods is possible.

With the aid of the fourth-last switch, the result parameters can be activated. The result display in the layout is completed by clicking the third-last switch.

With the next two buttons, the display of values and additional information can be activated. These two and the first button manage not only the layout graphics, but also several further results graphics.

The remaining buttons activate the different graphics.



Figure 11.23: Toolbar of the result display

All result graphics can be exported, can be transferred (Shift+B, Shift+W) as graphics into the clipboard. Furthermore, a CSV format (Comma Separated Values) can be created with the key combination Shift+C that can be transferred directly into a spreadsheet (for example, Excel). This happens in the target application in general to the insertion menu (Ctrl+V).

11.5.1 Representation of Percentages

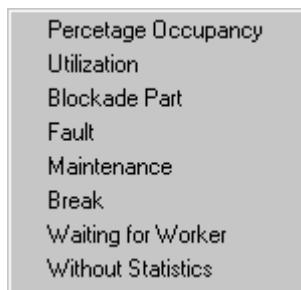


Figure 11.24: Selection of representation of percentages



For representation of statistical data in the layout, the above alternatives are available. These correspond to the percentages from the standard statistics. The exact values are normally not indicated. However this can be activated via the parameter *Show values* from the result parameter dialog. The statistics time instant can be selected as desired also in the result parameter from the statistics intervals.

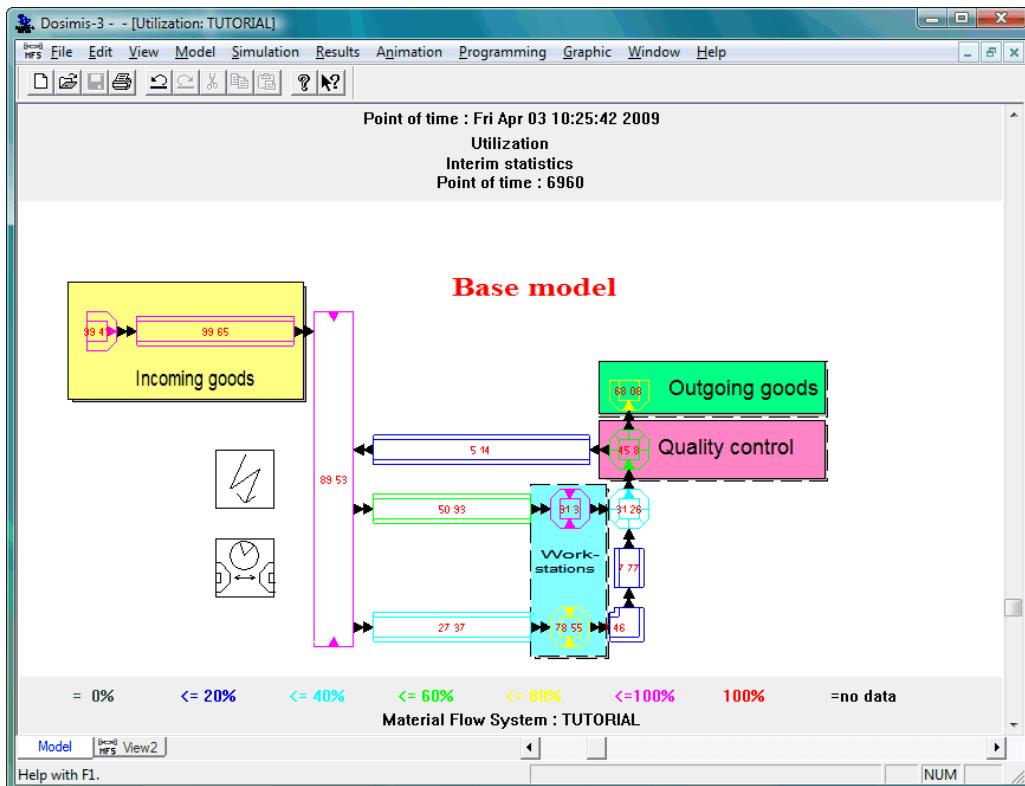


Figure 11.25: Utilization diagram

11.5.2 Representation of absolute Value

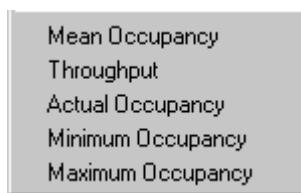


Figure 11.26: Selection of representation of absolute value

It is possible to display the corresponding data in accordance with the current parameterization of the result output in the layout for the above values. These correspond to the values of the modules from standard statistics.

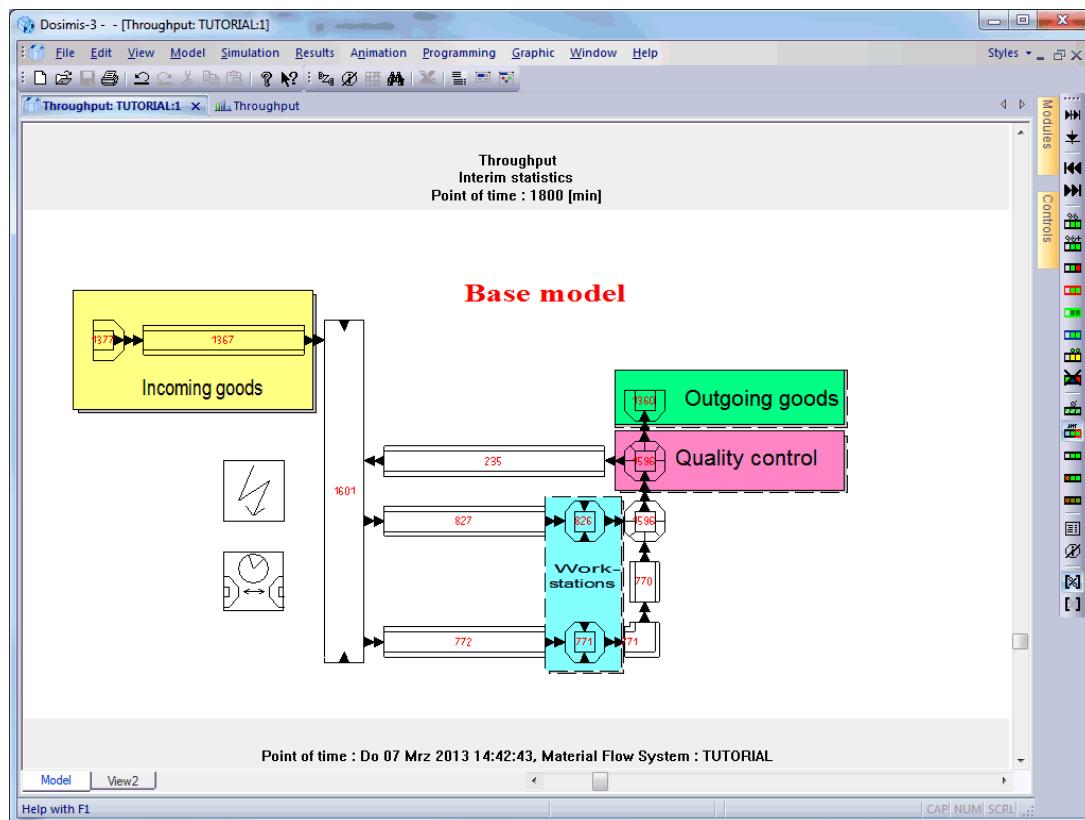


Figure 11.27: Throughput at a statistics time instant



11.6 Standard Results Diagram

All results, which can be viewed at particulars points in time in the layout, can be represented also over several statistic points of time in a separate diagram.

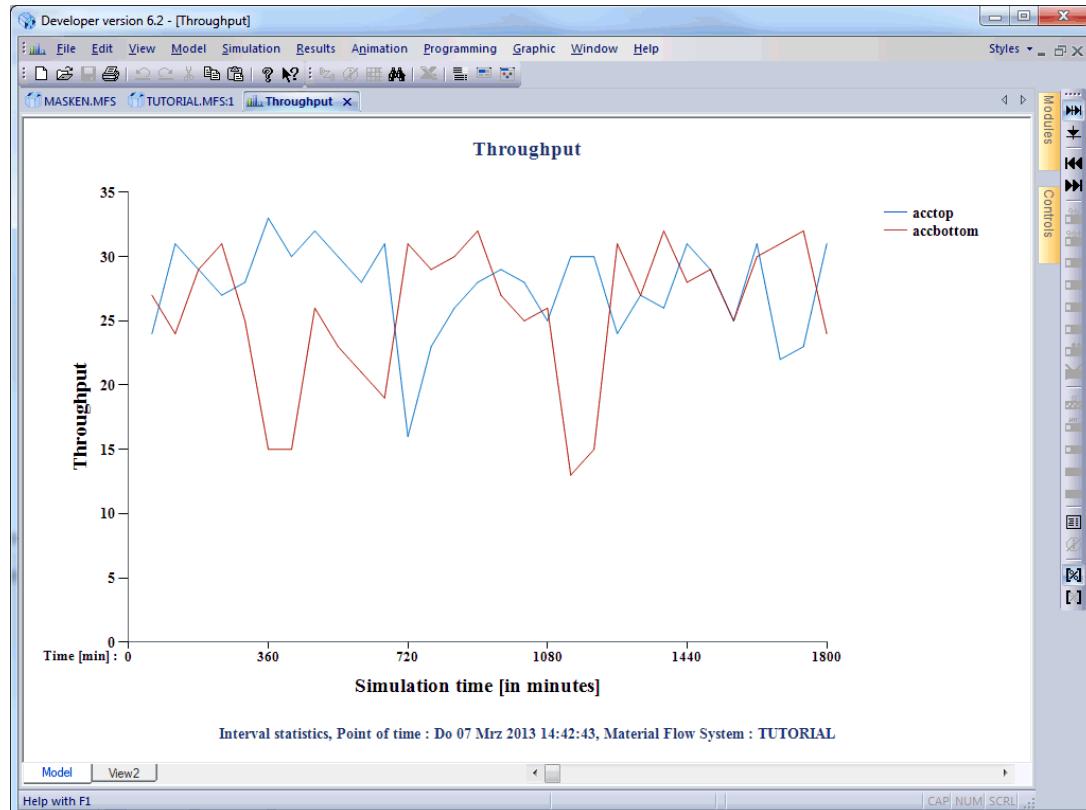


Figure 11.28: Standard result diagram: Throughput



Here the representation of the average occupancy offers a special case. If only one element is selected, the minimum and maximum occupancy of this element can be displayed by the switch *Additional information*.

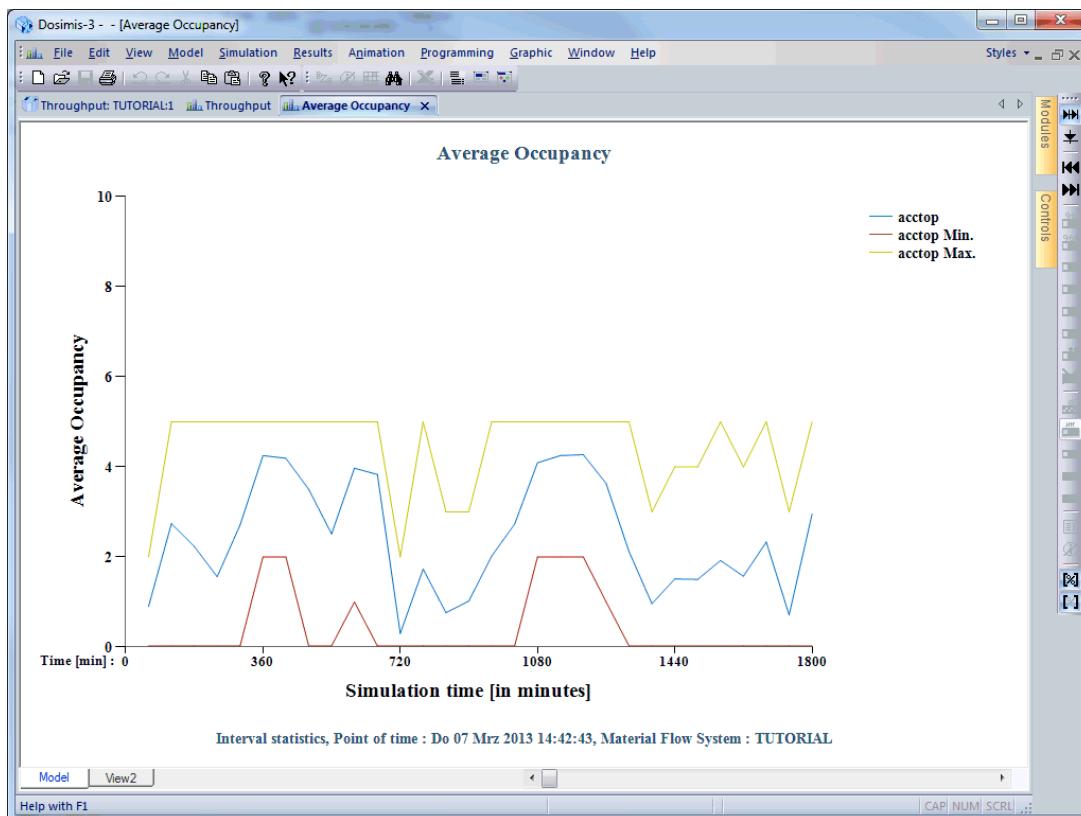


Figure 11.29: Additional information with standard result diagram



11.7 Cost

Under the generic term cost, the cost-relevant statistics are summarized. These are, on the one hand, the capital commitment in the system and evaluations on the state-dependent costs.

11.7.1 Capital Commitment

With the help of diagrams of the absolute costs, the capital commitment of the system can be analyzed.

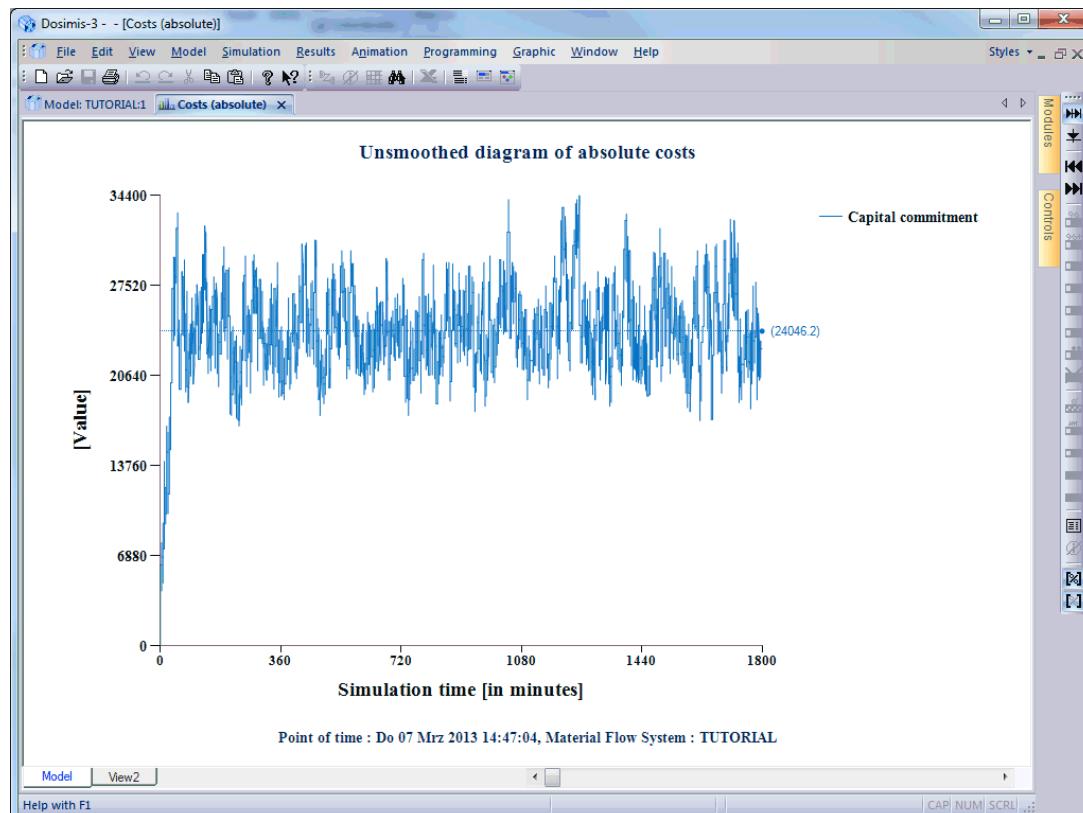


Figure 11.30: Diagram of Capital Commitment



11.7.2 Object Cost

In the diagram of the object costs, the measured value of the objects when leaving the model is displayed. For this the minimum, middle and maximum value is displayed in each case. The object costs are the summed state costs during the model run.

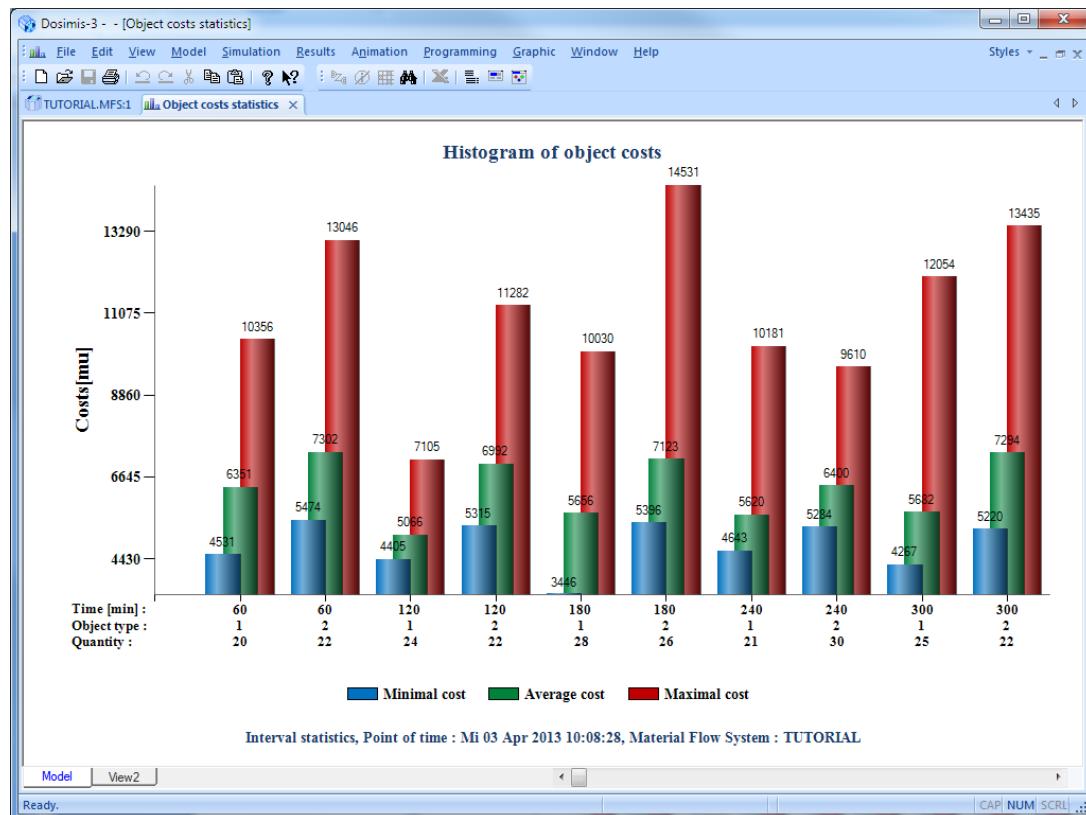


Figure 11.31: Diagram of object costs



11.7.3 Module Cost

In the diagram of module costs, the summed values of the state costs during the statistic time period are displayed.

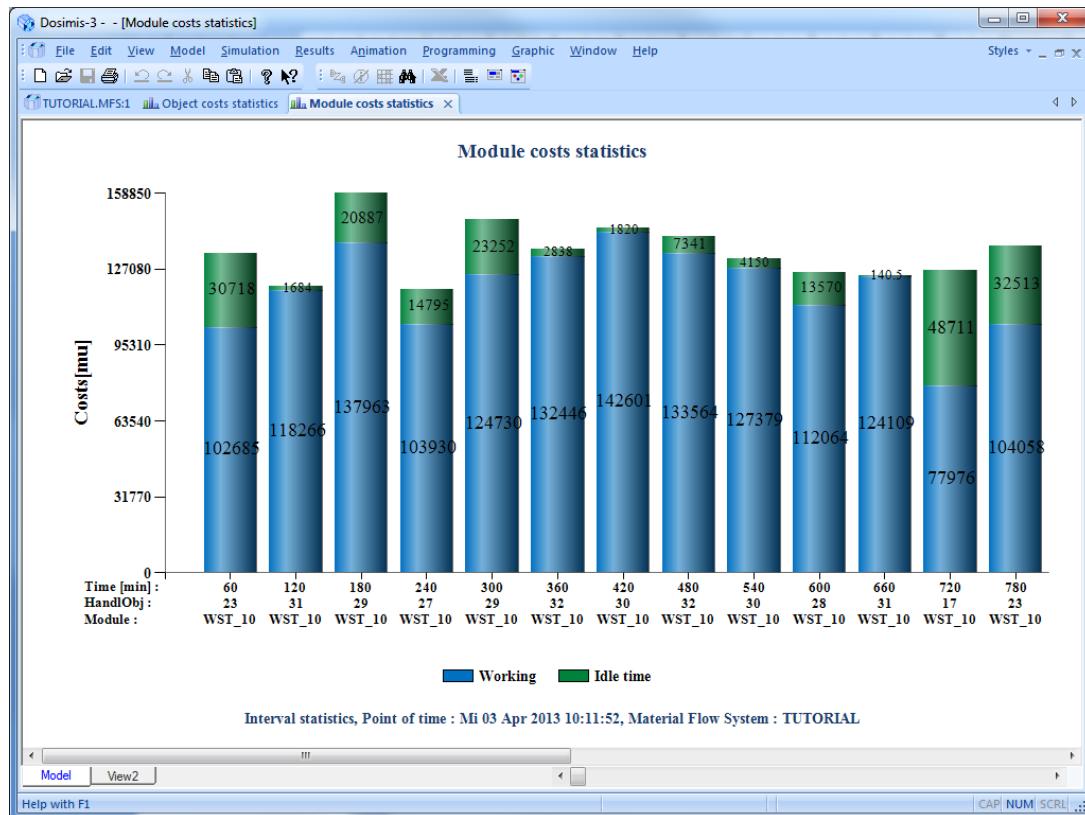


Figure 11.32: Diagram of module costs



11.7.4 Worker Costs

In the diagram of worker costs, the summed values of the state costs during the statistic time period are represented.

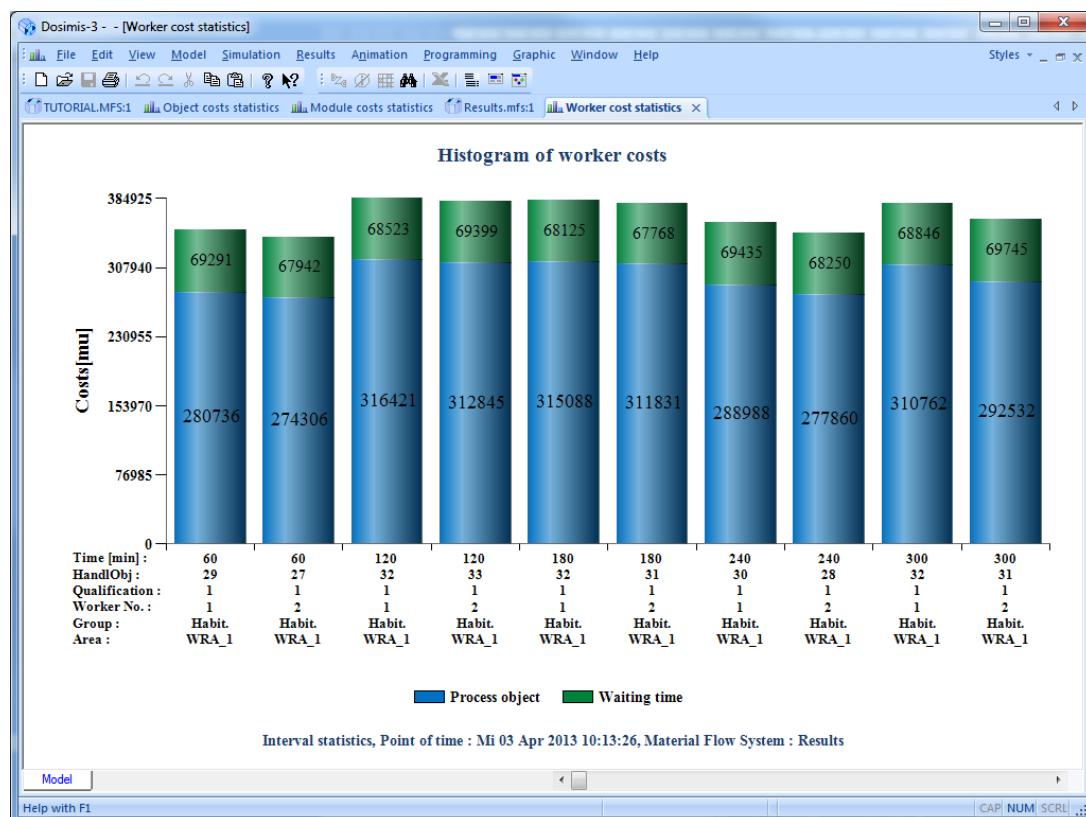


Figure 11.33: Diagram of worker costs



11.7.5 Cost Measurement

In the module for throughput-time measurement, the minimum, mean and maximum costs of the selected areas are also displayed.

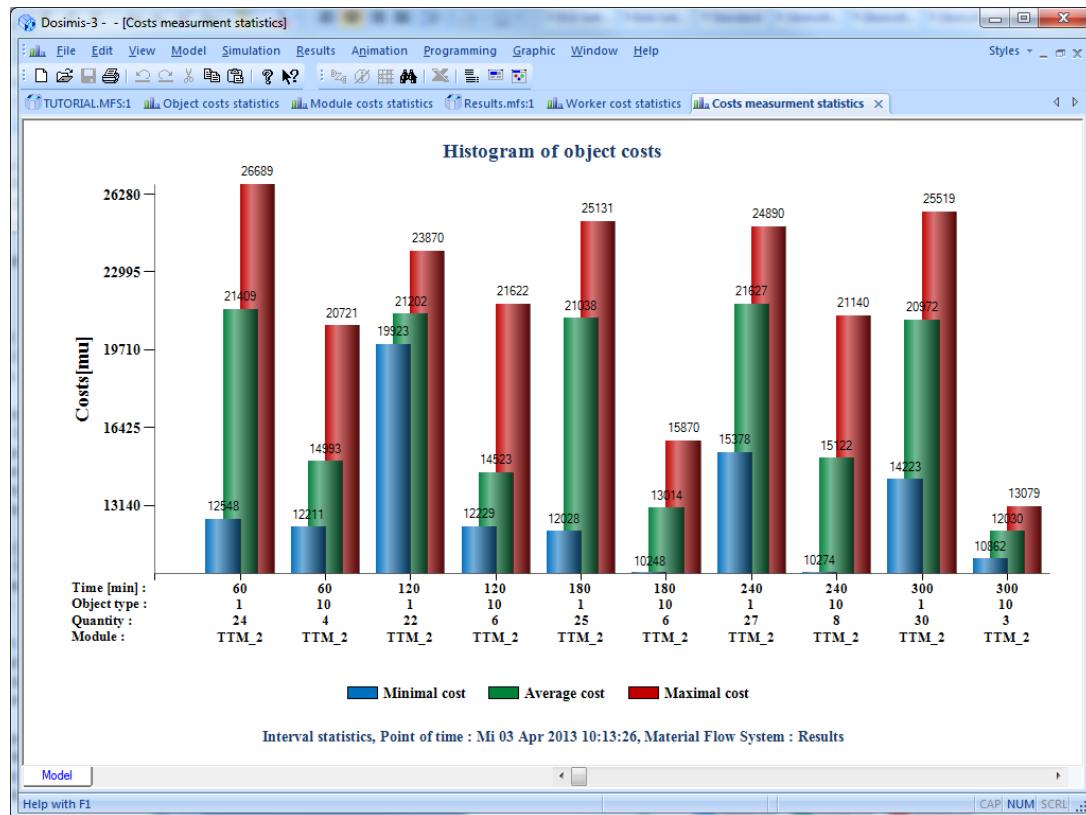


Figure 11.34: Diagram of cost measurement



11.7.6 Cost Measurement (single)

With the help of the diagram, each individual measurement is visualized. Thus, the time at which particularly large costs appear or how, for example, failure affects behavior can be analyzed more exactly.

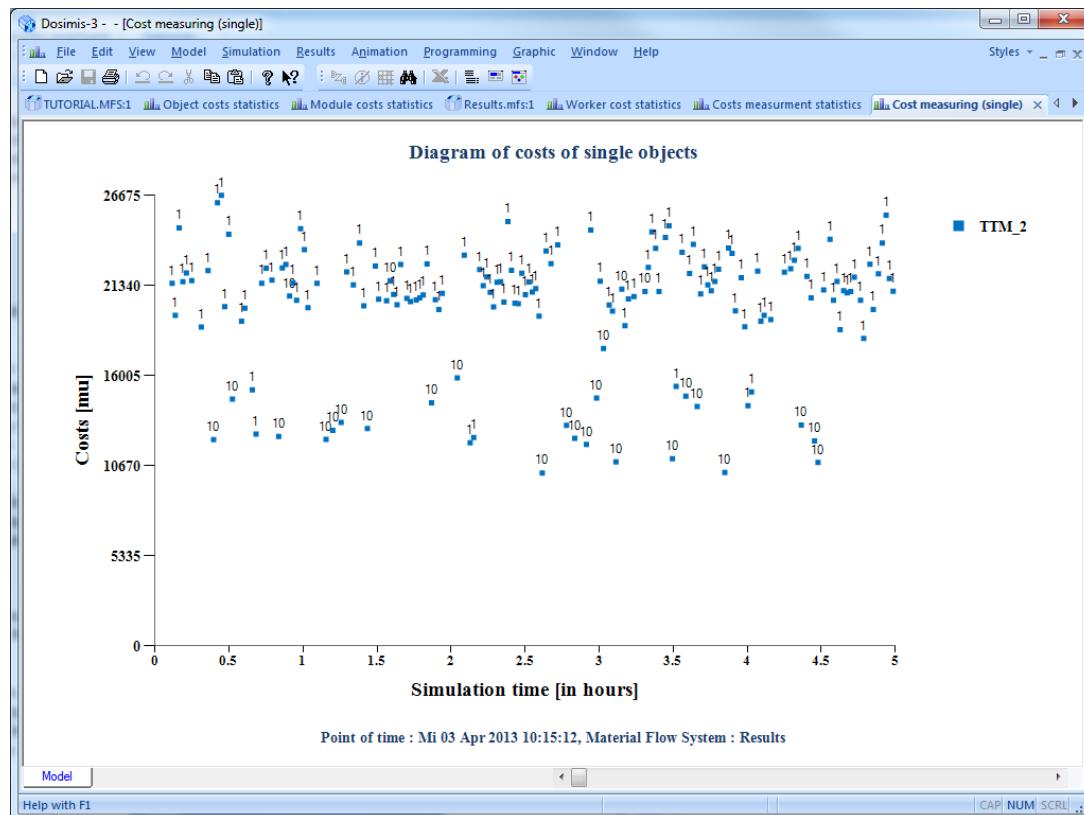


Figure 11.35: Cost measurement (single)

11.8 Continuous Diagrams

These diagrams are based on the data of the action protocol (trace file). This implies that these diagrams can be displayed only if the action protocol is activated for the corresponding modules. The modules for which this record should be operated must be defined via the submenu *Simulation/Define lists/Trace list*. With the help of the functions *Zoom enlarge*, *Zoom reduce*, the display window can be increased or reduced. Subsequently, the window can be shifted with the help of the scrollbar.

All results graphics can be transferred as graphics into the clipboard by the export functions (**Shift+B**, **Shift+W**). This is also possible for results graphics in a separate window with the key combination **Ctrl+C**. The **CSV-format** is only available for continuous diagrams, when exactly one element is selected.

Based on experience, occupancy diagrams usually occupy several megabytes of data space. Thus, there is the possibility of importing these only for selected modules. If, during the selection of a continuous diagram the **CTRL-key** is pressed, only the data for the selected components are read in. If further information for other elements is needed after that, the current data has to be deleted. This happens by means of the function *Rescan* from the result dialog.



11.8.1 Buffer Analysis

Under the term buffer analysis, different analyses are summarized, from which the behavior of elements, which can take up several objects at the same time (accumulation conveyor, conveying section,...), can be analyzed more exactly.

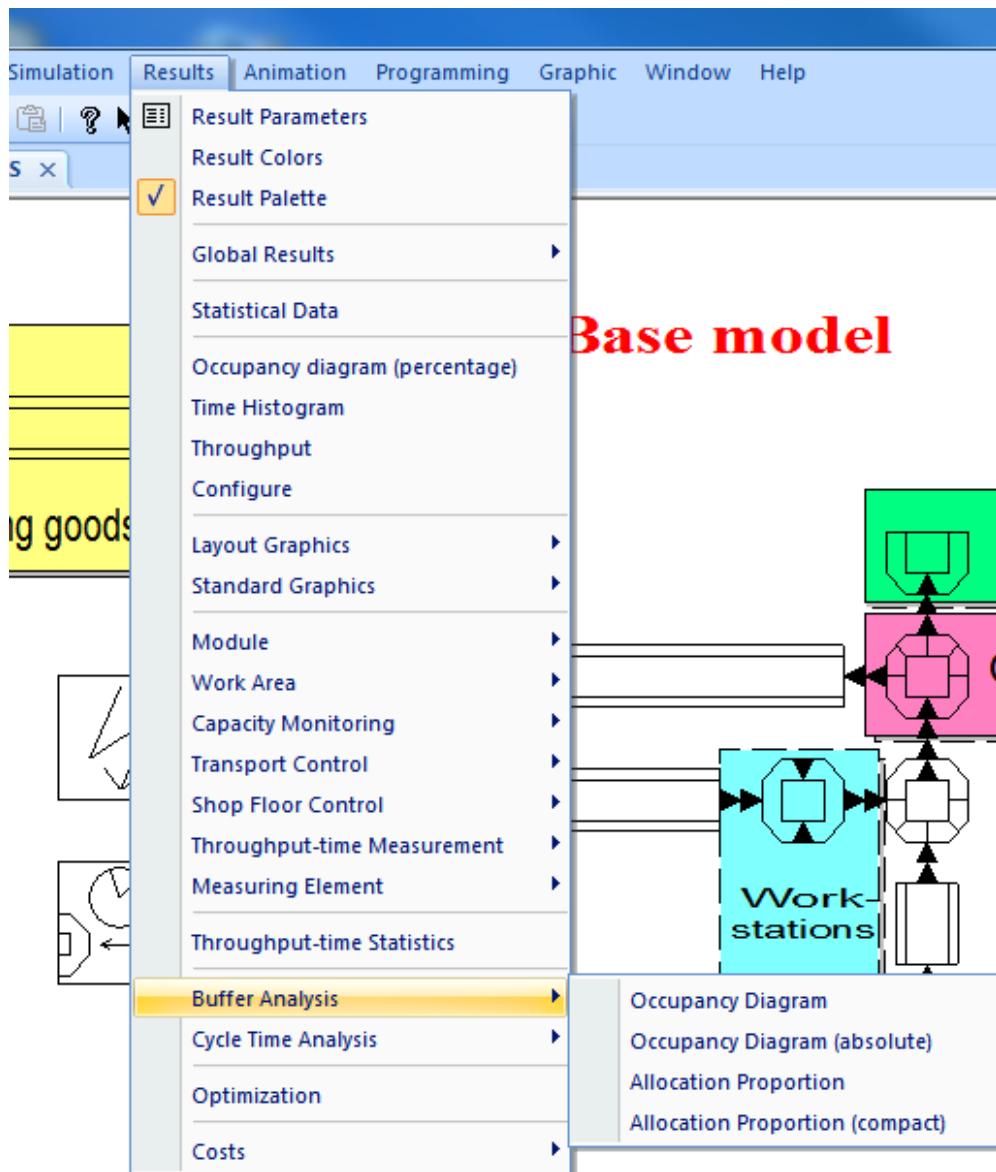


Figure 11.36: Menu „Buffer Analysis“

11.8.1.1 Occupancy Diagram

Occupancy diagrams allow a comparison of the percentage occupancy of several modules. Before such a diagram can be made, those modules which are to be analyzed are selected and the time period to be analyzed is fixed. Up to eight modules can be selected; their indicators appear in one of the eight colored bars. The data of this module is then shown in the respective color in the diagram. Single modules can be selected by a click with the mouse or by framing the modules with clicked mouse button. The selected elements are displayed in red.



Furthermore, it is necessary to enter the time period in the result dialog, which is to be processed. At this point, however, the user does not have to adhere to the fixed statistic time points of the *.slg-file but can enter any time interval (in minutes) of the simulation time, because the necessary data are read in from the movement protocol (*.tra-file) and not the statistic file. The meaning of the likewise entered interval number depends on whether the graphic presentation is to be made in the form of a smoothed or unsmoothed diagram.

In an unsmoothed diagram a horizontal line is drawn at the height of the current utilization. A change in the utilization level is shown by a vertical line at the height of the new level. Then, a new horizontal line follows for the next change and so on. This form of presentation is suitable for showing each object movement especially on tracks or for slewing belts. Therefore the following diagrams are made for one of the two buffers recorded in the additional statistics.

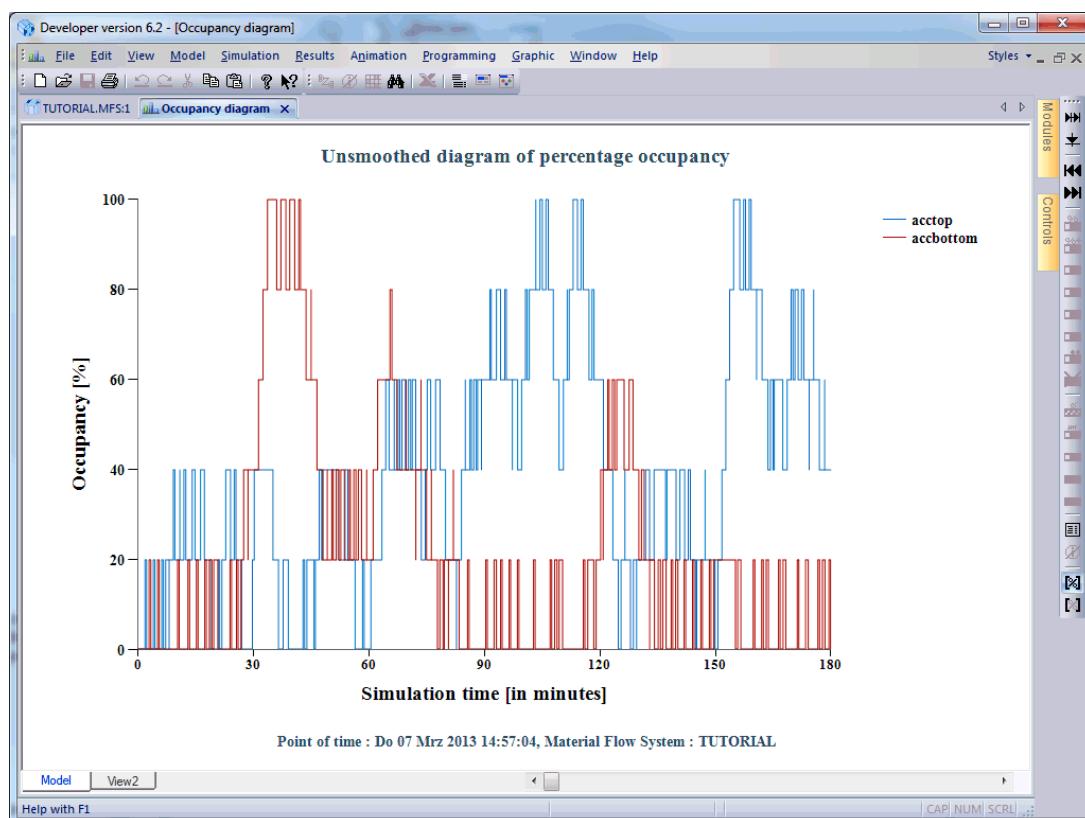


Figure 11.37: Unsmoothed diagram of percentage occupancy

In the period under observation from 0 to 300 minutes, each change in the utilization can be seen in the diagram.

11.8.1.1.1 Improving the Graphic

- | | |
|------------------|--|
| X-axis | The x-axis can be scaled arbitrarily. |
| Y-axis | The y-axis can be scaled arbitrarily. |
| Mean value: | The average value of all entries is faded in as reference line. |
| Indiv. per page: | The scaling of the y-axis results from the min and max values of the data within the visible range. If the average value is shown, this is formed only over the data within the visible range. |



Object types: The measured values can be limited to the lower and upper limit of the object types.

11.8.1.2 Smoothed Occupancy Diagram

If object movements can no longer be so easily reconstructed, these can be combined with the aid of the smoothing factor via the menu **Result parameters**. The smoothing factor determines into how many intervals the representation period of time should be subdivided. In each of these intervals, the movements are combined. Smoothing 0 means, that all movements are represented, smoothing of 1024 means, that the representation period of time is subdivided into 1024 intervals.

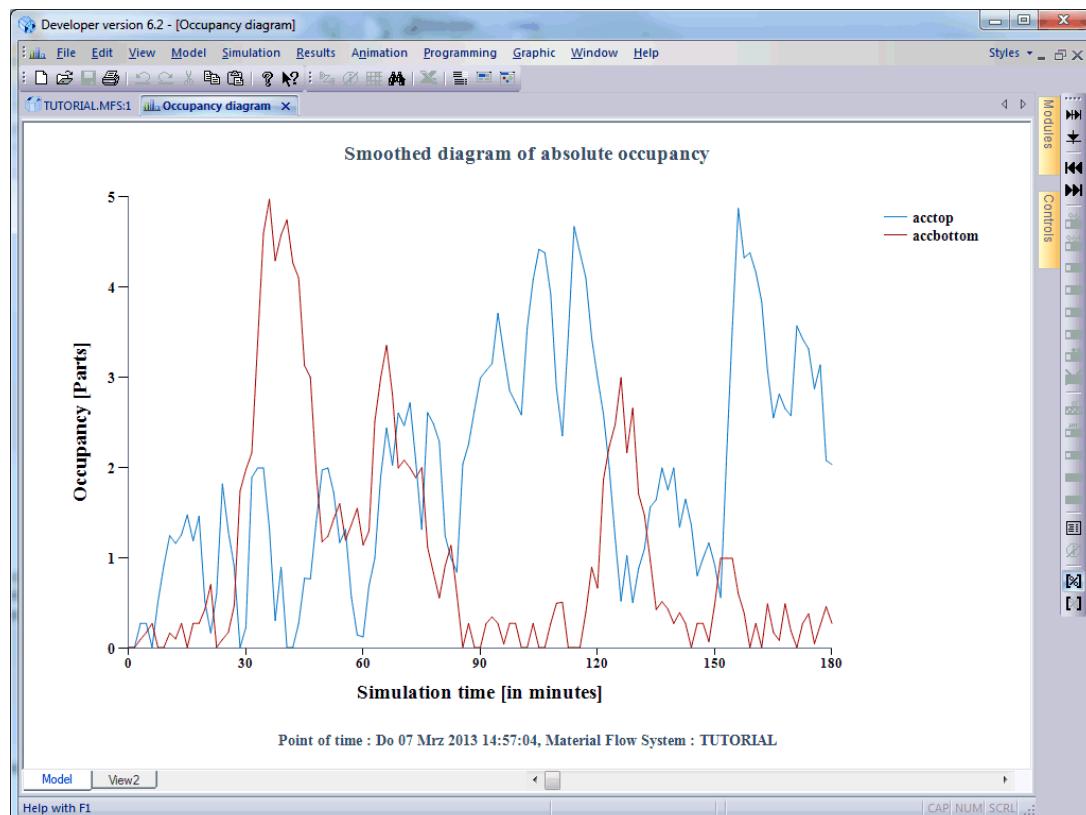


Figure 11.38: Smoothed diagram of percentage occupancy



11.8.1.3 Absolute Occupancy Diagram

The occupancy diagram allows a comparison between the absolute occupancy of different elements. The highest capacity of all elements determines the scaling of the y-axis.

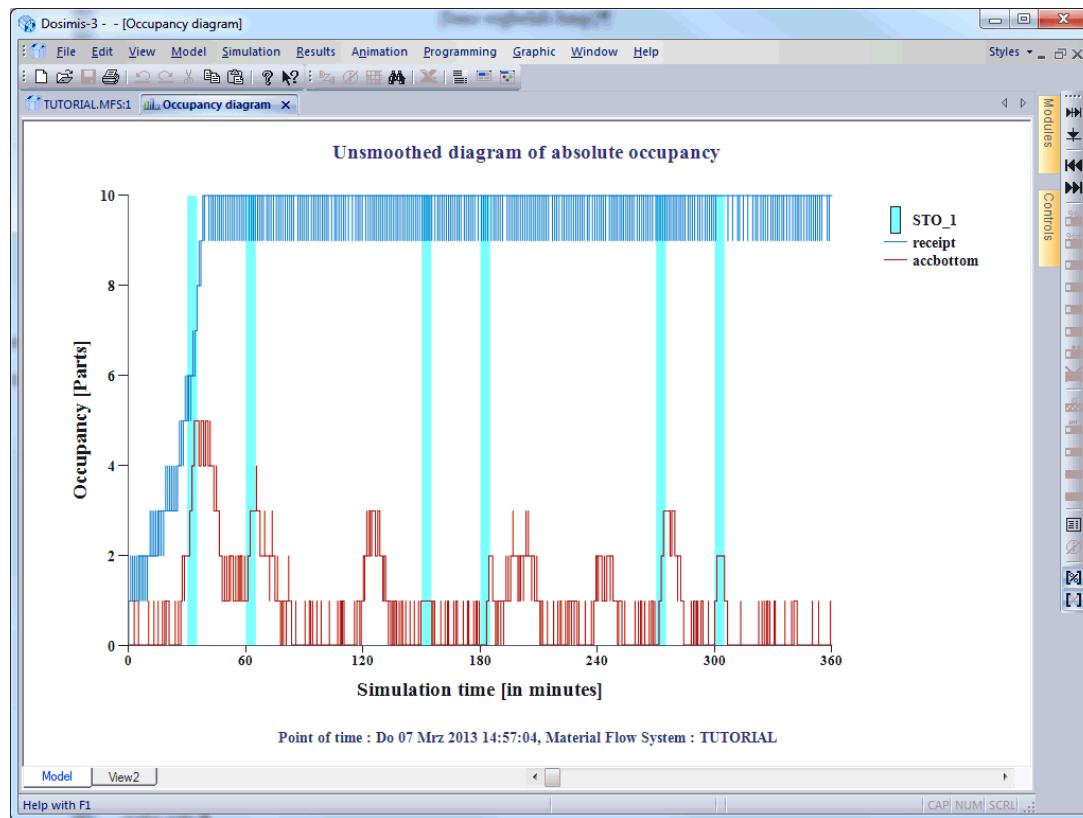


Figure 11.39: Unsmoothed diagram of the absolute occupancy with activated additional information

In the upper diagram, the usage of additional information in combination with failures is demonstrated. This will occur if the switch *Additional information* is on and the failure with *defined* or *periodical* activation time is selected.

When the switch *Show values* is selected, the mean value of the selected interval is shown on the right side.

As for the percentage occupancy diagrams, it is possible to smooth them as well.



11.8.1.4 Allocation Proportion

The allocation proportion summarizes the periods, in which the elements had the same allocation. This is displayed at each statistics point in time. For this, the internal message log (trace file) is analyzed. To analyze the element for this diagram, it must be added into the trace list.

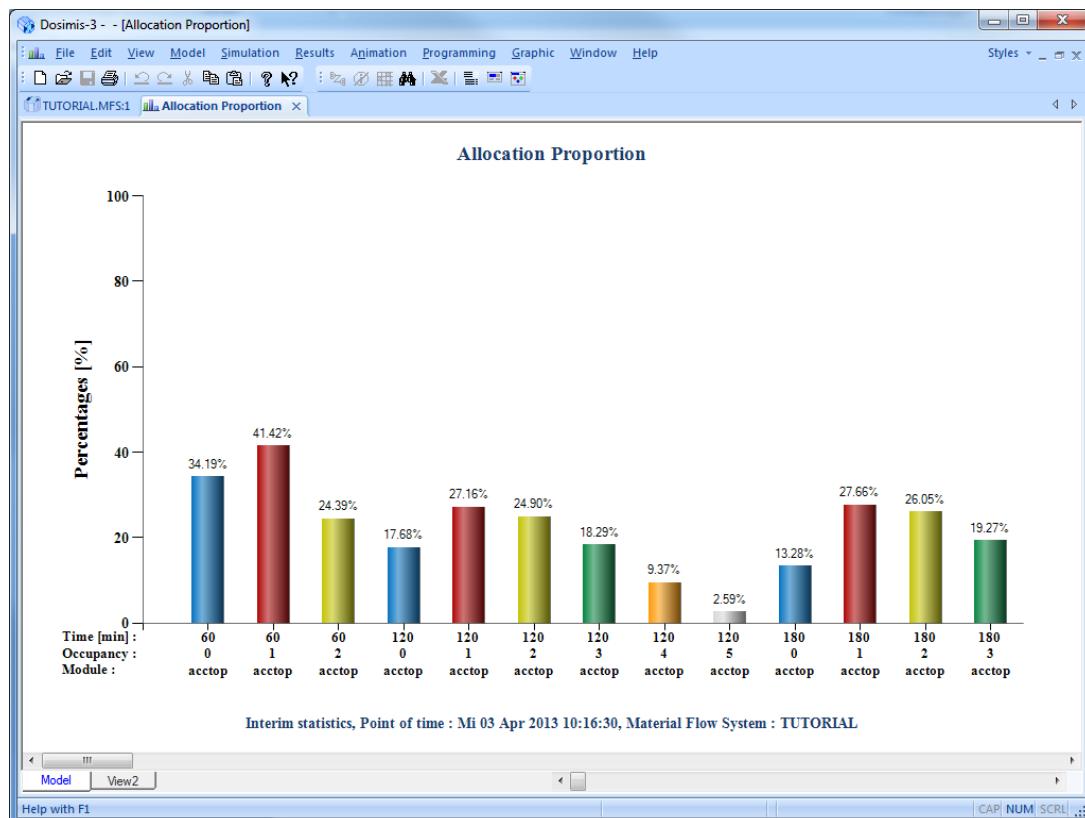


Figure 11.40: Allocation Proportion



11.8.1.5 Allocation Proportion (compact)

If the capacity of the buffer is not too large or the allocation does not vary too much, the allocation proportion can be compactly displayed in one block per element and point in time.

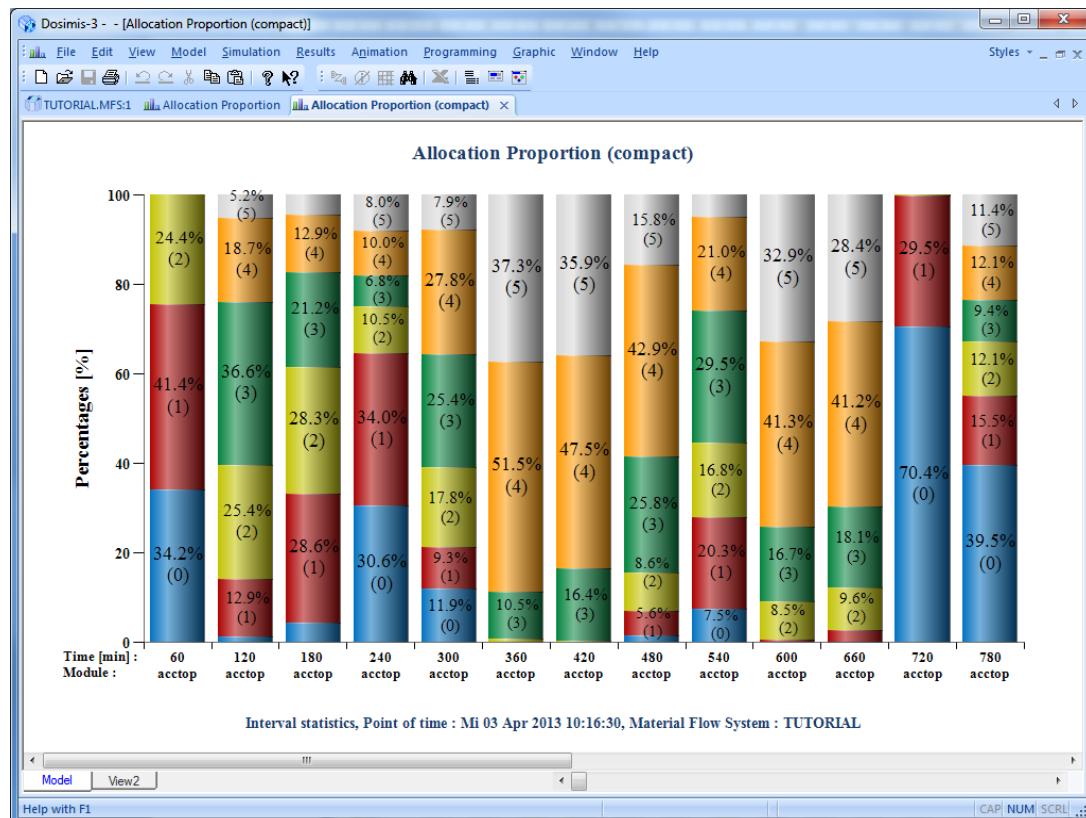


Figure 11.41: Allocation Proportion (compact)



11.8.2 Cycle Analysis

Under the term cycle time analysis, different evaluations are summarized which analyze events in modules more exactly. The distances from two events are determined and plotted. As events, the two predefined alternatives *entrance* (into the module) and *exit* (from the module) are available. In the third alternative the results can be configured with the result parameters.

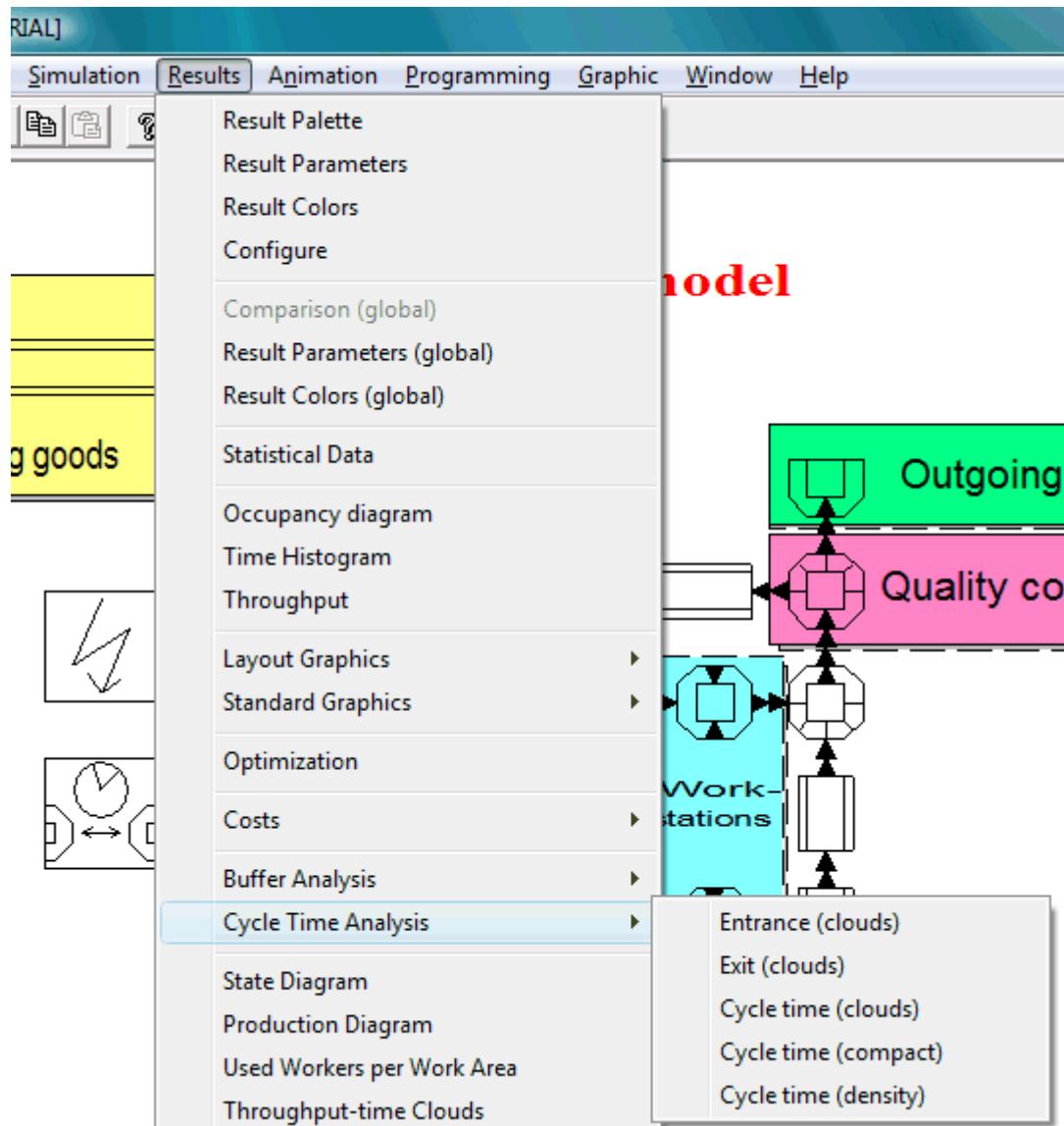


Figure 11.42: Menu „Cycle measuring“



11.8.2.1 Single Measuring (Clouds)

The individual measuring visualizes the distances from two events in a module. On the x-axis the simulation time, when this event was measured, is to be read. On the y-axis the period, which elapsed since the last event, is displayed.

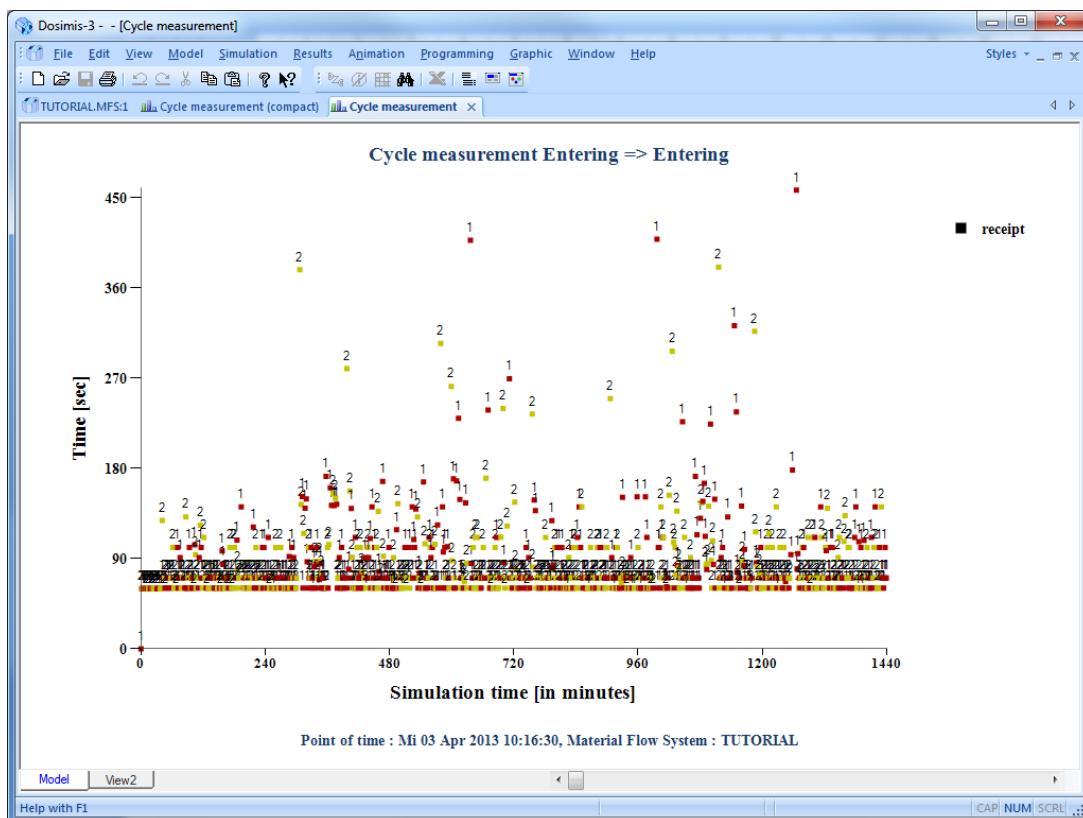


Figure 11.43: Single measuring of the cycle-time analysis

Furthermore, the display range can be limited by the input of the period in the result parameters, which can be indicated. By the delimitation of the object types, the analysis can be accomplished exactly based on object type.

Improving the Graphic

X-axis	The x-axis can be scaled arbitrarily.
Y-axis	The y-axis can be scaled arbitrarily.
Mean value:	The average value of all entries is shown as reference line.
Indiv. per page:	The scaling of the y-axis results from the min and max values of the data within the visible range. If the average value is shown, this is formed only over the data within the visible range.
Object types:	The measured values can be limited with the lower and upper limit of the object types.



11.8.2.2 Compact representation

If the measured distances do not vary too strongly, the single representation can be consolidated. For this, the interval between the minimum and the maximum value is divided into 50 subintervals. Subsequently, for each interval the number of values which lie there is calculated. In the diagram for each interval, the relative portion and the absolute portion of measured values are represented. The last value of the data indicates the upper limit of the interval. With the help of the smoothing factor from the result parameters, the number of 50 subintervals can be varied.

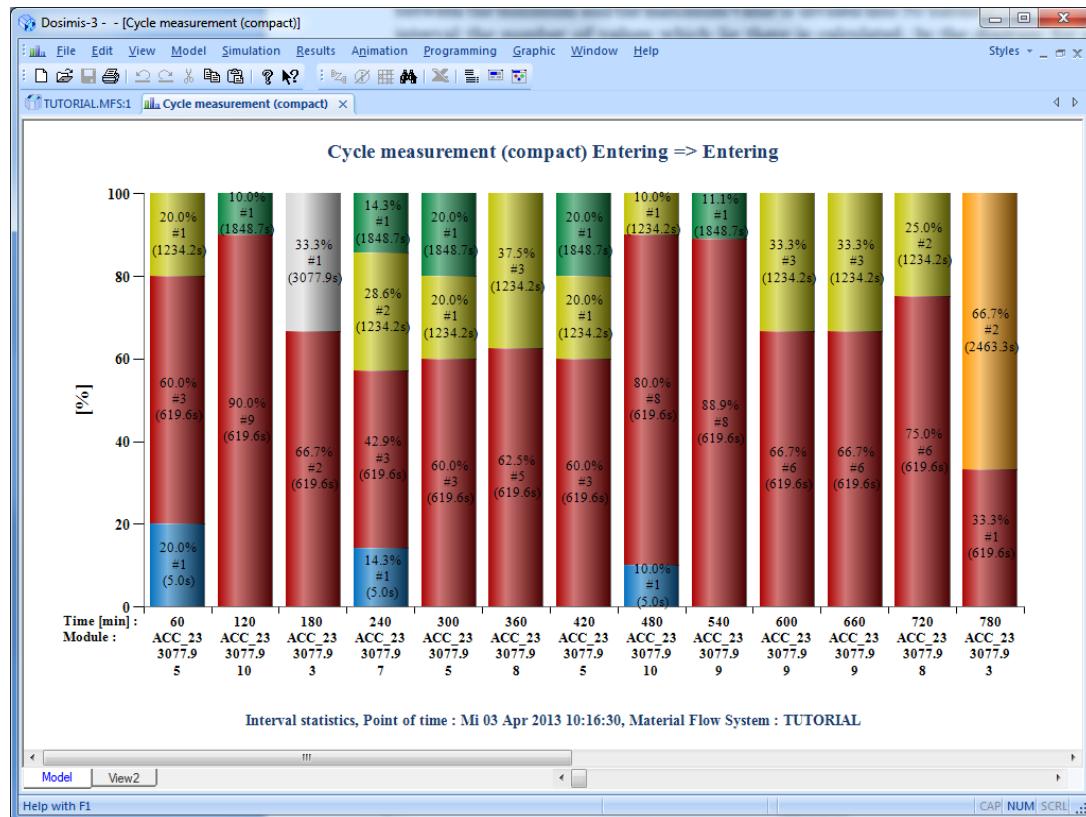


Figure 11.44: Compact representation of the cycle time analysis



11.8.2.3 Density Function

During representation of the density function the interval data of the compact representation are displayed somewhat differently. On the x-axis the interval end is shown, and in the y-axis the proportional portion of the values, which belong to this interval, is represented. The different colors correspond to the statistics times. With the help of result parameters this can be limited to one statistical point of time.

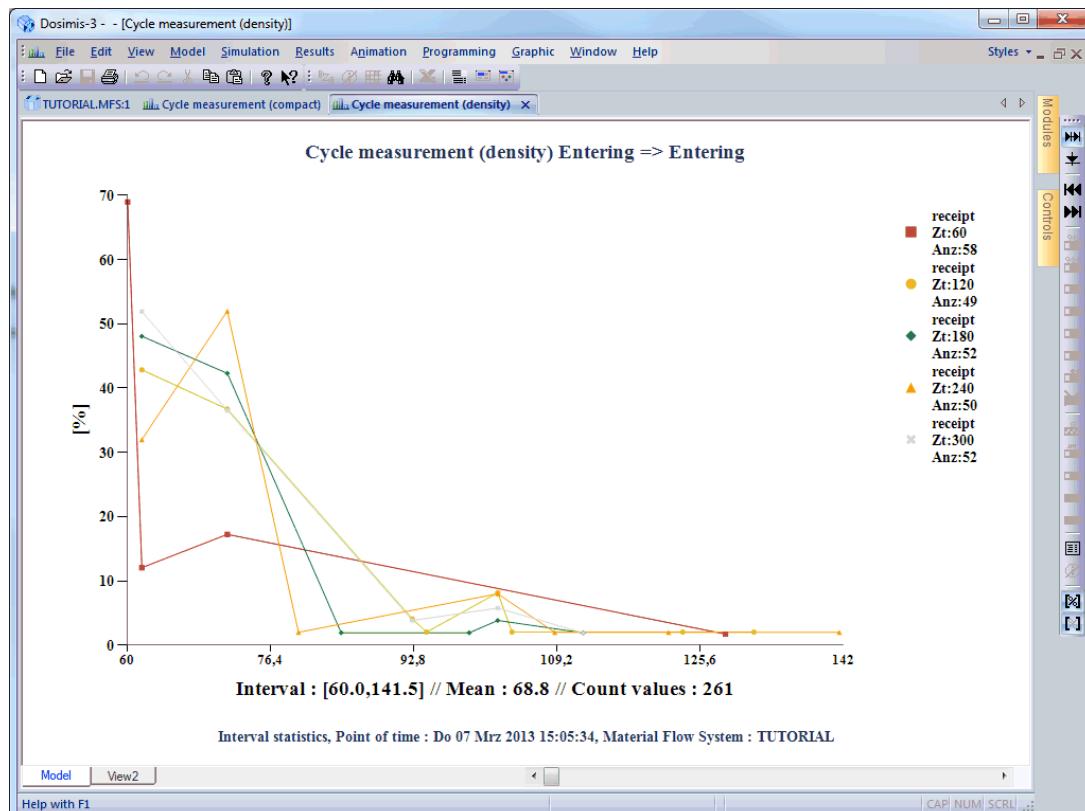


Figure 11.45: Density Function of the cycle time analysis



11.8.3 State Diagram

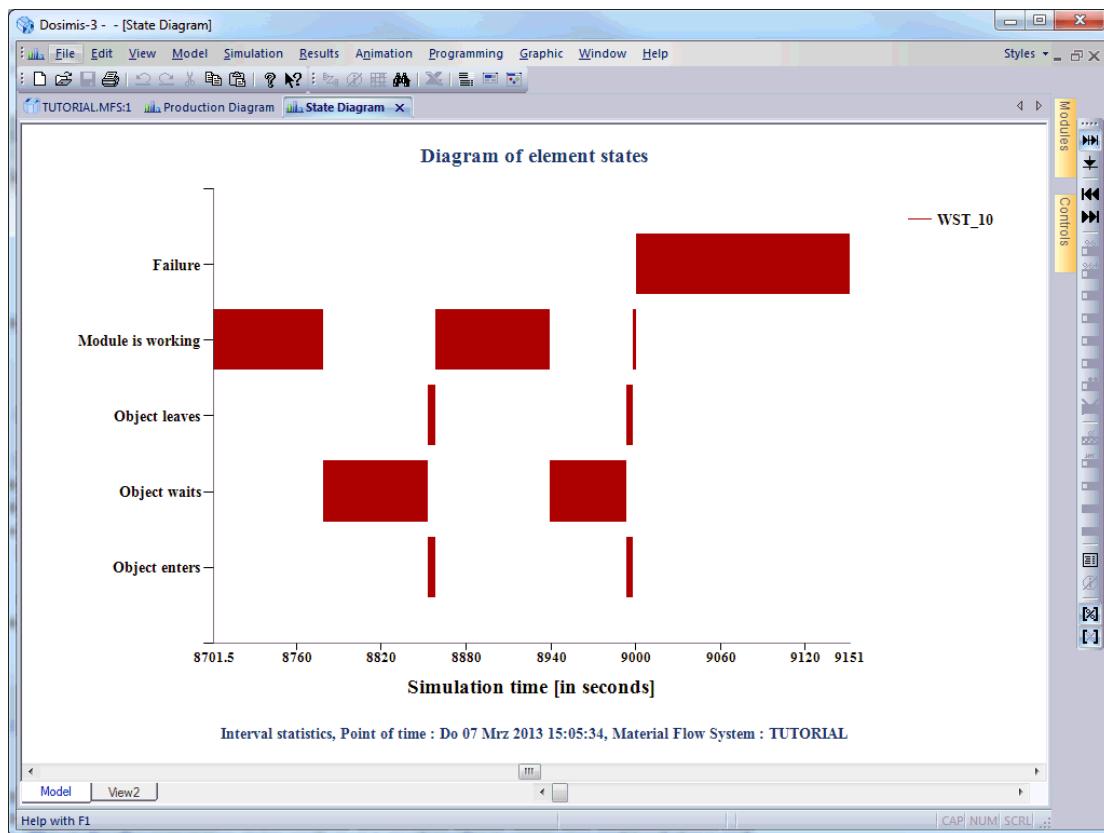


Figure 11.46: Status diagram for a workstation

With the aid of the status diagrams, processes within a module as well as between two modules can be analyzed. The possible states are represented here via the time axis. This is for the conveying states such as *run in of an object*, *empty run* and *loaded run* and work-specific states such as *wait for worker*, *working and set up*. Please note that several states can enter modules simultaneously without forward control or with capacity larger than 1. In such a way, object arriving and object exit occurs in general simultaneously.



11.8.4 Production Diagram

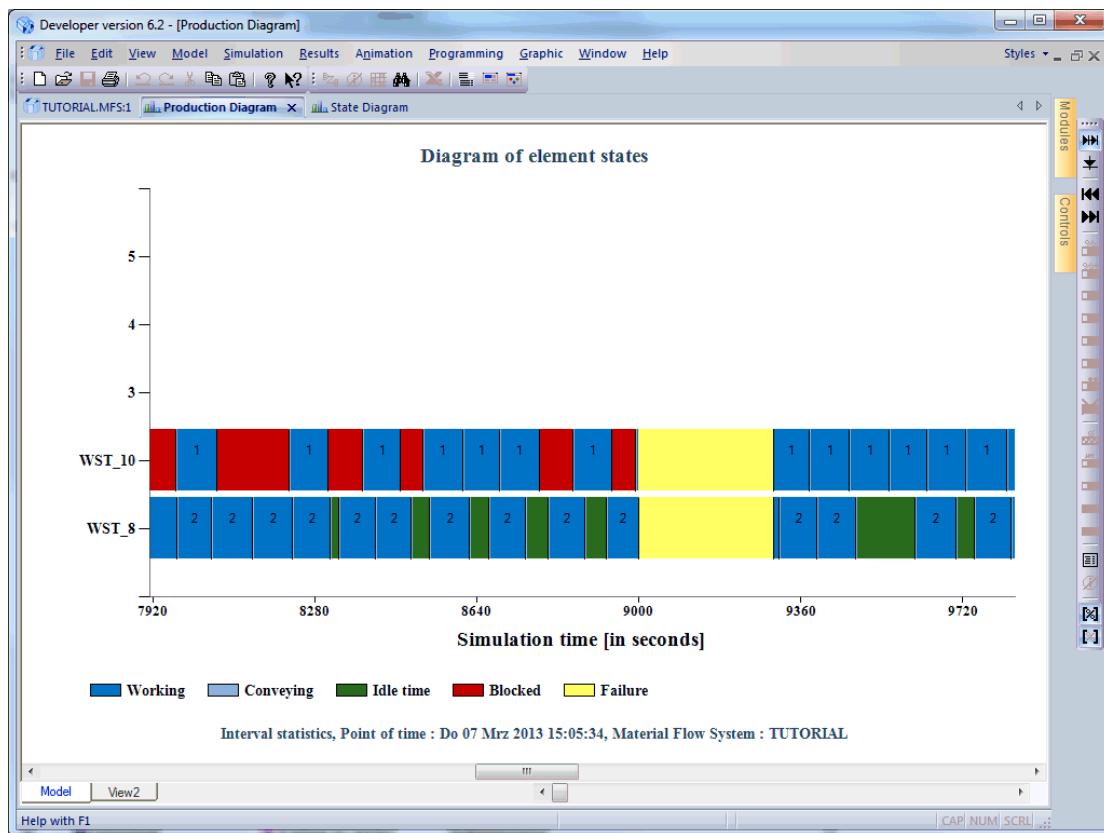


Figure 11.47: Production diagram of two workstations

With the aid of the production diagrams processes, can be analyzed within an element as well as also between two elements. Here the different states of the components are in a line represented in different colors. If the area is big enough for the processing, the types of the objects worked on are additionally indicated here.

11.9 Discrete-Time Histograms

Discrete-time histograms are based on the data in the statistics file. Therefore only the data, which are available in the statistics, can be represented. This applies both to the statistics points-of-time and for interval statistics. All data can be found in an additional statistic of the statistics file.

If *Interval statistics* is selected in the result parameters, display occurs at the end of interval statistics, whose time is specified.

The display follows for the currently selected modules/work areas. This is, if necessary, reduced to the modules of the acceptable type and modules, for which the additional statistics are carried out.

The display of all histograms can be parameterized via the result parameters *Interval statistics*, *Header*, *Footer*, *Show values* and *Hatched rectangles*. With exception of the module histogram, all diagrams are displayed for the display time period.



If further parameters additionally influence the display, they are to be taken from the individual statistics descriptions.

If more data are available than what can be displayed, then the displayed window can be shifted to left or right with the aid of the horizontal scrollbar.

All result diagrams can be transferred by the export functions (SHIFT+B, SHIFT+W) as a diagram into the clipboard. Furthermore the CSV format is available for these diagrams; it is carried out by the combination of keys CTRL+C.

11.9.1 Histograms of special Module Types

By means of block histograms, module histograms and the breakdown of the simulation time into percentage parts of type-specific module status can be shown in graphic form. The user can then compare several modules or several times.

Only modules of the same type can be compared. Modules with work (workstation, assembly, etc.) are handled like one type. If several modules of different types are selected, only the modules detected during selection as being of the first type are represented. These histograms are available for workstation, assembly, disassembly, turning table, shuttle and paired shuttle.

11.9.1.1 *Module Histogram*

In this display method, histograms of several modules of a permissible type can be shown in one picture. The type of modules found as the first in the list of selected modules is represented.

With the menu **Module Histogram**, the user receives a histogram of the percentage state parts for several modules of the chosen type. Here, the statistics of the modules with processing (workstation, assembly, etc.) can be shown together; for other modules (shuttle, turning table, paired shuttle), they are shown individually for each module. Each of the module-specific states shown in the additional statistics is displayed with a color, from which the time percentage of this state in proportion to the total statistical time period under observation can be read.



The following histogram is for the two workstations of the model example recorded in the additional statistics:

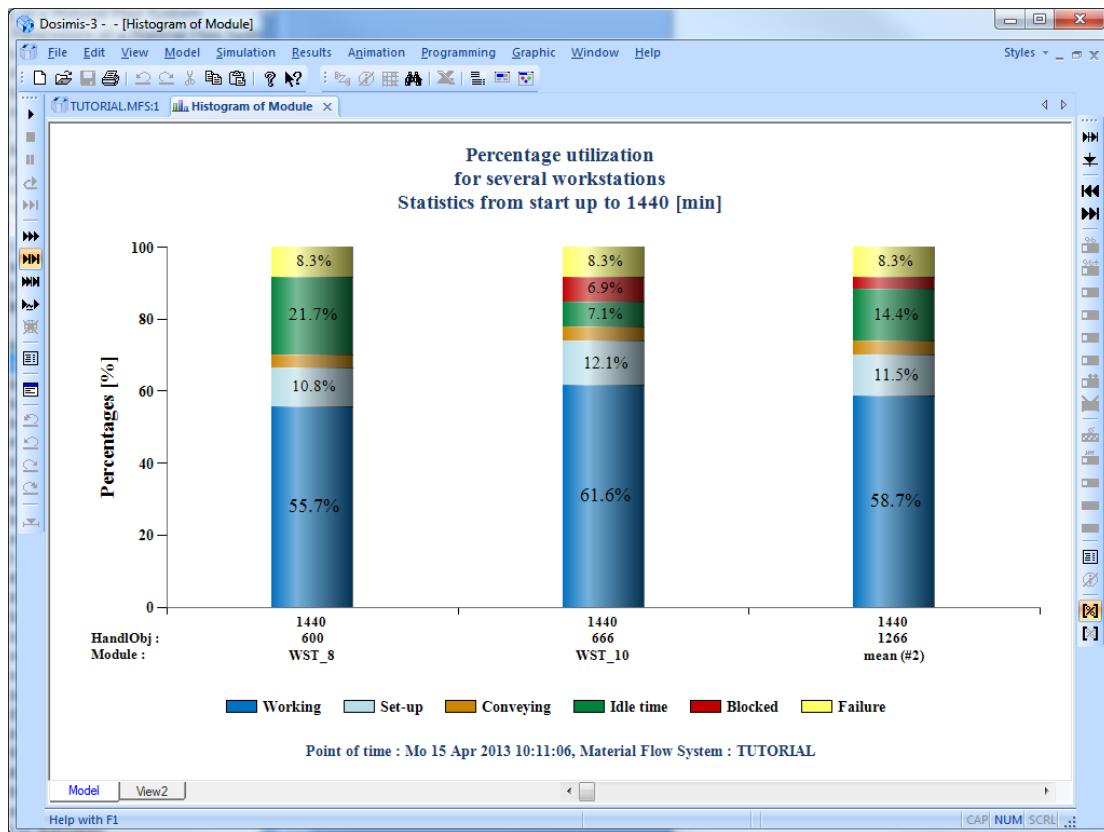


Figure 11.48: Module histogram

The histograms for the other modules are basically the same as the above example. They differ only in the module-specific meaning of the legend.



11.9.1.2 Time Histogram

If the user chooses to compare several different points in time, he must select the modules and, via **Results/Time Histogram**, open a window for the modules with a histogram with the status distribution of the modules at up to ten different times to be defined in a list.

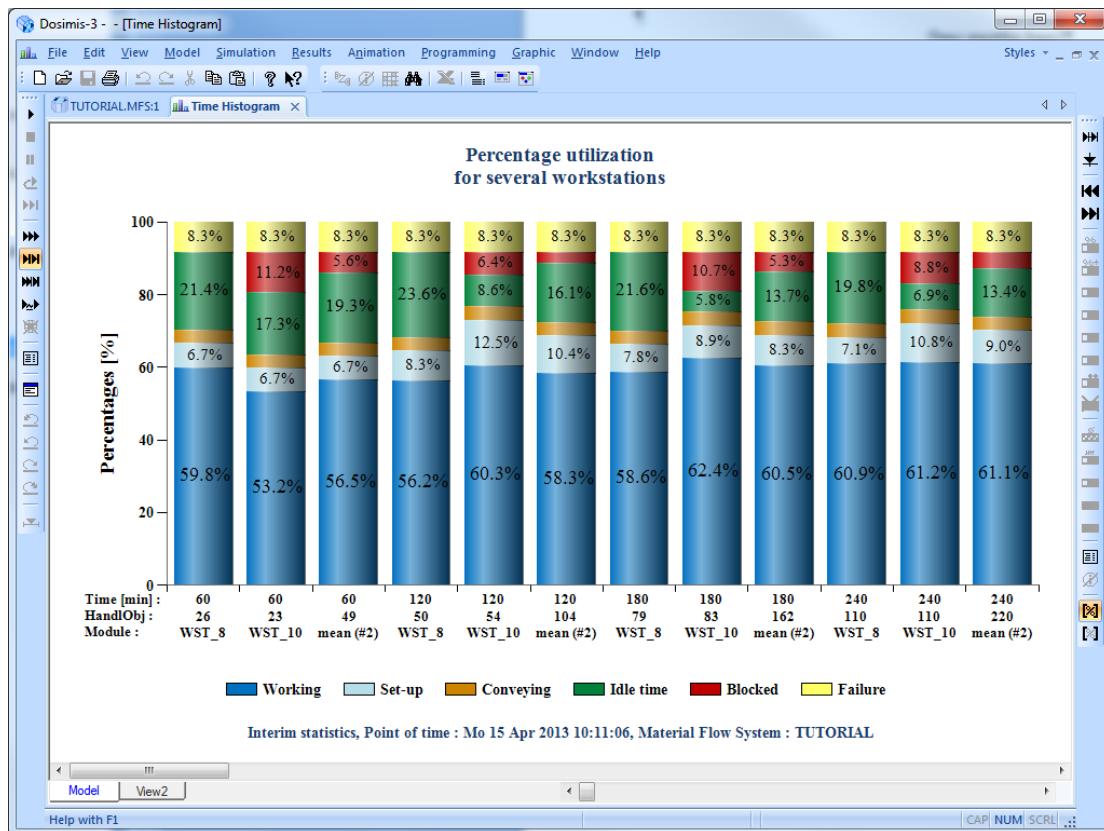


Figure 11.49: Comparison of several points in time

The importance of entered times in turn depends on whether *Interval statistics* was selected. In the latter case, the histograms for the intervals, which end at the entered time, are generated, otherwise, the entire statistics period of time is considered until the respective time.

If *Mean value selection* selected in the result parameter, the mean value is represented in addition to all elements of a statistical point of time. These blocks are marked with MEAN instead of the module name.

The data are located in the ***slg-file** below the module statistics categories within the module-type specific statistics



11.9.2 Throughput Histogram

The throughput histogram gives the user the option of observing the object throughput of a certain module. To do this, the module to be observed must first be determined (Select module). Only those objects that are delivered in the selected interval are considered.

The identifier of the selected modules appears additionally in x-axis marking. Object types and statistic periods can be reduced with the aid of the result parameters HIDD_ERG_AUSWAHL. The peak value of the y-axis results from the maximum value at the statistics period. This can be set to the maximum value with Adjust page format for the display period.

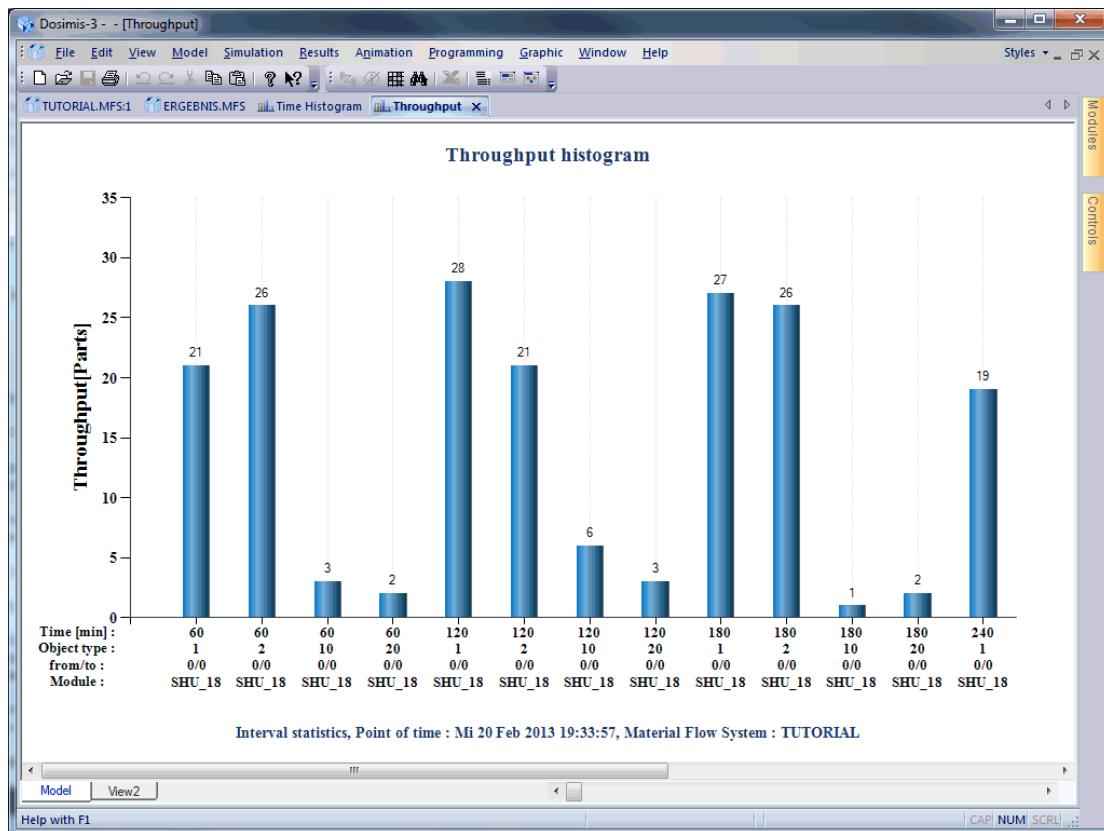


Figure 11.50: Throughput histogram

By means of the ordinate, object throughput can be recognized for the respective object types

If more data are available than are displayable in the window, then the x-axis can be shifted with the help of the horizontal scrollbar.

For additional information, the different bars are marked in color depending on the type of object. The consideration of the starting-destination relation as well as the type of object can be altered by the result parameters.

The data are located in the *slg-file in the additional statistics of the respective modules.



11.9.3 Throughput Time Histograms

11.9.3.1 Total Throughput Time

In the throughput-time histograms, the minimum, average and maximum throughput times are displayed in the form of a graph. The throughput time of an object begins at that moment when it is generated in a source or is created during disassembly. It ends as soon as the object leaves the material flow system either in a sink or in an assembly. If the object type changes, such as in a work station, the previous throughput time is transferred to the new object so that in the histogram an object is accounted for as the type it is at the end of the throughput time. In a throughput time histogram the minimum throughput time is shown by a blue bar, the average throughput time with a green bar and the maximum throughput time with a red bar. The following figure shows such a diagram.

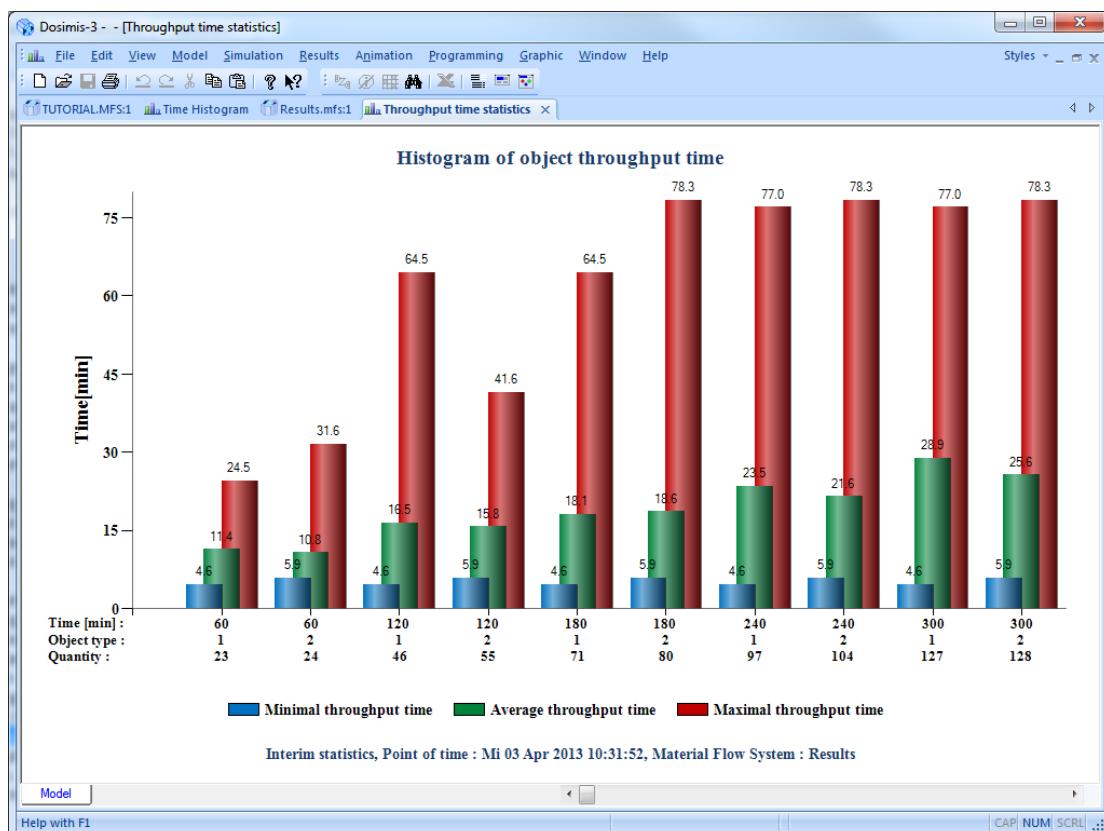


Figure 11.51: Histogram of throughput time

The x-axis shows the object type under consideration together with the number of objects which have completed the run and the y-axis shows the corresponding time. As in the case of throughput histograms the scale for the y-axis can be either fixed or dynamic. Here, dynamic means that the graduation of the y-axis is adjusted to the respective section while a fixed scale the graduation remains the same for all sections.

As with all the other histograms, a choice can be made between interim/final statistics and interval statistics. It is to be observed that by the evaluation of the .slg file only those objects are taken into account that leave the model within the selected time period.



The maximum value of the y-axis results from the maximum value in the statistics period. This can be set to the maximum value in the display period with *Adjust page format*. By setting *Start/End [obj.type]*, the quantity of the displayed object types can be limited.

The data are located in the ***slg-file** in the additional statistics of the modules.

11.10 Further Results Graphics

Dosimis-3 supports further result graphics, which are adapted especially to the controls. The description can be found in the chapter of the control

Work Area
Transport System
Shop Floor Control
Throughput Time Measurement
Measuring Element
Optimizing

11.11 Control of Diagrams

From DOSIMIS-3 version 6.1, the result graphs are drawn in a new and improved presentation. The following sections provide the control options for the user. These are on the one hand already well-known and on the other hand new added opportunities.

11.11.1 Scrolling and Zooming

Scrolling and zooming is still supported by the menu **View/Zoom** and the associated keys. An overview of all the options provided in the following table.

Process	Action
(Shift +)Cursor left / right	Scroll by line left / right
Mouse wheel down / up	
Ctrl + Cursor left / right	Scroll by page left / right
Screen up / down	
Pos1	Scroll to the beginning
End	Scroll to the end
Ctrl + Screen up / down	Zoom In/Out
Ctrl + Mouse wheel up / down	
Select section with pressed left mouse button (Pressing the escape key before releasing the mouse button stops the operation)	Zoom marked section

Table 11.1: Scroll- und Zoom keys

When the Units of the X-axis are changed from "Auto mode" to a fixed unit of time (eg minutes), you cannot zoom with any depth into a diagram. When the unit of time is minute, the minimum interval length is 15 seconds.

If this length is exceeded, a finer zoom level is prevented and the user gets a warning in the status bar: "For the selected X-axis unit cannot be zoomed. Change the units to auto mode or choose a smaller unit." After changing the time unit into a smaller one, or after a change in the Auto mode, a finer zooming is possible.



11.11.2 Filter Data

Time histograms automatically filter conditions that do not occur. To increase the clarity, only states are included in the legend, which are also relevant.

Continuous diagrams, such as occupancy diagrams, individual data series can be faded in or out on demand. Click with the left mouse button on the name of the element in the legend name of the respective series. Especially in a superposition of data series, you can thereby increase the clarity without a completely recreation of the chart.

11.11.3 Caption

If data labeling is activated in the result parameters, in some result representations the values of the basic data is labeled. If not enough space for the label is available, this remains out, however. By clicking on the color or in the legend, the labels for all these values are activated. Only the clicked result proportion is labeled, not all others. By a repeated clicking on the color of this mode is canceled.

11.11.4 Call of Dialog Result Parameter

The result dialog can be reached via the menu **results/result parameter** as usual. Additionally, you can access the individual tabs when you double click on certain areas in the diagram. So a double click on an axis opens the respective axis dialog. Each double click within the data area, eg on a bar of a flow chart or even on background, opens the tab series. In discrete-time diagrams of a double click on the legend leads to the selection of settings. In all other cases, the layout settings are shown.



12 Animation

The animation program serves to illustrate object movement in a material flow system. The decisive advantage of the animation program is the possibility to locate a deadlock relatively quickly. Under certain conditions, it can be very difficult in a large system to find a system disruption by means of the statistics file. Animation however shows immediately in which parts of the system there are disruptions, jams or deadlocks.

By means of the menu **View** the desired section of the total model can be selected. The menu **Animation** contains the following submenus:

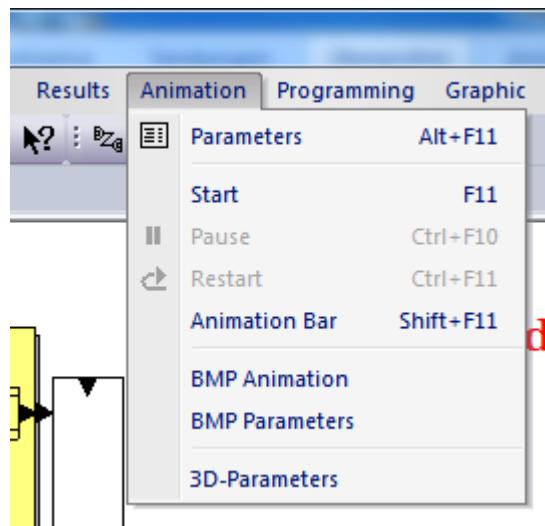


Figure 12.1: Menu list of „Animation“

With the menu **Parameters**, the animation parameters are adjusted as described below. With the function key **F11** or with the menu option **Start** you can start the animation. By pressing of **Shift+F11** or via the menu **Animation/Animation bar** the animation bar is called. The animation is started also if no adjustments were made with the animation parameters. It is then started with the default values.

When the digital display is activated, the time will be shown there. In the other case the time is shown in the right area of the status bar.

With the menu item **BMP Animation**, the bitmap animation mode will be activated. For detailed information see chapter [Bitmap-Animation](#).

With the menu item **BMP Parameters**, the bitmaps according to this animation can be scaled. Furthermore, the definition of substitute object numbers can be made.



12.1 Animation Parameters

For each simulation model, several sets of animation parameters can be assigned. This takes place with the navigation button in the lower left corner. With the help of the selection box, it can be switched between different sets of parameters. The vertical arrows serve to shift the data record within the list. The first data record is always the one that is active when opening the model.

12.1.1 Standard Parameters

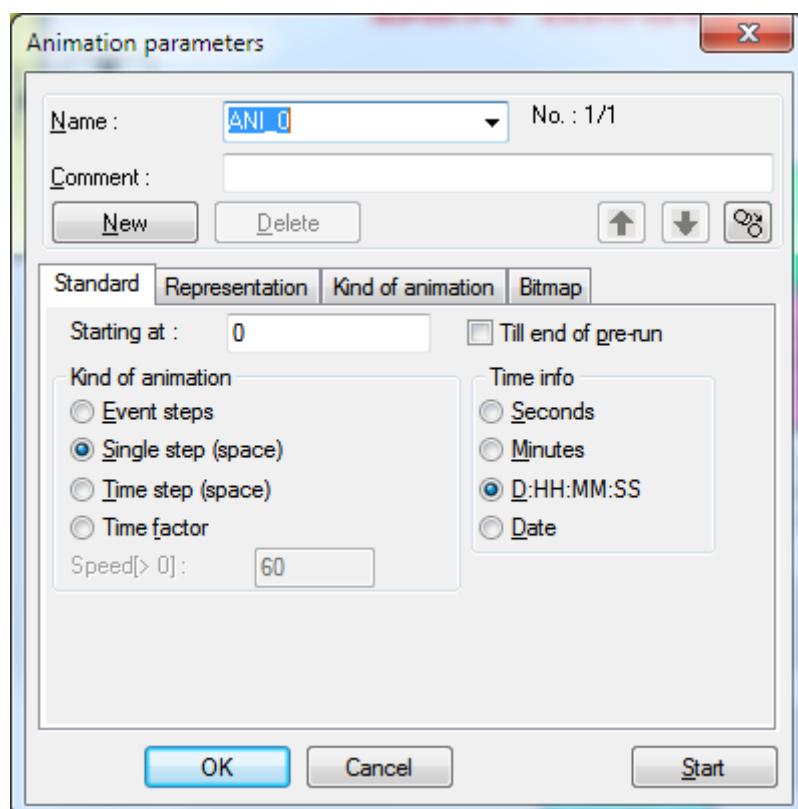


Figure 12.2: Animation parameters

The user can determine by *Animation/Parameters* the starting time of the animation, animation type and format of the time indication. The whole number entered as a starting time corresponds to the minute, until which the animation actions are forwarded without actual display of events. If the time is chosen as seconds, the entered time is also counted in seconds.

The different kinds of animation mean:

- | | |
|---------------------|--|
| Event step: | All events are represented as fast as possible. |
| Single step: | Every event is represented separately and must be recorded. |
| Time step: | Like single-step, but all events of the same point of time are drawn. |
| Time factor: | The animation is carried out proportional to time according to the given time factor (speed) . The pre selected value of 60 makes an animation minute per second. The animation according to a time factor cannot go faster than the animation according to event step. |



The format for time indication is selectable through the four examples. One can indicate **seconds** or **minutes** or also the normal setting **D:HH:MM:SS**. *D* is the abbreviation of days, *H* of hours, *M* of minutes and *S* of seconds. In the status bar, the time which indicates the current time of the animation run is displayed. In the representation **Date** the display is D:HH:MM:SS, but a short form of the day of week is also shown. Day 0 means Sunday, 1 Monday, ..., 7 Sunday again and so on.

The display of the simulation time takes place in the lower left corner of the status bar. This shows the actual point of time during the animation. When the digital display is activated, the output of the time will take place there.

12.1.2 Representation

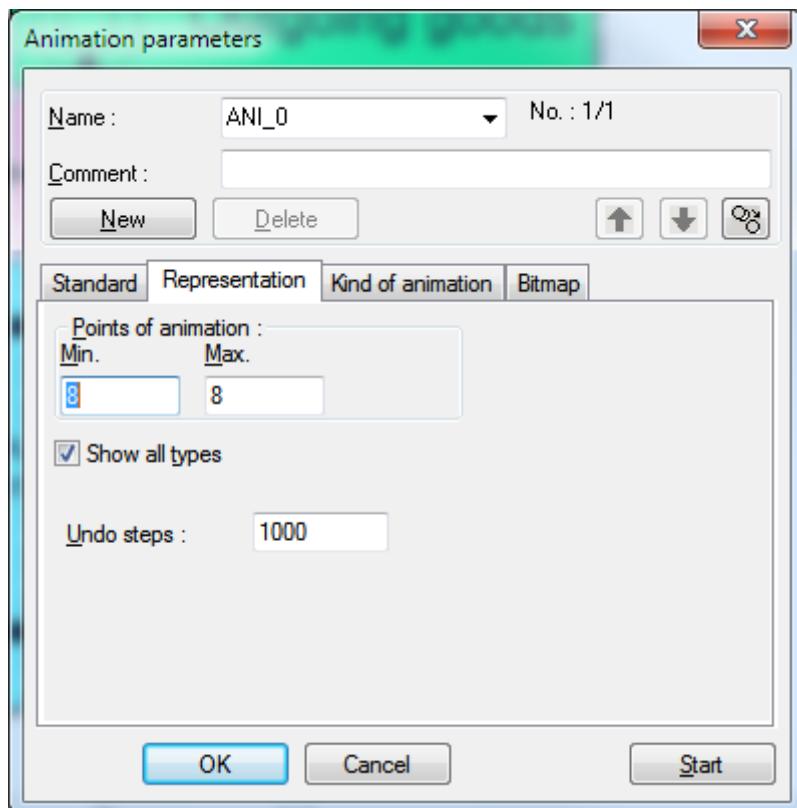


Figure 12.3: Animation parameters

The **Points of animation** specify how these are distributed on conveyors. If the minimum and the maximum value are set to 8, the animation boxes lay side-by-side; an increase of the maximum value leads to an even distribution of the points on the entire distance, if this is possible in the display. The reduction of the minimum value leads to an overlapping representation, if the representation is too short relative to the capacity of the conveyor. If not enough space is available for all points, further objects are animated all at the end of the element.

If **Show all types** is activated, all objects in the conveyors (e.g. accumulation conveyor or bulk section) are indicated with numbers, otherwise only the first object in conveyors is provided with a number. With that, the speed of the animation can be increased.



12.1.3 Kind of Animation

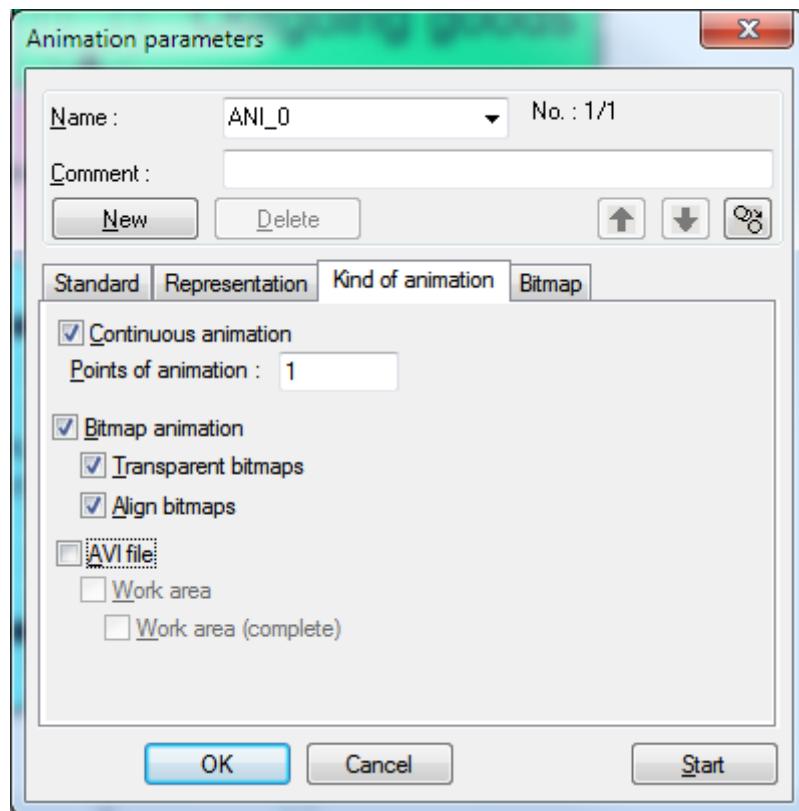


Figure 12.4: Animation parameters

The switch **Continuous animation** activates a separate animation mode. The objects on transportation conveyors are correctly drawn at their last position and are transported forwarded during animation. Since this mode is quite time-consuming, it should be used only for presentation purposes. Further [information](#) will be found below.

The switch **Bitmap animation** lets you toggle between bitmap animation and standard animation. Switching is possible only if the animation does not run.

If the switch **Transparent bitmap** is activated, the pictures with the bitmap animation are represented transparently. Otherwise the complete representation of the bitmaps takes place. It is to be noted that the transparent color is defined by the pixel in the upper left corner.

Align bitmaps means, that the bitmaps are rotated according the direction of the module. Otherwise the bitmaps will not be rotated.

If the switch **AVI file** is activated, the animation can be recorded. A file *modelname.avi* is produced, which can be played like a film with the Windows Media Player.

Depending on the configuration of the computer, different recording algorithms (codec) are offered at the start of the animation. The following sheet compares some of them:



Codec	Duration	Size	Remark
Fullsize	100%	100,00%	out of focus
MS Video 1	110%	16,16%	out of focus
MS RLE	91%	9,65%	out of focus
MS MPEG4 V2	107%	0,53%	well (slight shades)

Figure 12.5: Alternative recording methods

When some different methods have been installed on the target computer, they are offered at the start of animation.

If the switch **Work area** is activated, only the work area is used as film canvas. Otherwise the recording takes place including the window framework and the menus.

With the choice of the switch work area can be additionally still selected, as this range is recorded. If **work area (complete)** is active, all elements are recorded, which are within the work area (e.g. the digital display, module pallet, ...). Otherwise only the modules and controls are considered.

12.2 Animation Menu

A running animation can be interrupted by the menu item **Pause**. The animation stops at the current situation. It is then possible to either **Restart** the animation, to continue (Pause) or to finish (Stop) the animation.

After starting animation in single-step mode, the user has two items at his disposal, namely **Stop** and **Next step**. Alternatively he can press the spacebar for this operation.

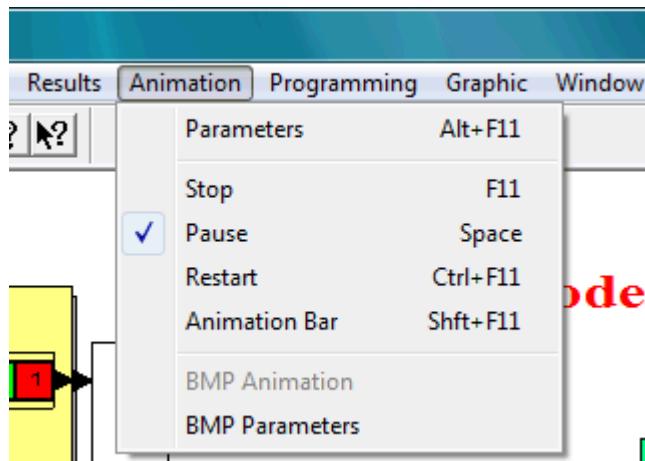


Figure 12.6: Animation parameters during current animation

The animation can be suspended by pressing of the spacebar and/or be continued in the case of single-step animation step-by-step.

Interrupting or stopping is also possible with the aid of the **Animation bar**.



12.3 Animation States

12.3.1 Modules and Objects

Individual objects are shown in the form of rectangles, which are given a number corresponding to the respective object type as far as a sufficiently zoomed section has been selected. The color of the rectangle shows the present state of an object, whereby the following apply:

	Green	= moved object
	Red	= blocked object (Remark: is only one of several possible situations that lead to a blockage in terms of statistic items. Additionally the times waiting for the last object leaving the module are e.g. blockages as well.)
	Blue	= processed object
	Light blue	= set up
	Yellow	= wait for worker
	Violet	= wait for worker for set up

The **state of the first object** is only represented in a section. All other objects remain green without specification of the object type. If the number of the positions of a route is not sufficient to represent all objects during the graphic representation, the count of the current occupancy is indicated in the last represented segment.

With the **exit of an object**, the frame of the first position appears in red till the object is pulled in completely into the next module.



12.3.2 Disturbed Modules

Disturbed modules are shown colored. This occurs as a function of the type of the failure.

	Red	Failure
	Green	Maintenance
	Blue	Break
	Cyan	Failure by decision table
	Yellow	Waiting for worker
	Light blue	Cleaning

Disturbed modules are only marked, if for the corresponding failure the trace output is activated.

12.3.3 Movement

Some modules possess movement that is visualized in the animation as an animation characteristic. In the case of an **empty run in the shuttle**, the moved vehicle table is framed green. Then the triangles appear for the period of a run (empty run or loaded run) in green. The arrows in the turntable are marked green if the **turntable is in motion**. The crossing is star-shaped and green-marked for the period of the lift or switching times.

12.3.4 Conveying Circuit

The animation of the conveying circuit is analogous to those of accumulation conveyors. Here it is to be noted that the conveying circuit is closed. In order to specify the direction of the representation, this is to be considered when placing the module. The position of the first place is with the first way point toward the second way point. If necessary, the placement of the conveying circuit must be corrected. Since the operations in the simulator depend only on the parameterization of the conveying circuit, these are not affected by a changed representation.

12.3.5 Storage

When setting the parameters, the position of the storage module was fixed with a start and end point and divided into equal-sized segments for graphic illustration. In the first three



segments, i.e. those nearest to the start, data is issued during animation which affects the whole of the storage. One item of data deals with the inventory. This states the total number of stored objects in the storage and can be seen in the ‘first’ segment. In the ‘second’ segment, the number of delivery orders which have not been allotted to an area is given.

As already mentioned, a storage module consists of different separate areas, also called storage alleys, which have a separate entrance and exit. During animation each storage alley is given two numbers. In the segment nearest to the entrance, the inventory of stored objects in each alley is given. In the subsequent segment, the number of delivery orders for the alley can be read. When positioning the entrances and exits within the parameters, the following is to be observed: as the first three segments are always reserved for the issuing of storage module data an entrance should never be placed at the end of a module otherwise this data will be overwritten by data concerning the storage alley.

Furthermore, each individual storage alley has its own storage and retrieval system (ASRS) which serves to transport objects during storage and retrieval. This ASRS can be in three different states, which are marked in different colors during animation:

- **Loaded run:** a loaded run is shown in green. It can only take place from the material entrance to the storage or from the storage to the material exit.
- **Empty run:** an empty run is shown in white and can only take place within the storage alley or to the material entrance.
- **No activity:** a waiting ASRS is shown in black.

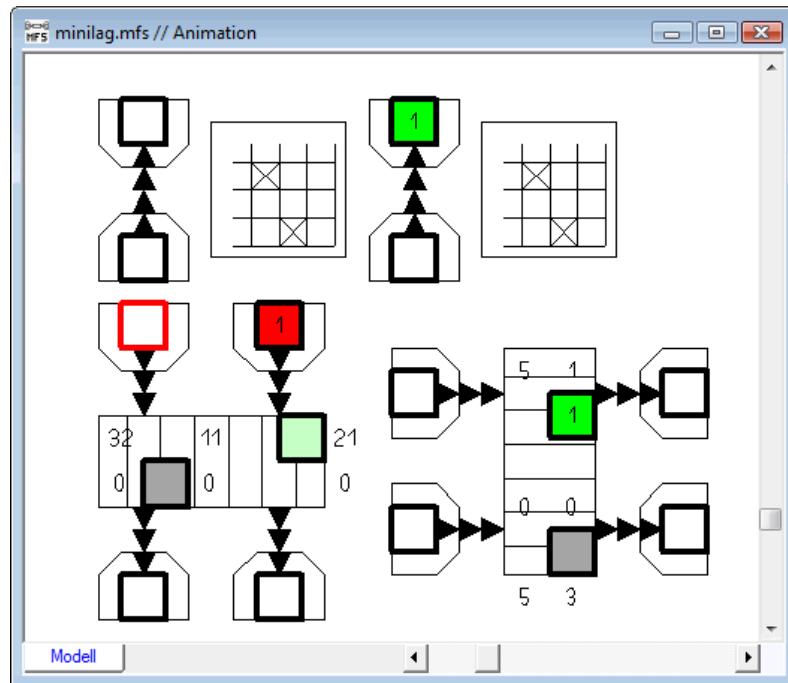


Figure 12.7: Storage animation

Interpretation of the data:

Left stock is occupied with 32 objects. 11 of them are in the left area, 21 in the right. Furthermore no delivery order is active and the waiting list is empty.

Right stock is occupied with 5 objects. All are in the upper area. Furthermore a delivery order is active in the upper area (Object 1 travels to the exit). Additionally 3 delivery order, which are not yet assigned to any area, are present (the object type is not available in the storage).



12.3.6 Junctions

Blocked Junctions are marked red.

In the case of activated junction animation, further states are encouraged. In junctions that are in the state *announced* the first triangles are drawn yellow, in the state *waiting* this occurs in green. In the state *occupied* the marking occurs through a green second triangle. So the blocked junctions can be recognized simultaneously, because the other triangle and the connecting line appear red there.

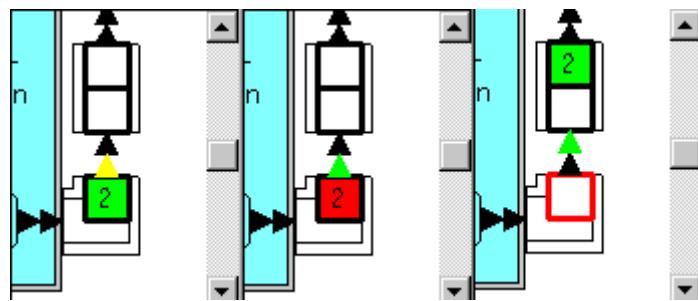


Figure 12.8: Animation of junctions (from left: announced, waiting, occupied)



12.3.7 Failures

Active failures are marked with a blue frame. This means that this disturbance even affects its items.

Disturbed failures are marked with a red frame. This means, this disturbance was passivated by another main failure. It is not differentiated whether the main disturbance is of the type break, maintenance or failure.

	Blue	Failure is active.
	Red	Failure is disturbed by another failure.

12.3.8 Work Areas

During the animation the worker positions appear in different colors. This can occur within the work area symbol or on the working places, which must be placed for this purpose in the layout. Within the set are the numbers of the workers. The numbering occurs in the order of the work areas and within the work areas in the order of the declaration of the workers. The colors have the following meaning:

	Black	The worker is idle.
	Green	Worker is on the way to a task.
	Light Green	The worker is on a break.
	Blue	The worker is active.
	Light blue	Worker is waiting for a further worker, when several workers are needed for that task.
	Yellow	Worker remains at the working station waiting for the next object

12.4 Animations Toolbar

By pressing **Shift+F11** or via the menu sequence **Animation/Animation Bar**, the animation bar is open. Hereby the animation can be influenced.



Figure 12.9: Menu items in „Animation“

12.4.1 Control

Shortcut

Toolbar:



Keyboard:

F11 / F11 / Spacebar / CTRL+F11



With the first button the animation is started. The animation is terminated by the second. By the third one the animation can be interrupted. The animation is again started by the fourth.

12.4.2 Forward

Shortcut

Toolbar:	
Keyboard:	F10

If animation by time factor is selected, then one can skip to the next event with this button.

In the case of the single-step animation, event animation is switched to until the animation is interrupted by breakpoint. Then the animation switches automatically back to single step.

12.4.3 Kind of Animation

Shortcut

Toolbar:	
----------	--

With these four buttons, you can select the kind of the animation:

- **event steps**
- **single step (space)**
- **time step (space)**
- **time factor**

12.4.4 Hide

In order to let the animation move faster, drawing can be hidden. Thus only the events are recorded and the desired point in time is reached earlier.

Shortcut

Toolbar:	
----------	--

12.4.5 Parameters

Shortcut

Toolbar:	
Keyboard:	Alt+F11

By pressing the button the dialog Animation parameters is called.

12.4.6 Log Window

Shortcut

Toolbar:	
----------	--

This last button activates the Log window of the decision tables.



12.4.7 Backward

A status with interrupted animation can be analyzed with the following four buttons more exactly. With these simple arrows, individual events can be connected backwards and afterwards also forwards. In this mode, the animation can be continued again only if all backspaces were recalled. This is done directly via the fourth button.

Shortcut

Toolbar:



Figure 12.10: Parameters of Backward animation

The number of steps is limited. The upper limit can be set by the animation parameters. When the value 0 is entered, no statuses are stored, as a result of which the animation processes a little faster. The recording starts only after achieving the adjusted starting point.

12.4.8 Save State

A once-achieved animation status can be saved by this switch. If now the animation is again started, it begins directly from here. In contrast to the advance by point of starting time, the internal message log does not need to be analyzed any longer.

Shortcut

Toolbar:



If an animation status were stored, the point of starting time is ignored when starting animation. By pressing this button again the saved status will rejected and the animation can be normally started again.



12.5 Breakpoint

The animation can be stopped by the setting of break points purposefully. The event, which is supposed to stop the animation, is determined by the context menu of the module.

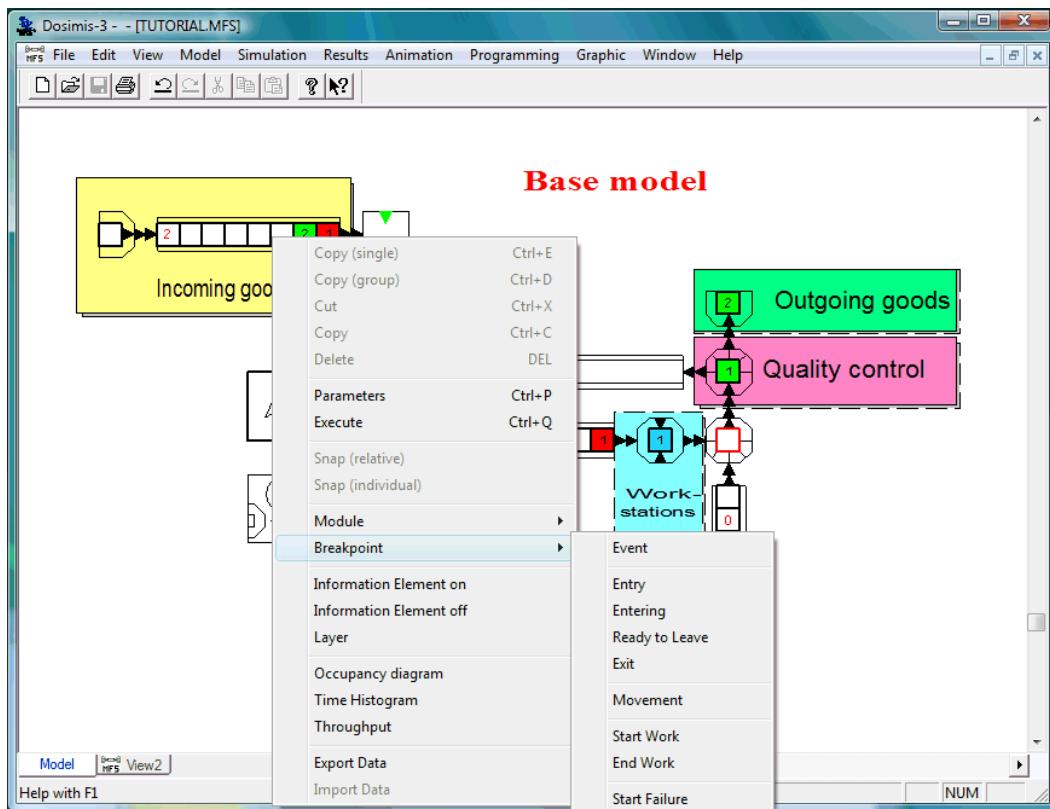


Figure 12.11: Break points in the animation

After the break, the information of the activating element is available in the variable window. Since the animation is based on the trace file, only the information which can be determined from the materials flow system or the action protocol can be indicated.

The elements equipped with break points are marked with a square on the upper left corner.



12.6 View Objects

Through pressing of the right mouse button onto a module during the animation the possibility exists to have a look at the order of the objects in that module.

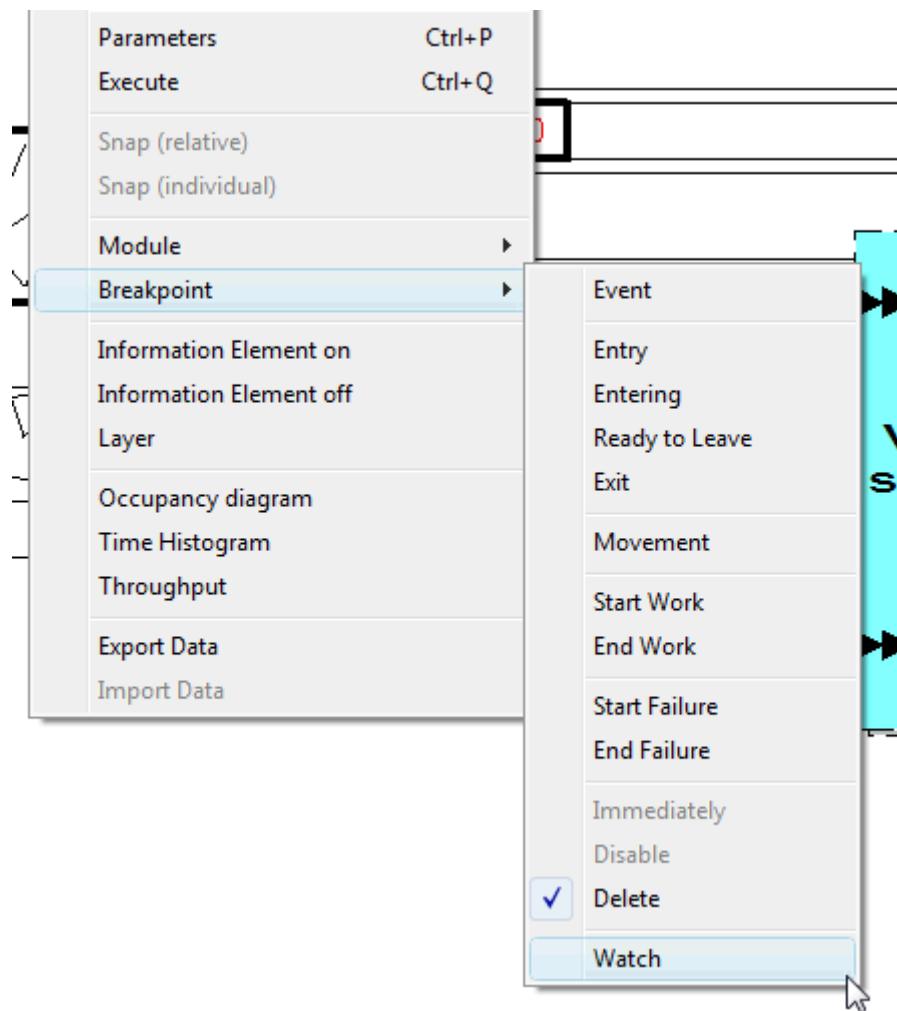


Figure 12.12: Context menu for the element object view



The display then occurs together in the protocol window with further characteristics of the selected elements.

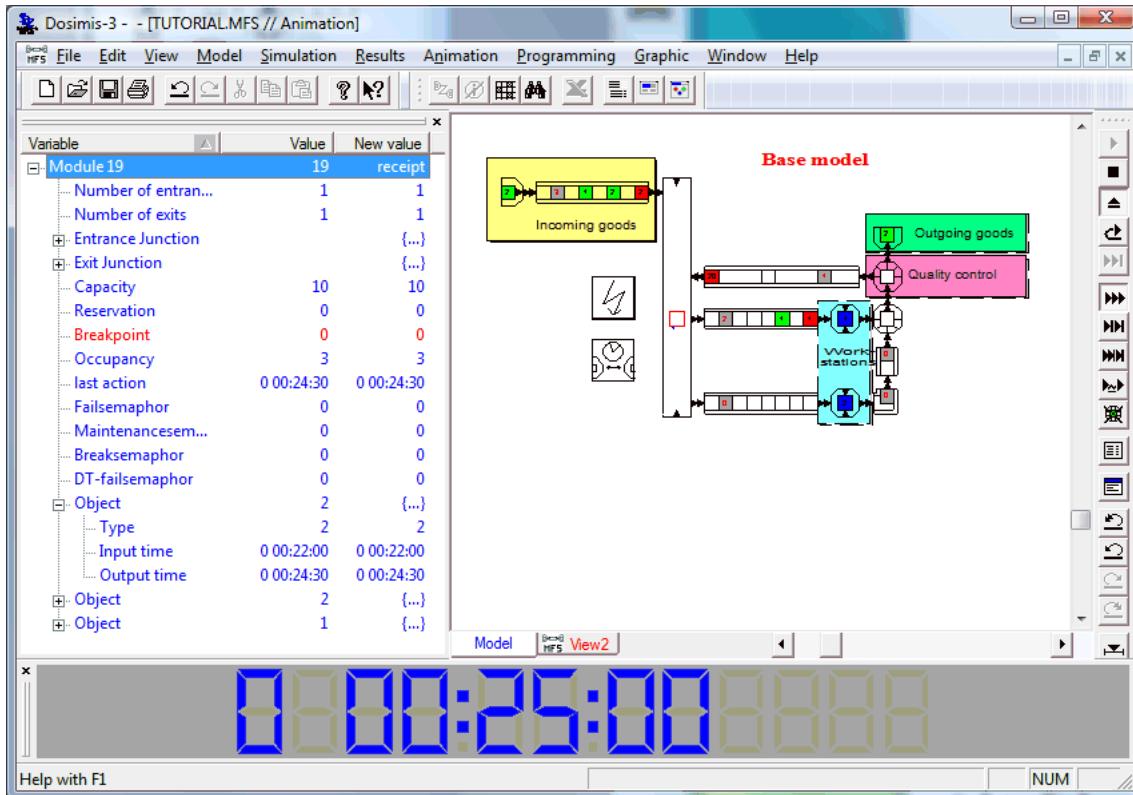


Figure 12.13: List of the objects in the component

The values are actualized with each break. Once added, modules can be taken away with a right-click on the headline of a module.

12.7 Bitmap Animation

12.7.1 Introduction

During the bitmap animation it is a question of a form of the process animation which illustrates the simulation result better through support of bitmap graphics.

In "Standard animation" DOSIMIS-3-objects are only represented by colored squares, but in bitmap animation DOSIMIS-3 objects become represented by arbitrary bitmaps. Through the use of different graphics depending on the object type one is enabled to use DOSIMIS-3 as a presentation tool to represent also complex problems simply and vividly.

12.7.2 Create Bitmap

In order to create bitmap graphics, standard graphics tools can be used. Also graphics from clipart collections can be used in general. It is to be noticed that the graphics must be stored in the Windows BMP-format (File extension .bmp). The graphics can subsequently be scaled by DOSIMIS-3. It is to be noted however that this can often lead to loss of quality.



12.7.3 Adjusting Bitmaps for DOSIMIS-3

The bitmap graphics must suffice for specific conditions: These are however scalable within DOSIMIS-3, a size is recommended however to pixel by 70x70. These graphics are represented in the layout according to the 4 directions. The calculation occurs again after each change of the view.

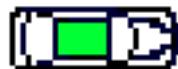


Figure 12.14: Graphic objects for DOSIMIS-3

If a colored representation of the states is required, the indices 1 to 10 of the color palette are used for that. The color of the index 1 is replaced by those of the indices 1 until 10. By this, the color indices correspond to the colors of the standard animation as shown in the following table:

1		Red	= waiting object
2		Green	= object in motion
3		Blue	= object worked on
4		Light Blue	= set up
5		Yellow	= observations on worker
8		Violet	= observations on worker for set up

The values 6, 7 and 9 are not yet occupied.

If the colored animation of the objects is not wanted, the entries of the indices 1-9 are to be assigned the same color.



12.7.4 Bitmaps for different Objects

The assignment of graphics to an object type is defined through their file name. File names follow the convention <number>.bmp here. For object type 10 the corresponding file would therefore be called 10.bmp. The file 0.bmp has a special function. If no suitable BMP file is found for an object type, 0.bmp is used. If 0.bmp does not exist, then bitmap animation cannot be started.

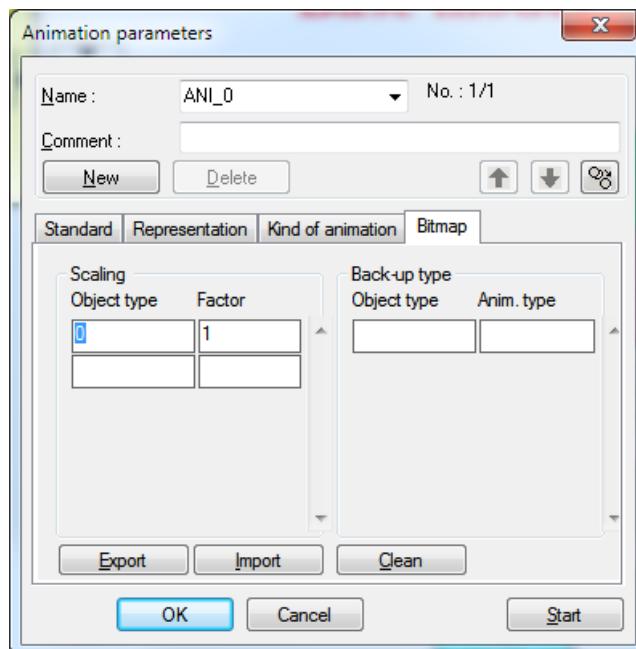


Figure 12.15: Parameters of bitmap animation 2

Fundamentally object types are represented through their corresponding graphics if this is available. If the scaling does not correspond with the size of the elements, scaling the bitmaps by a factor is possible. This happens in the left table. It is to be noted that the corresponding bitmap - file of the type of object must exist.

Since in simulation projects different object types are represented in the animation frequently with the same symbol, in the right part of the dialog a animation type can be assigned to each type of object. For object types only such are allowed, which do not possess a corresponding BMP file. As well as types may be defined as animation type, which posses a BMP file.

NOTE: Entries, which became invalid (e.g. missing bitmap files after copying a model into another folder), lead to a consistency violation. Bitmap animation cannot be started.

There is the possibility of storing a configuration. This happens with the switch **Export**. Then a folder is to be selected, where the information is stored in the file *bmpani.dat*. Likewise the information can be also taken over with the help of the switch **Import**. A file *bmpani.dat* is to be selected, in which the information was stored.

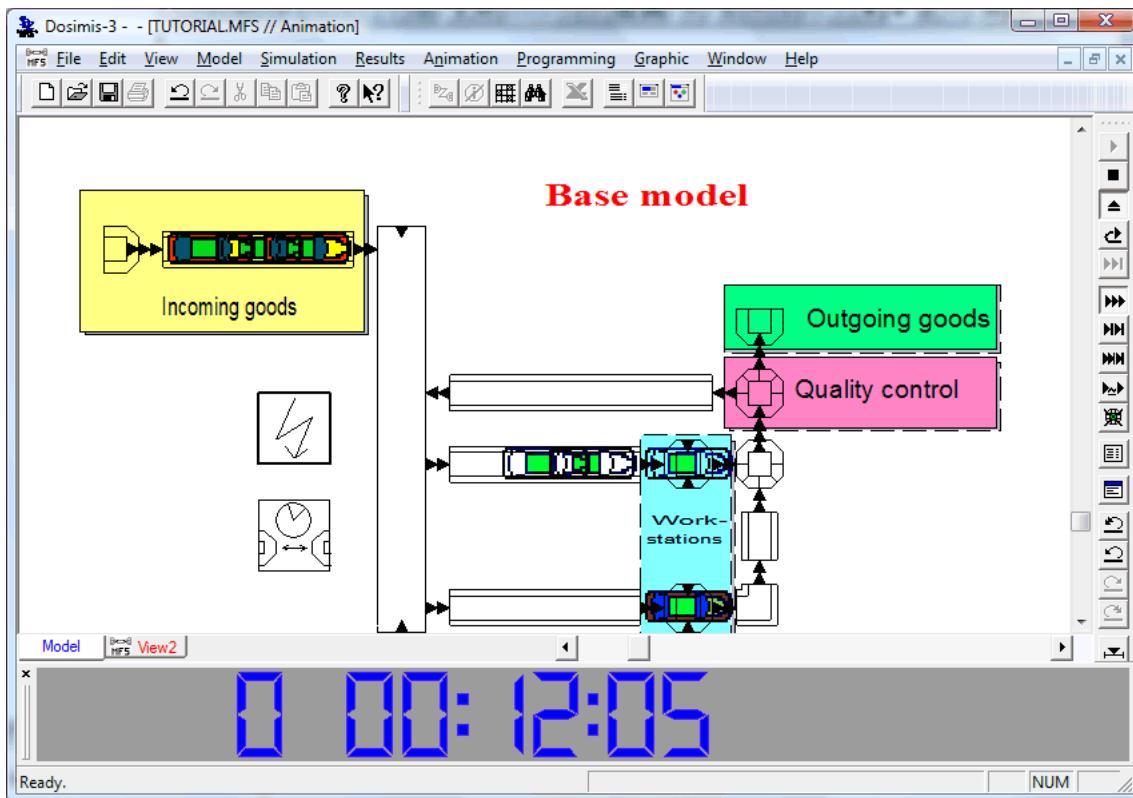


Figure 12.16: In this diagram one recognizes different object types. The vehicles with the colored car body were stored as 2.bmp, that with the skeleton-like one as 0.bmp

12.7.5 Animation Run

After successful simulation, bitmap animation can be started. This happens through activation in the menu **Animation/BMP Animation** or by activating the **bitmap animation** in the sub dialog *Kind of Animation* of the *animation parameters*.

This is only possible if:

- animation is not running,
- the file *0.bmp* is in the project directory,
- the bitmap parameters are consistent
(see messages in *Simulation/Check output*).

The operation of this animation occurs analogous to standard animation. All shortcuts, the animation menus as well as the function toolbar are available.

In this window, navigation can be performed as usual (zoom, scroll, etc.). Since however, with every navigation, the layout must be scaled again, these processes last a little bit longer.

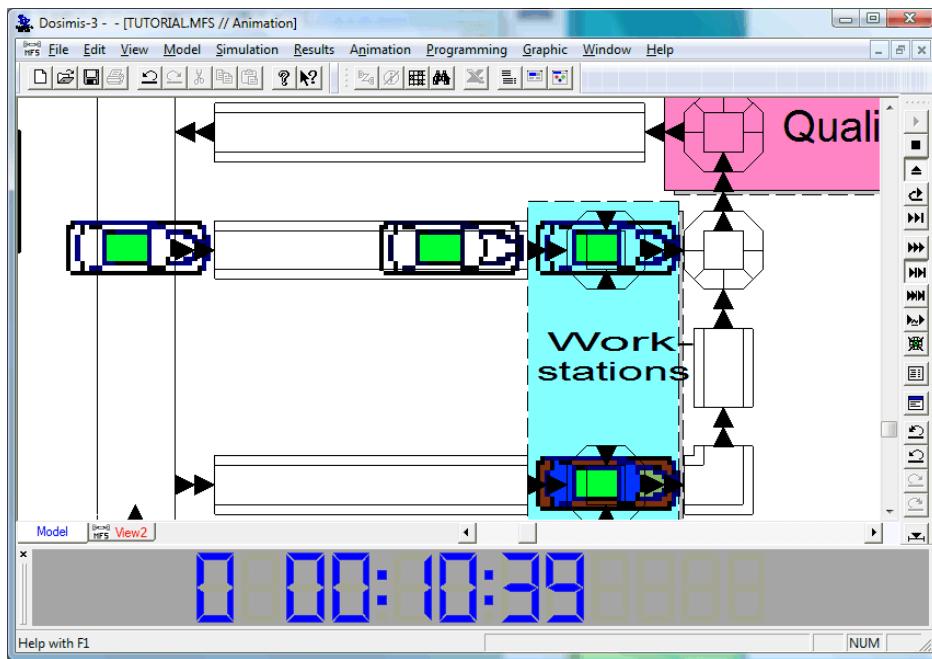


Figure 12.17: The bitmap-animation

12.7.6 Animation Figures Export

Figures of the bitmap animation can be copied into the clipboard with the aid of the **File/Export/BMP Export** function. After that they are available in other applications. A direct printing of the animation figures is not possible.

12.8 Continuous Animation

12.8.1 Introduction

In standard animation, objects, if they occur in an element, are always drawn on the first free position. This has the consequence that the objects jump. To suppress this, the user can use continuous animation. This calculates the position of the objects in the elements of the type accumulation conveyor, conveying section, bulk section and track during the passage. These are updated constantly, so that the impression of flowing develops.

Since the animation of such elements means a substantially higher cost of computation than in the standard case, continuous animation should be used only for presentation purposes.

Continuous animation also works with bitmap animation, so that the animations can appear more real.

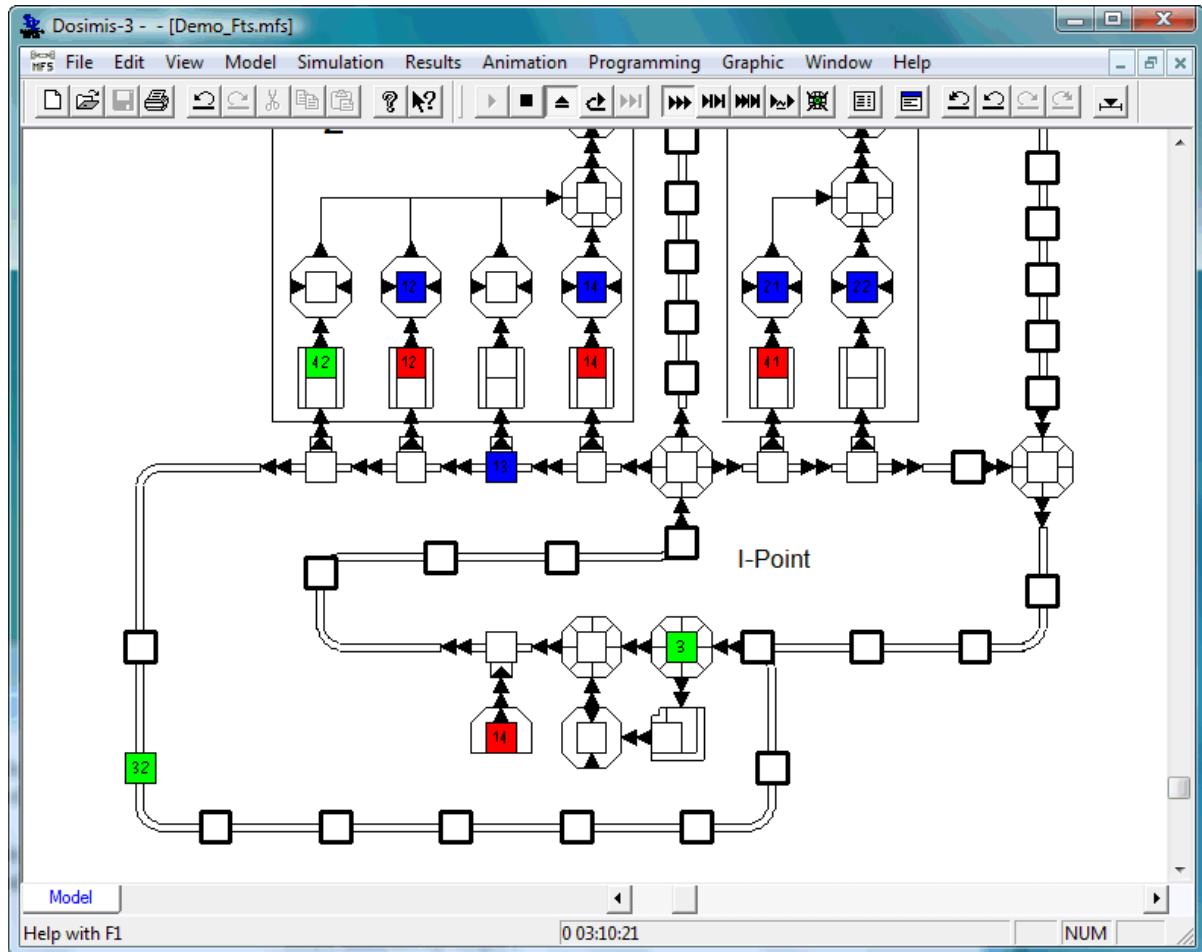


Figure 12.18: Example of a situation during continuous animation

The objects are animated exactly to their position. The numbering and placing of points of animation can be configured with the help of parameters.

12.8.2 Parameters

Continuous animation can be configured with a few parameters. These are part of the animation parameters. This is to be described here by the example of an accumulation conveyor. The said applies similarly to the other types of modules.

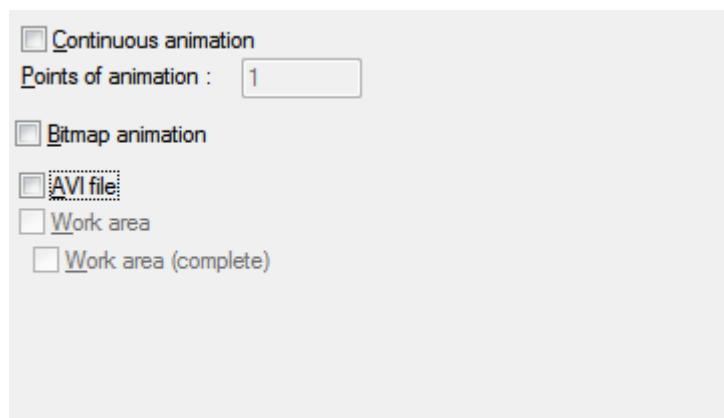


Figure 12.19: Parameters of continuous animation



In the parameter **Points of Animation**, additional points of animation can be inserted. The object is animated additionally between the segments. In the case of reverse jams, the objects are represented however at the place, which corresponds with their segments. The intermediate places remain free.

By the switch **Continuous animation** this will be activated.

Since the events in the trace file are logged with an accuracy of 0.1 seconds, it leads to problems, when an increment smaller than 0.01 results when dividing speed and length of the element. Then a passage through the element takes place in practically no time, which appears simultaneously with the accuracy of the trace file. The intermediate steps of the continuous animation are then wrongly arranged. Therefore a message appears in error file of the consistency check, if this situation is detected. The information in brackets refers to the number and the name of the module, where this occurs. At least this module should be excluded from tracing.



13 Files

13.1 File Types

During the processing of an MFS, different files are created. For every material flow system during a project several, files will be created. These differ in their extension (3 letters). **%TEMP%** means the directory, where the system variable *TEMP* points to. **!INSTALL!** is the directory, where the executable files of DOSIMIS-3 are placed during set-up. They are:

Extension	Meaning
<i>name.mfs</i>	In the .mfs file the data relating to the modules existing in the MFS concerned are managed. It contains all information on the positioning of the system elements and their parameterization.
<i>name.dar</i>	In the .dar file the information relating to the graphical representation of the MFS is stored.
<i>name.dxg</i>	In the .dxg file the information relating to the graphic comments (additional graphics) is stored.
<i>name.chk</i>	The .chk file contains all errors that were detected during the consistency check.
<i>name.pty</i>	The .pty file contains the output of a pretty print. This extension is optional and can be changed by the user.
<i>name.apl</i>	The .apl file is for the definition of work plans. If this file exists, it would be read at the start of the simulation and used.
<i>name.slg</i>	If a simulation has been executed, the resulting data (statistics, etc.) are stored in the .slg file . This file is the basic of all discrete statistic results. In order to reduce the size, elements, which are not to be logged, could be selectively taken out (see Simulation run).
<i>name.tra</i>	During a simulation, the .tra file is created. It contains all information necessary for executing an animation, i.e. all events happening during a simulation are prepared for the animation. Further on from this file, all results of continuous statistics are calculated. In order to reduce size, elements, which are not to be logged, could be selectively taken out (see Simulation run).
<i>name.err</i>	In the .err file all errors coming up during the simulation are logged.
<i>name.log</i>	In the .log file all actions of the guided vehicles are logged. This must be explicitly activated.
<i>name.aws</i>	In the .aws file the order waiting queue will be logged.
<i>name mtx bin</i>	Binary version of course matrix.
<i>name mtx</i>	In the .mtx file the course matrix of a transport system is saved.
<i>name.aps</i>	If a work plan exists, during the simulation the statistic for the work plans are written here.
<i>name.pic</i>	Assignment – file from symbols to elements.
<i>name.vbk</i>	Special statistics for random processes of failure and breaks.
<i>name.vbs</i>	
<i>name.vbt</i>	
<i>bmpani.dat</i>	Scaling and substitute types in the bitmap animation.
<i>Ds3Edit.dxg</i>	File with symbols for elements
<i>!INSTALL!/Ds3Library.ini</i>	File with the announced interface and the configuration.
<i>%TEMP%/DS3*.tmp</i>	Temporary DOSIMIS-3 files. These are created in the directory that the system variable TEMP points to. If this is



`%TEMP%/~CopyDs3*.tmp`

undefined, the files are created in the project directory. Temporary DOSIMIS-3 files during usage of clipboard. These are created in the directory that the system variable TEMP points to. If this is undefined, the files are created in the project directory.

For the backup of a DOSIMIS-3 of model the files * mfs and * dar are very necessary. Important is also the file * dxg, in which the graphic elements are stored. To be examined are the files with the extension * apl, if work plans were defined, * pic and *Ds3edit.dxg*, if module symbols were used and possibly defined initialization files of sources. An automatic compilation of these files is made with the help of the menu *File/Export/Zip archive*. On the other hand temporary files are removed with the help of the menu *File/Clean/Model* from the current folder.

13.2 Task Protocol

While statistics data of a simulation are recorded in the .slg file, the user can duplicate each object task and system behavior at fault, break and work area level in the .tra file. Here it is necessary to know the file format of this trace file, especially the abbreviations used therein.

In the first two lines of a .tra file are always the beginning and the end of a simulation, followed by the task data protocol line for line.

The continuation lines possess a uniform structure in the first 4 fields. In the first field the simulation time is located. This is followed by a letter, from which the subgroup for this animation entry is selected.

B	Module
K	Junction
C	Control (capacity monitoring, signal, ...)
A	Work area
S	Failure
E	Decision table

In the third column the number of the element concerned is located. The abbreviation for the event follows. The different events are to be taken from the following table. Some events possess further parameters. These are more exactly described in the table in the column Additional entries.

Abbr.	Sub-group	Meaning	Additional entries
A-	B	AS/RS: number of external storage orders minus 1	
A+	B	AS/RS: number of external storage orders plus 1	
AB	B	Work begins	
AE	B	Work ends	
AT	E	Decision tables, animation text output	
AW	B	Task, waiting for worker	
B-	B	Object delivered	Object number
B+	B	Object accepted	Object number
BB	B	Start movement	



BE	B	End movement	
BI	B	Object enters	
BK	B	Actual occupancy of the capacity monitoring	occupancy
BO	B	Object leaves	
BW	B	Module waiting for worker	
DZ	C	Throughput-time measurement	
ED	E	Decision tables, debug output	
EE	B	AS/RS: empty run from storage to entrance	
EI	B	AS/RS: empty run to storage	
EL	B	AS/RS: empty run from exit to storage	
ET	E	Decision tables, text issue	
FT	B	Cycle of conveying circuit	
FP	B	Object enters the conveying circuit	Type of object, number of position at entrance
FM	B	Object leaves the conveying circuit	Type of object, number of position at entrance
FW	B	Object waits at exit of conveying circuit	Type of object, number of position at entrance
G-	B	AS/RS: object delivery out of alley	
G+	B	AS/RS: object pick-up in alley	
GI	B	S/RS: alley initialization	
KA	B	AS/RS: no task, AS/RS at exit	
KB	B	Capital commitment	
KE	B	AS/RS: no task, AS/RS at entrance	
KF	K	Release link	
KL	B	AS/RS: no task, AS/RS in storage	
KS	K	Block link	
KT	K	Animation of the states occupied, waiting and announced	(±1): occupied, (±2): waiting, (±3): announced
LA	B	AS/RS: loaded run from storage to exit	
LL	B	AS/RS: loaded run from entrance to storage	
ME	C	Measuring module	
OW	B	Object standing by	
PA	B	Break begins	
PE	B	Break ends	
RB	B	Set-up time begins	
RE	B	Set-up time ends	
RW	B	Set up, wait for worker	
SA	S	Failure begins	Duration of the failure
SE	S	Failure ends	
SM	B, S	Failure, manual repair	
SW	B, S	Failure, wait for worker	
UA	B	Start failure (by decision table)	
UE	B	End failure (by decision table)	
VS	B	Shuttle position	Number of position, the shuttle is driving to.



WA	A	Worker, start work	Number of worker
WB	A	Worker end break	Number of module
WE	A	Worker, end work	Number of worker
WN	A	Worker wait in place, strategically time	Number of module
WP	A	Worker break	Number of worker
WR	A	Worker reserved, strategically timed	Number of module
WW	A	Worker in waiting status	Number of worker
WZ	A	Worker assigned task, he is going to the station	Number of module
XA	B	Start failure (module)	Number of worker
XE	B	End failure (module)	Number of module
YA	B	Start maintenance (module)	Number of worker
YE	B	End maintenance (module)	Number of module
Z-	B	S/RS: number of non-allocable external orders minus 1	Number of worker
Z+	B	AS/RS: number of non-allocable external storage orders plus 1	Number of module

For example, the following lines could occur in the trace file:

- 1) 61.0 B 34 B+ 2
- 2) 10.0 K 39 KS
- 3) 71.0 S 14 SA 600.0

When decoded, this information means:

Module 43 has at time point 61.0 seconds picked up an object type 2
Link 39 was blocked after 10 seconds
A disruption begins after 71 seconds in module 14. The duration is 600 seconds.

13.3 The MFS file

When the same two abbreviations are used for different information the relevant information is given from the situation in which the abbreviation is used. The abbreviations for the *.mfs file are given in alphabetical order as follows with no consideration for any particular situation.

Modifying this file with an editor is strongly discouraged. All short cuts are derived from their German meaning.

ANZ	Number of objects to be assembled or disassembled
ARB	Area of definition for work areas
ART	Type of fault or break
ASG	Discharger speed



ASL	Module of discharger type
AST	Module of work station type
ANH	Stop object yes or no (CRS)
AUS	Count of exits
BAU	Area in which a module is defined
BED	Area of definition for GSS
BEL	Module of loading station type
BEZ	Name of module, material flow system
BLK	Module of type block section
BLG	Loading speed
BNR	GSS number
CHK	J or N shows whether a consistency check has been carried out
DEF	Determination of standard values (SLA = object length, GSW= conveying speed, FRP= forward control yes or no)
DEF	In definition area of module states whether the relevant module is defined or not (white or green screen display)
DEM	Module of disassembly type
DFG	Conveying speed
DRT	Module of type turntable
DVW	Module of type double shuttle
EIN	Number of entrances, assembly entrance
ELG	Unloading speed
ENT	Module of type unloading station
ERW	Expectation value
ESG	Infeed module speed
ESL	Module of type infeed station
ETAK	Action within a decision table
ETBE	Condition in a decision table
ETGS	Decision table in a GSS
ETIA	Initial action within a decision table
FIN	End of a definition area. There are the following areas: - standard values , modules, work areas, faults, breaks, modules in the statistics issue, modules in the trace issue
FKR	Module of type conveying circuit
FRP	Forward control
FST	Module type conveying section
GES	Conveying speed
GST	Basic position
GSW	Conveying speed
GWL	Conveying speed - slow
GWS	Conveying speed - fast
HIS	Histogram distributed work mode
INT	Interval statistics yes or no
KA1	Exit link number 1 etc.
KE1	Entrance link number 1 etc
KNA	Link module exit
KNE	Link module entrance
KPA	F-parameter for Erlang distribution
KKN	Module type crossing
KWA	Stop conveying circuit on object delivery, yes or no



LAE	Belt length
LAG	Module type storage
LAN	Conveying route
LAW	Discharger route
LBW	Loading route
LEW	Slip-in route, unloading route
LFP	Module type LIFO-buffer
LFW	Main conveying section (dependent on module type), slow route
MAN	Manual task yes or no
MFS	Material flow system (IND. gives name of material flow system)
MOL	Shows via module number with which module an additional statistic is to be drawn up
MON	Module type assembly station
MTW	Average value of Erlang distribution
NUM	Number of module
OBL	Length of object
OFZ	Opening times
OTA	Task list for an object type in a work station
OTL	Frequency of object type change in a work station
PAL	Module type pallet changer
PNZ	Module type Petri_net_state
PNE	Module type Petri_net_event
PRI	Assembly priority
PST	Module type bulk section
QUE	Module type source
RST	Set-up matrix
RSZ	Set-up time
RWK	Worker required for set-up yes or no
SEG	Segment length
SEN	Module type sink
SIZ	Simulation time
SLA	Standard object length
SLZ	Closing time
SST	Module type buffer section
STL	Shows via module number which module is taken into account in the statistics issue
STO	Definition area for faults or breaks
SWB	Module type slewing belt
TYP	Shows module type
TZS	Coincidental start value for object-dependent task
VAR	Variance, standard deviation
VEG	Distribution of assembly time
VEL	Module type distributor
VES	Distribution strategy
VLZ	Preparatory run time
VST	Stop slewing belt yes or no, preparatory run statistics yes or no
VTR	Preparatory trace yes or no
VTW	Module type shuttle
WAR	Waiting for assembly yes or no
WEG	Mutual conveying route (CXN)
YGL	Conveying speed slow in y direction



YGS	Conveying speed fast in y direction
YLW	Slow route in y direction
ZEL	Module type co-coordinating merging station
ZST	Start value of a coincidental number generator allocated to a module
ZSW	Start value of a coincidental number generator, unchanged basic value: 69100



14 Introduction to the Decision Tables

If the control options that are offered by the standard, should not be sufficient, they may be extended by the concept of decision tables. With the help of decision tables (abbreviated DT), the user of DOSIMIS-3 can implement exactly of the needed control strategies.

Decision tables are tools for:

- blocking and de-blocking of junctions
- changing object-types
- generating new object-types
- choosing one exit among others for an object to depart a module
(user-defined output strategy)
- choosing one entrance among others for an object to enter a module
(user-defined input strategy)
- elementary math-operations
- keeping track of global user-defined variables
- and many more.

The decision tables make an efficient instrument available to the DOSIMIS-3 user for the complex and flexible material flow control of DOSIMIS-3 models. Both while processing global system statuses and in the case of definition of own strategies, programming specific procedures is no more necessary for a longer time in the source code of the simulator, but only processing of decision tables. The big advantage of using the decision tables is that the user does not need to master sophisticated programming languages or to know the data structure of DOSIMIS-3. He just has to memorize the programming parameters that are listed in the appendix.

The user of DOSIMIS-3 has several ways to extend the standard control options. A basic distinction is made between superordinated decision tables (GCT-DT) and local DTs. A GCT-DT has its own icon in the model layout and can be flexibly connected to the controlled modules. Local decision tables are created by pressing a button within each element using the appropriate button in the parameter screen.





15 Connection of the Decision Tables

Before setting up a decision table the user has to choose an element which supports decision tables. Some modules allow defining of strategies like right of way strategy or distribution strategy by decision table. A special element created for working with decision tables is the global control (GCT).

With the GCT the user can define lists of the modules and junctions, which are relevant for the respective problem definition. It proceeds similarly to failures and work areas. In the mode *linking active* will actively first the GCT symbol is to be clicked and then the modules and junctions are to be selected, which are to be taken up to the list.

Following modules can contain a decision table:

- A **right of way strategy** can be defined for turntable, Crossing, Slip in module, shuttle, paired shuttle and infeed element by a DT (right of way - DT)
- A **distribution strategy** can be defined fur turntable, crossing, discharger, shuttle, paired shuttle and distributor by a DT (distribution - DT)
- Workstation, assembly and disassembly can possess a **working time -DT**.
- By the global control (GCT) you can define a decision table, which reacts up on arbitrary events in the model (GCT-DT)
- In the dialog from *Model/Transport-Strategies* order-DT, vehicle-DT and restriction-DT can be created (user defined strategies for transport systems (TS)))

Further DT types are the global decision tables and the sub decision table. With these special forms will be discussed later.

All decision tables co-operate with reference modules and junctions. These are model items, which are influenced by the decision table or to decision making to contribute.

One GCT-DT contains additionally activating modules. Events in one of these modules lead to the processing of the decision table.

First is to be clarified, with which events a decision table is executed. The simulation program DOSIMIS-3 treats any change in state as an event. Among others an **event** is happening in the cases of an object passing through a module, an object being processed, the object type changing.

For decision tables which are not defined by GCT, the impulse always occurs when the corresponding action (e.g. determination of next output at a distribution DT or a calculation of the operating time with a working time DT) of the module/Transport control is evaluated.

The GCT-DT is analyzed, when an event in one of the activating modules takes place. Here the question arises, which modules concerning a problem are reference modules and which not. This can be clarified by the term of an event. The list of the activating modules of a global control should contain modules, where an event involves the activating of the decision table directly. The remaining modules are reference modules (indirectly concerned) and



should be placed into the appropriate reference list. This problem arises, as mentioned above, only with GCT decision tables.

The distinction between modules which call up the table and modules of reference makes sense because a module can contribute towards the solution of the problem without needing to call up the decision table. That happens in the case of just recalling data of this module, i.e. the module only provides information required for processing a decision table but does not call it up. A major advantage that comes with this distinction is the reduction of computing time. This is obvious, because reference modules do not activate the decision table. Because of performance consideration it is advised, not to define every module as activating module.

An event in a junction does never result in calling up the decision table.

15.1 User-defined Strategies

The definition of own strategies does work without the distinction between references and non-references. In this case the only module with an event causing the table to be called up is the one featuring several inputs/outputs and having a strategy defined for it. To be able to set up the table at all, the parameterization of the module must be preceded by the choice of **user-defined strategy**. Otherwise the module would utilize a strategy given by the system and ignore the decision table.

A **restriction decision table** is used to prevent a combination of two elements. If the restriction-DT returns 0 via the return statement, the combination is allowed (the restriction does not apply). If the return value is not 0, there is no disposition.

A **priority decision table** is used to evaluate a combination of two elements. The return value of the priorities DT (return statement) assigns a value to each combination. This is then used to choose the best among all combinations.

This type of decision tables is used for example in the transport strategies, where can help the combination of a vehicle and a transport order are tested or rated.

15.1.1 User-defined Right-of-way / Distribution Strategy

Right of way DTs determine the entrance, where an object enters a module (Right of way Strategy). In the same way a distribution DTs determine the exit, where an object leaves a module (Distribution strategy). The object of the decision table is to determine and return the number of the entrance or exit. To return the number of the entrance / exits the return instruction is available (e.g., in a distribution decision table return (3) means: select exit no. 3).

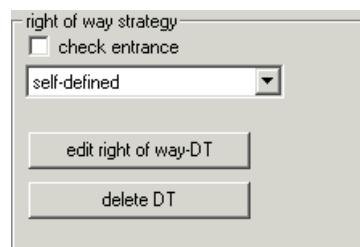


Figure 15.1: Connection of right of way-DT



When user-defined distribution strategy the following predefined values are available. **Act_object** identifies the object which is to be taken over. **Act_entrance** contains the input number where the object to be acquired. If stopping is prohibited in the module, the object has not yet been adopted. The analysis of the distribution strategy in this case will be held in combination with the analysis of the right of way strategy, because it must be guaranteed, that the object can leave the module.

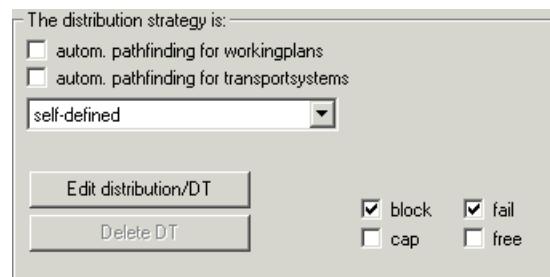


Figure 15.2: Connection of distribution-DT



15.1.2 Self defined Working Time

The definition of a working time DT is connected in the parameter masks of the workstation, assembly and disassembly. If a decision table is defined, this is evaluated with the determination of the processing period. For this the objects, which are to be processed at this station, must be entered into the list like used. Now, if a decision table is defined, this is evaluated. The decision table must be defined that way, so the duration of processing is passed back by the return statement. If the result is positive (a value is genuinely larger than 0), this value is as the processing time. If calculation result is 0, the processing period is taken from the entry in the list in the dialog.

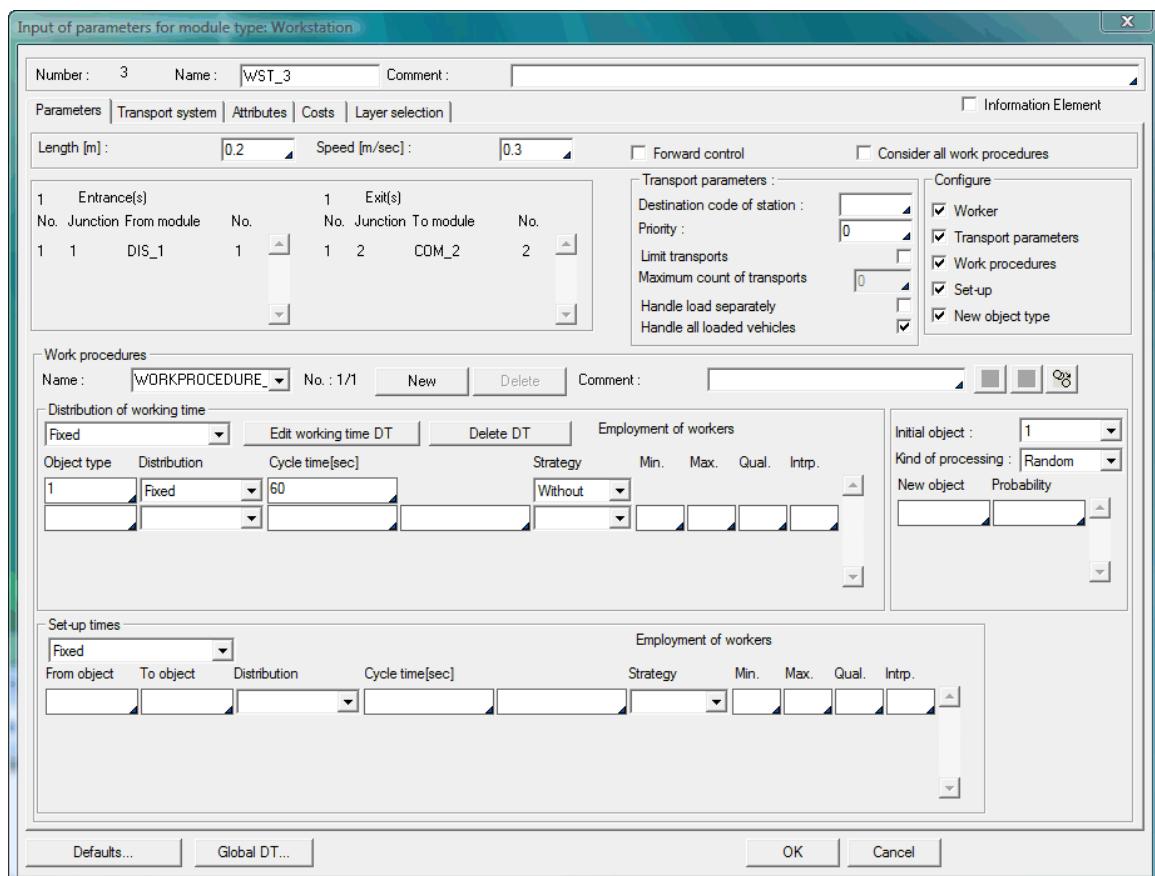


Figure 15.3: Connection of working time- decision table



15.1.3 Shuttle Decision table

The definition of a shuttle - decision table is used to let a not fully discharged shuttle drive to an input for receiving another object. This only makes sense if the shuttle has a capacity > 1 and thus has multiple collections of objects is possible. The call will take place whenever an object either has entered or left the shuttle and still capacity is available. If it returns a value greater than 0, this entrance is approached. It is not checked whether there really an object waits.

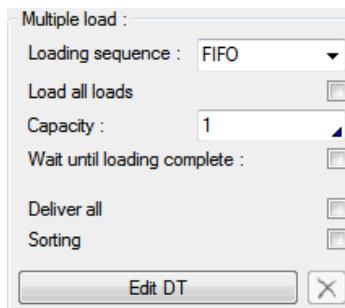


Figure 15.4: Connection of a shuttle decision table

The following predefined values are available. **Act_entrance** contains the input number by which an object was taken. **Act_exit** contains the number of the exit over which a object has left the module. Since either a takeover or a levy has taken place, only one of the two values is positive. **Act_object** contains a reference to the object, which was entering / exiting the shuttle.



15.1.4 Conveying Circuit Decision Table

The conveying circuit - decision table is a restriction decision table. This is called when an object should be delivered to an output. If the restriction is valid, the object is not given but moves on.

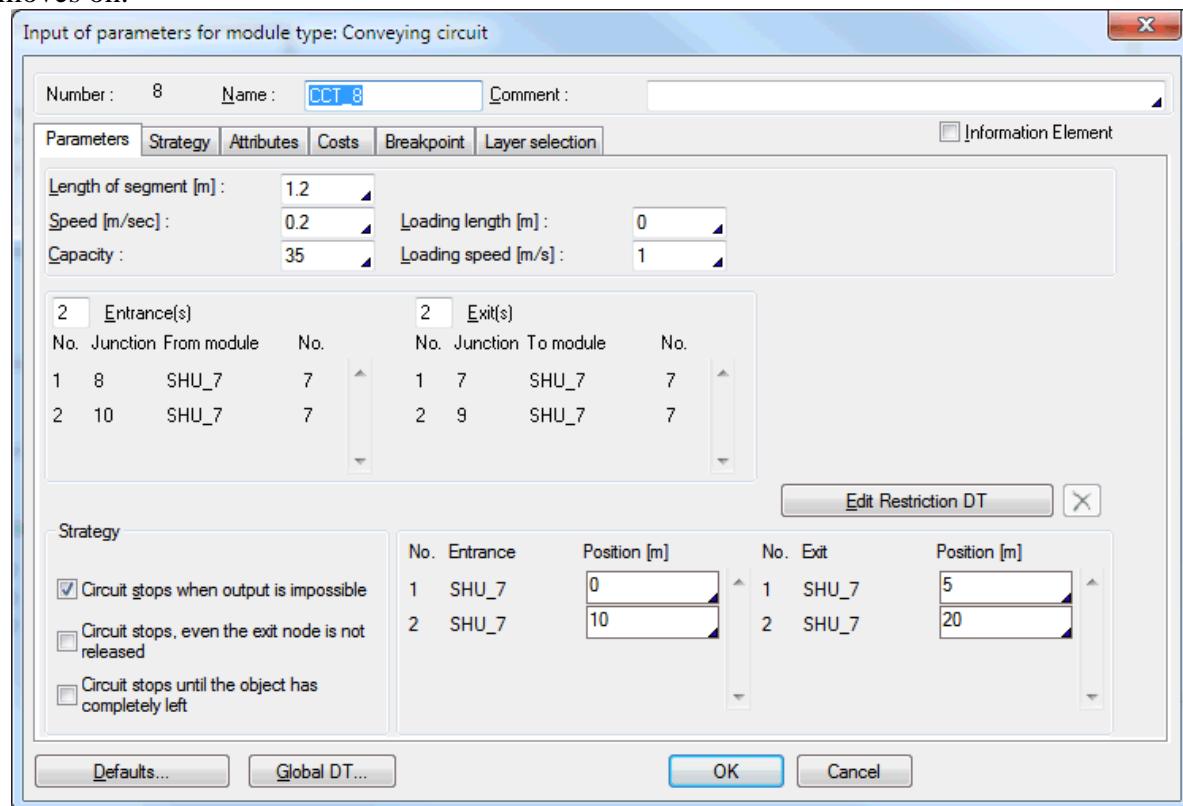


Figure 15.5: Connecting decision table of conveying circuit

The following predefined values are available. **Act_object** contains the object that is to be delivered. **Act_entrance** contains the number of the exit on which the object will be placed.

15.1.5 Transport Strategy

The menu **Model/Transport Strategy** leads the user to the menu, which is already described in the first part of the manual in chapter *Transport-Strategy*. By using decision tables it is possible to create self-defined strategies. There are three decision tables available:

Restriction-DT: A restriction-DT is created when clicking on the button **Edit Restriction-DT**. Such a DT will be called for every order to be assigned. Actual data will be found in **act_agv** und **act_tsorder**. If the restriction DT returns 0 (by the return statement), **act_agv** and **act_tsorder** can be combined. If the value supplied is not 0, there is no disposition

Vehicle DT: **Self-defined strategies** are created by selecting the item **self-defined**. For every released vehicle this DT will be called - multiple for each free order. Order and vehicle will be found in **act_agv** und **act_tsorder**. This DT should leave a priority with the return statement. DOSIMIS-3 will combine the vehicle/order pair with the lowest priority value (1 is better than 3).

Order -DT: Analogy to vehicle-DT **Self-defined strategies** are created by selecting the item **self-defined**. For every released order this DT will be called -



multiple for each free vehicle. The order-DT works in the same manner as a vehicle DT.

In context with these transport strategies, special options in own menus are available to the user.

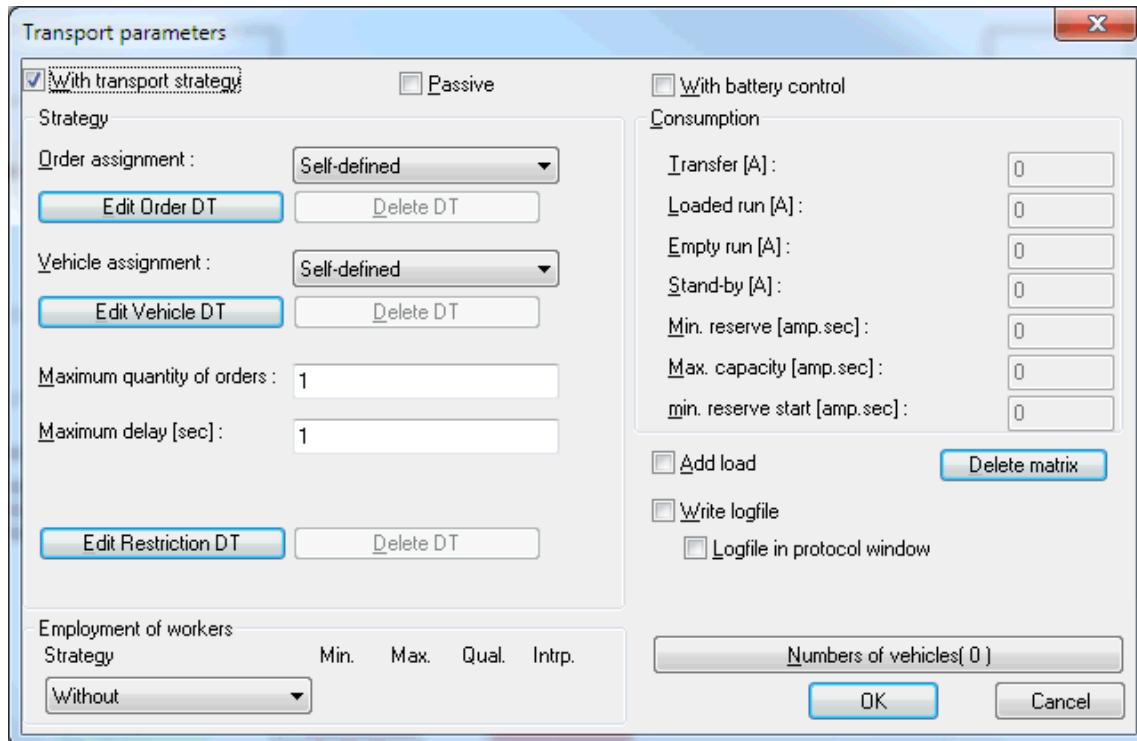


Figure 15.6: Mask Transport-Strategy

The restriction decision table is always available, the other two only if the appropriate strategy is *self-defined*.

15.1.6 Work Area Strategies

In the parameters of the work area a restriction decision table can be created.

Restriction-DT: Creating a restriction DT can be done by clicking **Edit Restriction DT**. Such DT is called for each work to be scheduled. Current data are in **act_worker** and **act_work**. If the restriction-DT returns a value of 0, **act_worker** and **act_work** can be scheduled. If a value supplied is not 0, there is no disposition.



15.2 Definition of a Decision Table

To define a decision table the user has to press the button **edit GC-DT** (shown on the example of a global control).

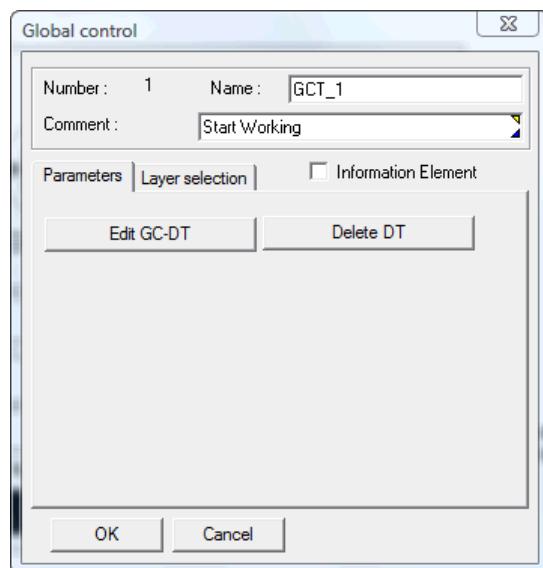


Figure 15.7: Mask „System control“

This leads to the **Parameter Mask for Decision Tables** will be opened.





16 The Parameter Mask of Decision Tables

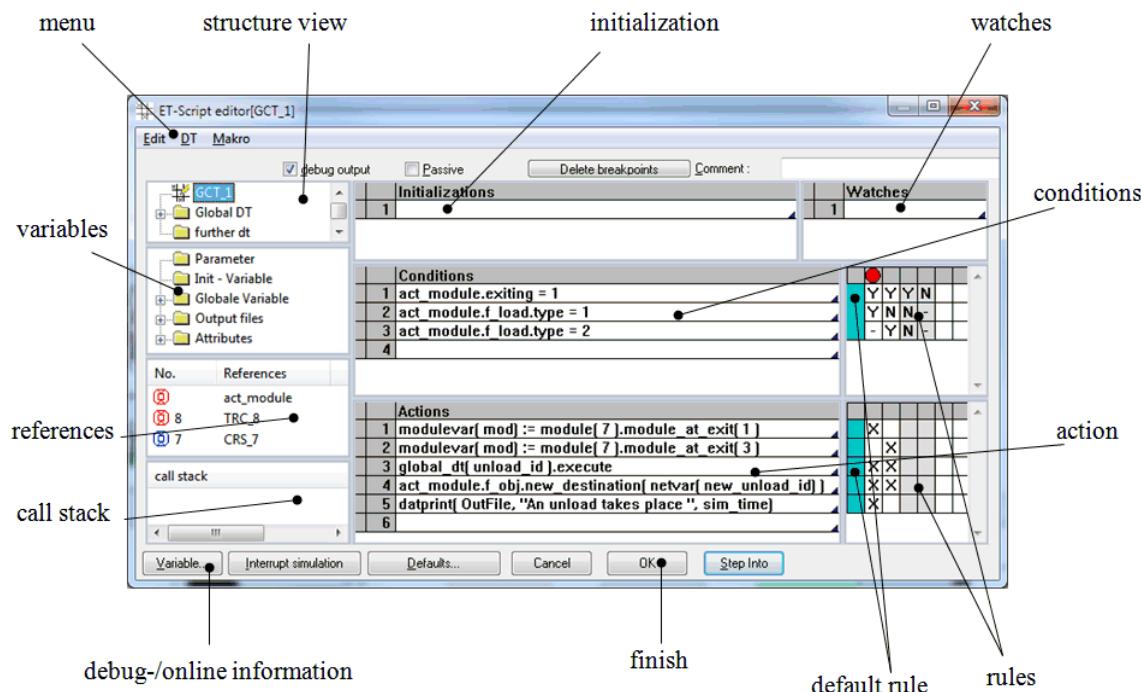


Figure 16.1: Editor

The decision table parameter mask is the user interface to the decision tables and enables the direct manipulation of the DT. Furthermore the administration of the global decision tables and the net attributes is integrated. Additionally it enables an overview over the added modules and junctions.

In models that were created with a foreign-language version of DOSIMIS-3, the text of the decision tables in the relevant foreign language are stored. If this is to be translated into the language of the current version of the program, this can be done using the Translate button. In the menu *Model* first click on the *Configure* item. In the register *Decision tables* the translate button can be found.

- **Switch Debug Output**
Via the **debug output** it is steered whether debug protocol for the decision table is to do during simulation.
- **Switch Passive**
Decision tables can be **passivated**. For this the appropriate switch is to be set. Passive decision tables are marked with a green square in the left upper corner.
- **Button Defaults ...**
In each case it is possible to see or change the model constants via **Defaults**. The global decision table data can be defined in the structure view.
- **Button Delete Breakpoints**



With a click on the button **Delete Breakpoints** all breakpoints of the decision table can be deleted simultaneously. When the CTRL-Key is pressed, the debug output is deactivated too.

In the lower area there are some buttons that appear only in the animation or online simulation.

- **Button Step Into**

By clicking on the button **Step Into** the next decision table call that is triggered by an action will be pursued. The simulation is interrupted, even if no breakpoint is set in that decision table.

- **Button Interrupt Simulation (Animation)**

With a click on the button **interrupt simulation** the sequence of the simulation is interrupted after closing the dialog. The same applies to when the decision table dialog is displayed during the animation.

- **Button Variable ...**

With a click on the button **variable ...** the variable window can be activated during online simulation. There you can investigate the key parameters of the modules and decision tables.

16.1 Initializations / Conditions / Actions

In these three windows initializations, as well as conditions and actions. The methodology is identical in all three windows.

Basically three steps belong to the complete definition of a decision table.

Step 1: Definition of initializations (optional)

Step 2: Definition of conditions

Step 3: Definition of actions

Step 4: Combination of certain actions to certain conditions. This step is called also the definition of the rules. A rule corresponds to a column in the rule matrix, which is on the right from conditions and actions.

The definition of initial statements is not necessary. It depends on the need of initializations during the handling of conditions and actions of the respective table. If this is the case, then the initialization variables can be initialized here.

16.1.1 Step 1: Initialization

An **initialization** defines an action, which will be executed during the processing of a decision table.

Example: **count := 1**

Assigns the value of 1 to a variable *count*.



16.1.2 Step 2: Condition

A **condition** is a query, or a comparison. Conditions always meet a predicate, which can be answered by Dosimis-3 with "yes" or "no". Depending upon status of the system during a simulation the response can fail naturally differently.

Example: **count > 5**

Checks whether the variable *count* has a value greater than 5.

16.1.3 Step 3: Action

An **action** is executed always, when a determined record of conditions is fulfilled, or is not fulfilled. Actions can direct influence the simulation flow. So e.g. junction are blocked, modules be disturbed or data be outputted. Naturally there is a still greater spectrum of possibilities for the influencing control.

Example: **act_module.junction_at_entrace(1).block**

Blocks the junction from entrance 1 of the activating module.

16.1.4 Step 4: Rules

An action is performed whenever a fixed set of conditions fulfilled or not fulfilled. This combination of conditions is called a **rule**. A rule is a column (column in the right pane). The first column defines a specific rule, namely, the **default rule**. The actions that are in the column marked with an "X" are always executed if no other rule applies.



16.2 Input

16.2.1 Selection of a Line

A text line is to select for later operations by a simple click. The selected line is black emphasized.

16.2.2 Adding Lines

To add a line it is enough to double-click with the left mouse button on the free input line in the desired window. Alternatively one can operate with the context menu in each window through the right mouse button. In that menu one selects the option **Add** or one selects in the main menu **Edit/New initializations** (or **New condition/New action**).

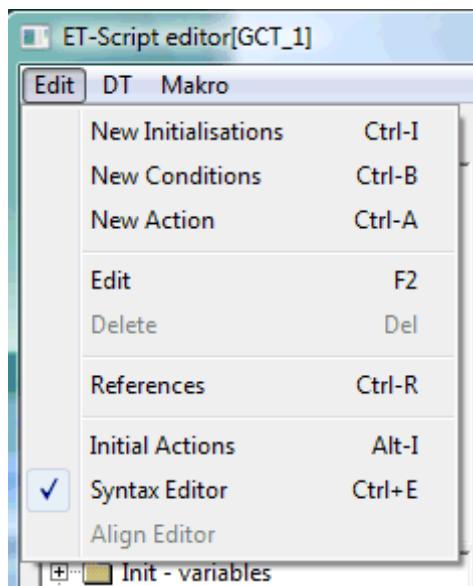


Figure 16.2: Menu Edit

16.2.3 Editing Lines

For line processing a double-click on that line is enough. Alternatively one can do it with a single click and by the context menu **Edit**. Further on the possibility exists, to start editing by the main menu **Edit/Edit selection**. Also the key *F2* or the *return key* starts the editing mode for a selected line.

In the opened line the input cursor appears. You can now enter free text and edit it with the windows standard keyboard operations. The inputs must correspond to the syntax rules of the decision tables.

If the input is erroneous, the cursor induces itself to the position of the error. One should note that the cursor branches at the place of an error recognition. The actual error can have occurred however already beforehand!

Example: Input of an action:

```
print(Count is“, act_module.actocc)
```



After input of this action line a syntax error is announced and the cursor is set behind the word *is*. There the error has been detected by the system. However a quote behind the first bracket is actually missing. The correct text would look in such a way:

```
print("Count is“, act_module.actocc)
```

The definition of a condition/an action may extend over several lines. For this a return is to be placed at the end of a line. This is reached by simultaneous pressing of ctrl and return key.

16.2.4 Finish Input of a Line

Pressing the *Return / Enter key* terminates the input. The cursor branches automatically into the next line and expects now further inputs. Pressing the *return key* in an empty line will leave the input mode.

Alternatively one can click also into the free area below the last free line.

If one uses the *Escape key*, then the input is also terminated. All input is however rejected. The line is afterwards again in the status, in which it was before the last handling.

If the input was not terminated by pressing the *Escape key*, then a syntax check takes place. This parses the input line for syntactic correctness. If input error should have occurred, the line is marked with a red background.

16.2.5 Syntax Editor

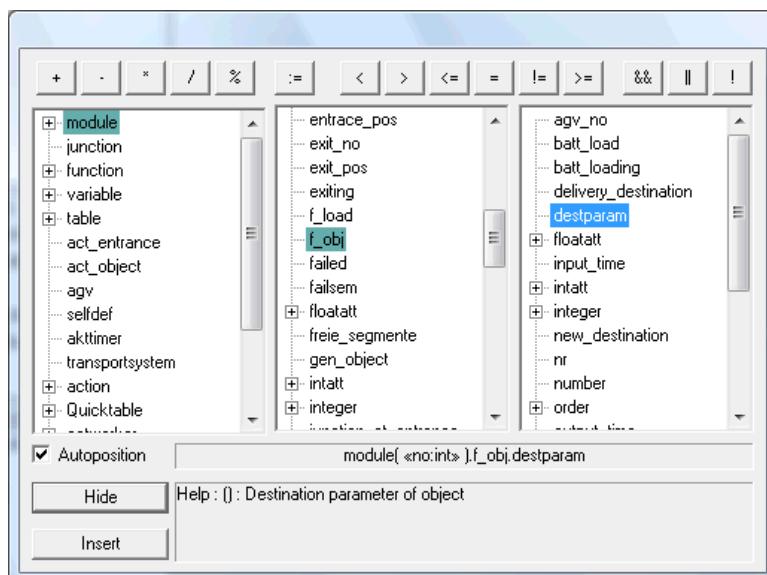


Figure 16.3: Syntax Editor

As assistance during the input the syntax editor is available. During the handling of a line it can be switch on or off by the menu item **Edit/Syntax editor**.

The syntax editor supplies a selection of keywords in three windows. The displayed items depend on the selection in the left window. In the first window base items are displayed. These include all activating and reference modules, as well as reference junctions, variables,



simulator functions, further decision tables and complementary data for certain table types (actual entrance, selfdef, etc.).

There is an extra output area below the selection windows where the created expression and a small help text is drawn. This contains a description of the current selection with a list of the needed parameters. The text will be pasted into the edited line by pressing **insert**.

At the top the entry *module* is found. If reference or activating modules for the decision table are defined, on the left of module a "+" appears. By click on this sign all modules important for this decision table are displayed. Here at least the activating module of the decision table will be shown, which is always called *act_module*. Also with decision tables without explicit activating this happens. In this situation is however the use of *act_module* may be illegal.

If one selects now *act_module* by a simple click, then all attributes and operations for this module appear in the second window. Those keywords are called module attributes, which refer directly to a module - thus values of the module, like its number. Operations are functions, which can be executed on this module, e.g. disturbing the module. If one selects for example *f_obj*, the attribute refers to the first object in the module. All attributes and operations to this object appear immediately in the most right window. Here one finds the attribute *type* among others. Double-clicking on this item now inserts the following text at the cursor position of the input line:

act_module.f_obj.type

The syntax editor thus offers the possibility defining complex queries and actions without detailed knowledge of the decision table language by operating from the left the right through the windows. As soon as an entry is selected by a double-click, the appropriate text is inserted into the input field.

When a selection should need further parameters, then these are emphasized in the input field, so that one can complete immediately the missing parameters. An example:

Double-click in the left editor window on the word *module*. The following text is inserted:

module(«no: int»)

Since no module was specified, the syntax of the decision tables expects the specification of a module number in the parentheses. The absence of this value is clarified by the text **«no: int»**. One detects that on the one hand a number (here: the number of the questionable module) is to be indicated, on the other hand an int value is expected. After a double-click the text in the parentheses is automatically selected, so that only the input of the appropriate number must be done by the keyboard.

By the buttons at the upper edge of the window the possibility exists, to insert special characters and formula symbols directly.

By the selection of **auto position** the editor is instructed to look itself for a favorable position on the display so that the input line is never covered. Naturally the editor can be shifted freely. When the editor should again be opened at a later point in time - this is done also by



editing the next line - it automatically looks itself for a good position. If this behavior is unwanted, it can be prevented by voting out *auto position*.

16.2.6 Context Sensitive Selection

During some inputs a so-called popup window is inserted. This window shows possibilities of the further input and facilitates a creating of terms.

Example: Input of a condition

Therefore the word *act_module* should be typed and afterwards the *dot key* (.).

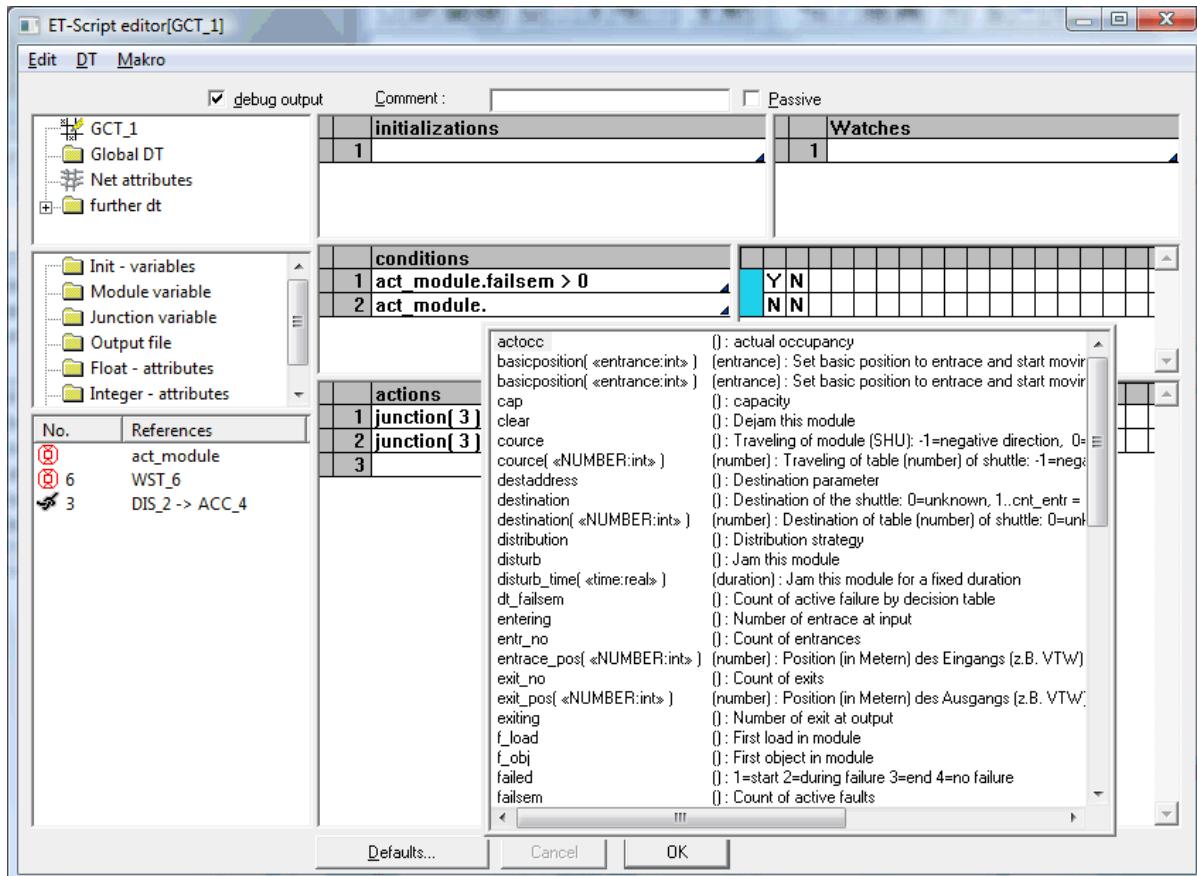


Figure 16.4: Context Sensitive Selection

A window, which displays the possibilities after the input *act_module*, is opened. With *act_module* it concerns the current activating module, to which the table refers. After pressing the dot properties (*attributes*) and actions (*operations*) follow, which are possible with the module. ***act_module.throughput*** refers to the throughput of the current module - which then could be compared in a condition with another value (***act_module.throughput = 15***).

In the left column of the window the syntactically correct text to an input is displayed. The right column serves as reference for the user. On the one hand it shows a possible parameter - e.g. it means (no) that a number must be assigned to this keyword as parameters - on the other hand it receives a short assistance text to the respective keyword.

Example: ***act_module.module_at_exit(1)*** supplies the module at the output no. 1 of the current module.



The selection of text within the popup window is done by mouse, as well as by keyboard. With the cursor keys *up* and *down* a selection beam can be moved through the popup window. With the return key the selected text can be transferred to the input line. This can be achieved also by a double-click on the keyword with the mouse.

The selection in the window is updated synchronously to the input. When the input is e.g. *act_module.e*, the selection branches in the popup window on the first entry with e (here: *entering*). After pressing area further keys xiti (*exit*), the selection branch occurs on *exiting*. By pressing the return key, the word is transferred and the input line now is **act_module.exiting**.

The popup window closes automatically, as soon as an input is finished. This happens e.g., when the space key or a special character is pressed, when the dot, which opened the popup, is deleted or when the cursor is positioned on a new line.

16.2.7 Deleting the Input Line

To delete an input line completely it is to mark with the left mouse button and from the context menu **delete** is to select (or in the main menu **Edit/Delete Selection**).

The deletion of an input line is not possible, when from that a surplus of *rules* results (see below).

16.2.8 Deactivating single action

For test purposes single actions can be deactivated in a decision table. For this the field left to the numbering of the action is to be turned on. The action is grayed and not considered during simulation.

16.3 Rule Window

The rule windows visualize the connection between conditions and actions. For this beside the condition window and the action window a matrix is displayed in each case.

The common columns of both matrixes are called rules. If the condition section of a rule is fulfilled during the simulation, then those actions are executed, which are marked in the appropriate column of the action matrix.

Entries in the rule matrix are *Y* for yes (the condition is fulfilled), *N* for no (... does not fulfill). “-“ means indifferently - the result of the condition is not analyzed.

Relations represent a special case of condition. Relations are numeric comparisons, for which the comparison operator was not indicated yet. Instead of it in the condition the substitute symbol ?? is used. With a relation the following characters can occur in the matrix:



- indifferent
- << less or equal
- >> greater or equal
- < less
- = equal
- > greater
- U not equal

It is advised not to use relations. The legibility of the rule matrix is often more difficult.

16.3.1 Creating Rules

To create a new rule column one clicks into the first free column of a rule matrix. The column is filled now with values: All lines above the cursor are filled *indifferently*, the array module with *Y* and all items below with *N*.

There is a limit, how many rules can be created. When no more rules can be created, then this is not a limitation of Dosimis-3, but a mathematical limitation. A boolean (Y/N) - condition can have only two statuses - *true* (*Y*) and *false* (*N*). Therefore one condition can only have maximal two rules. Two conditions permit up to four rules,

Rules without actions or only deactivated actions are marked in light grey.

16.3.2 Change Rules

By clicking matrix entries with the mouse one modifies the entry. With conditions the entry changes from *indifferently* to *yes*, then from *yes* to *no* and at least from *no* back to *indifferent*. With actions it can be changed between *active* (with a cross) or *passive* (empty field).

16.3.3 Delete Rules

In order to delete a rule (that means here: a complete column of both matrixes!) one clicks into the gray area at the top of the matrix. Thus the rule is selected and appears marked black. Through right-click into one of the two matrix windows a context menu is opened, in which *delete column* is to select.

16.3.4 Copy Rules

Rules can be duplicated by the context menu. After selection of a rule the context menu can be opened by a click with the right mouse button. By the selection of *Duplicate Column* a copy of the rule will be inserted right of the original.



16.3.5 Set Breakpoint

By a double-click on the gray area at the top of a matrix a breakpoint can be set. Rules with a breakpoint let the simulation interrupt during the on-line simulation (if the rule reaches, and the appropriate actions are to execute), in order to analyze the flow in a decision table.

Example of rules:

	conditions		
1	act_module.exiting = 1	Y	Y
2	act_module.f_load.type = 1	Y	N
3	act_module.f_load.type = 2	N	Ø
4		Ø	Ø

	actions		
1	modulevar(mod) := module[7].module_at_exit[1]	X	
2	modulevar(mod) := module[7].module_at_exit[3]	X	
3	global_dt(unload_id).execute	XX	
4	act_module.f_obj.new_destination(netvar[new_unload_id])	XX	
5	datprint(OutFile, "An unload takes place ", sim_time)	X	
6			

Figure 16.5: Rules

Explanation (look at the first rule):

- If condition 1 AND 2 are valid (an object has left the actual module by exit number 1 and the load of this object is of type 1 (the third condition is irrelevant))

THEN

- The three actions 1, 3 and 4 are executed.

The second rule determines:

- If condition 1 and 3 are valid (type of load of the first object is equal to 2 during exiting through exit 1)

THEN

- Only action 2, 3 and 4 are executed.

Here additionally a breakpoint is set. If this case should arise during the animation, or the online simulation, the simulation is interrupted temporarily and the decision table is displayed (the releasing rule is emphasized). It is to be noted that for the decision table the *debug outputs* must be activated (bottom of the decision table dialog) and for DT Debug at least the decision table animation (global DT - dialog) is to be selected.

If neither condition 2 nor 3 are valid (the load is neither of the type 1 nor of the type 2) or condition 1 invalidly (every other event except entering into the module by entrance 1), then no action is executed.



16.4 Highlighting by Colors

The following figure shows the highlighting by colors of rules and conditions.

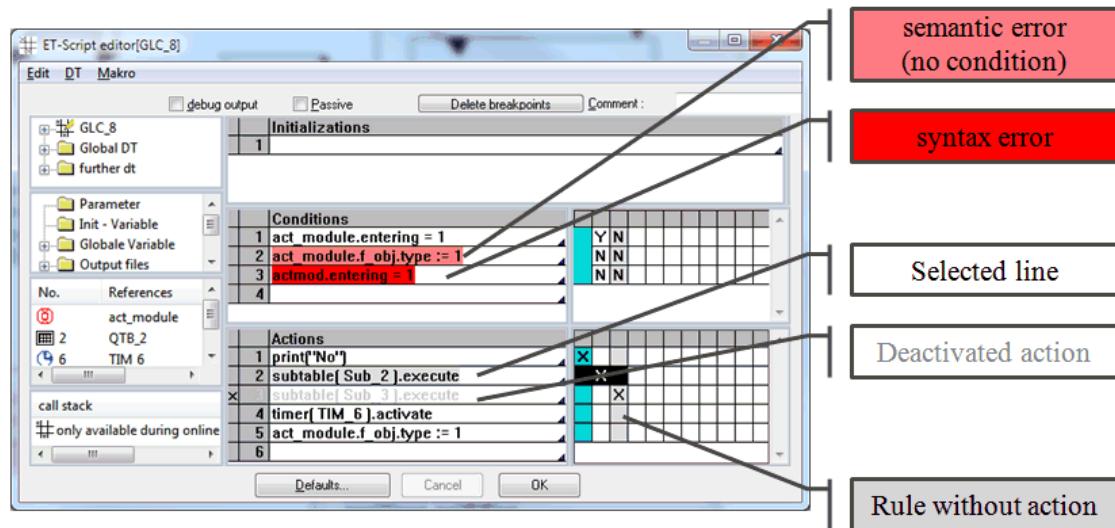


Figure 16.6: Highlighting in editor

16.5 The Watch Window

The watch window (right apart from the initialization) allows watching of variables and expressions during the online simulation.

Line by line the controlled expressions can be indicated. As soon as the editor - by a break point – rises up during simulation, the appropriate value appears in each line apart from the indicated expression.

Only detectable (scalar) expressions can be evaluated. It is to be noted absolutely that the evaluation from expressions can lead to so-called side effects: If one indicates e.g. "subtable(x).execute", then with every reach of the break point the appropriate subtable is executed. If the table changes thereby variables, the simulation run with break points differs from the simulation run without break points. This is usually intended and leads to problems with the error analysis, which are difficult to understand.



16.6 Structure View

The tree view in the upper left corner of the dialog supports arrangement of decision tables in sub-tables and modeling of special behavior e.g. initial actions.

The structure view can be used like the directory tree in the Windows Explorer. One can select and draw entries with the mouse, in order to copy or move it (*drag & drop*). For each entry a context menu can be opened by the right mouse button.

Tables with subordinated tables are emphasized by "+" at the left of the name. A click on this "+" opens a hierarchy level of the tree (to see the sub items). "+" becomes then "-". Repeated clicking closes the hierarchy level.

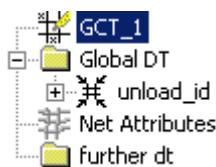


Figure 16.7: Tables

The first entry is always the decision table, on which the DT dialog was opened. Below this entry in **Global DT** the global decision tables are listed.

The last entry **further dt** contains all other decision tables of the model. These can also be reviewed or manipulated directly.

To each decision table the coordinates will be administered. With each call the position and the structure of the decision table are stored. To transfer these coordinates with the change between two decision tables, this can be achieved by pressing the SHIFT or CTRL key. With pressed SHIFT - key only the position is transferred, while with pressed CTRL key additionally the arrangement and size of the sub windows will be transferred.

16.6.1 Type of Tables

Fundamentally 5 table types can be differentiated. These are activating - tables (like GCT, working time DT, restriction DT, etc.), the global DTs (not bound to any module), the sub decision tables, as well as initial and final decision tables.

Activating-DT:

Activating-DTs are activated by events in a module or the transport control. A GCT DT for example is processed with each event in one of the activating modules, a working time DT whenever the working time of the module is determined. Activating DTs can contain *initials actions* as well as *sub-tables*.

Global DT:

Global decision tables contain general functionality. For global DTs there is no independent activation - they are called by other tables (by the action **global_dt(name).execute**). Global DTs are administered in the file **Global DT**. A global DT can contain both *initials actions* as well as *sub-tables*.



Parameters can be used when calling a global decision table. These can be used further in the conditions and actions of the global decision-table. The use of global variables can thus be reduced. The parameters have no data type and will be checked at runtime. The number of parameters is, however, controlled the consistency check. For an example, see the section **execute decision table**.

Sub Decision Table:

A sub decision table (sub DT) contains functionality of the super ordinate decision table. By the sub DT it will be possible to keep decision tables small and arrange them. Sub DT can be processed only by the action **subtable(name).execute** of the super ordinate table (main DT). Sub-tables can contain *initials actions*.

Initial Decision table:

All initials decision tables are processed with the start of the simulation. Initial decision tables can not contain any other table type.

Final Decision table:

All final decision tables are processed at the end of simulation. Final decision tables cannot contain other table types.

16.6.2 Creating Decision Tables

In order to create decision tables, first an entry in the table tree must be selected. In order to create a new global DT, one selects the file *global DT*. One creates sub-tables and initials decision tables by clicking the super ordinate DT. Module referred DT cannot be created in the dialog. These are created, depending upon type, by the modeling mechanisms of DOSIMIS -3.

After selection of the super ordinate DT one opens the context menu with a right-click and selects **add initial decision table**, **add final decision table** or **add DT**. This adds a sub-table to a main DT, in the directory *global DT* however a new global DT.

These operations can be activated by the menu of the DT.

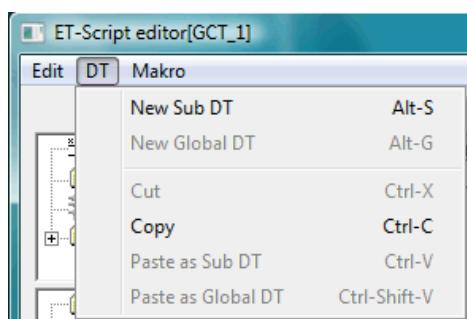


Figure 16.8: Menu DT

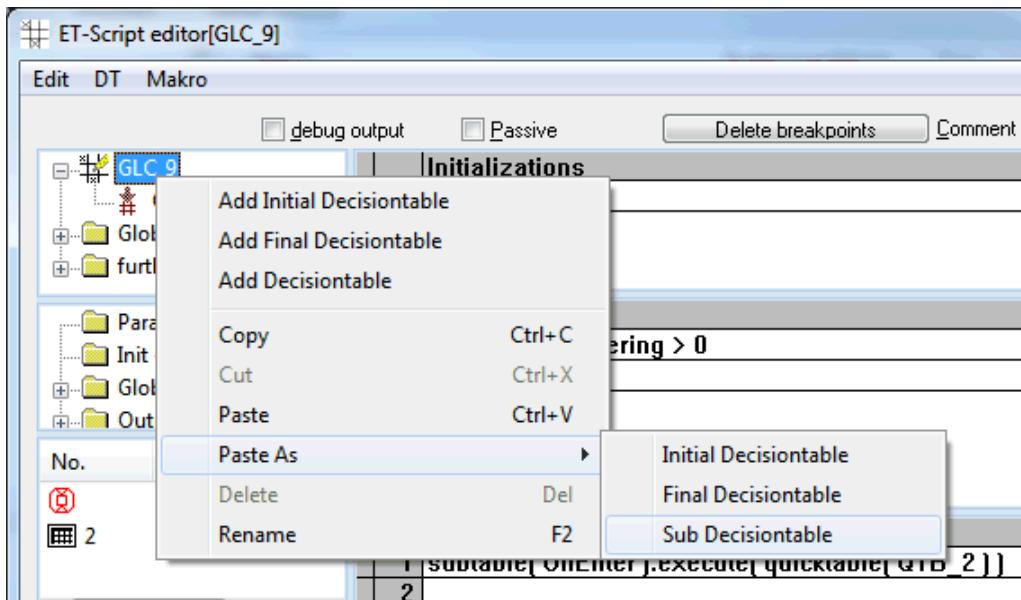


Figure 16.9: Menu DT

A copied decision table is inserted in the normal case as sub-decision table. However, if this is to be used as initial or final decision table can be determined by the selection of **paste as**.

16.6.3 Delete Decision Tables

One deletes decision tables or complete sub trees by a click on an entry and the selection of **delete** in the context menu. The file *global DT* cannot be deleted. Deleting the net attributes is preserving the (now emptied) table of *net attributes* in the tree.

16.6.4 Rename Decision Table

The function **rename** in the context menu of an entry enables the simple renaming of tables. It is not possible to rename the file *global DT* or the *net attribute* table.

16.6.5 Copy Decision Table

One can copy tables within the hierarchy levels of the tree. Here if necessary the type of the table is adapted. One copies by selection of **copy** from the context menu of the table. Subsequently, one selects the target of the copying action and clicks in the context menu on **paste**. Basically the entire subtree is copied.

The following operations are possible:

- a module DT can be made a global DT (copy to the file *global DT*).
- a module DT can be made a Sub DT (copy into a valid table).
- a global DT can be made a Sub DT (copy into a valid table).
- a Sub DT can become a global DT or a Sub DT of another table.
- a global DT can be copied as new global DT again into the file *global DT*.

16.6.6 Move Decision Table

Decision tables are movable with a similar mechanism. From the context menu one selects first **cut** and afterwards in the context menu of the target table **paste**. For moved decision tables the same rules to apply as with copying.



16.6.7 Copy/Move per Drag & Drop

Decision tables can also be copied without context menu. In addition one accesses the decision table with the mouse (to click and leave mouse button pressed) and drags the table with pressed mouse button to its destination. When releasing the mouse button the table is copied to its target. If one drags a table with the right mouse button, a selection menu appears, in which one can determine whether the table is to be copied or moved.

16.7 Variable view

The tree view within the middle left range of the editor helps with the definition of variables and attributes.

The variable tree can be used like the folder view of the windows Explorer. For each entry a context menu can be opened by pressing the right mouse button.

Variables with subordinated lists are emphasized by a "+" at the left of variable names. Click on this "+" opens a hierarchy level of the tree (the sub entries are shown). The "+" becomes now "-". A further clicking closes the hierarchy level.

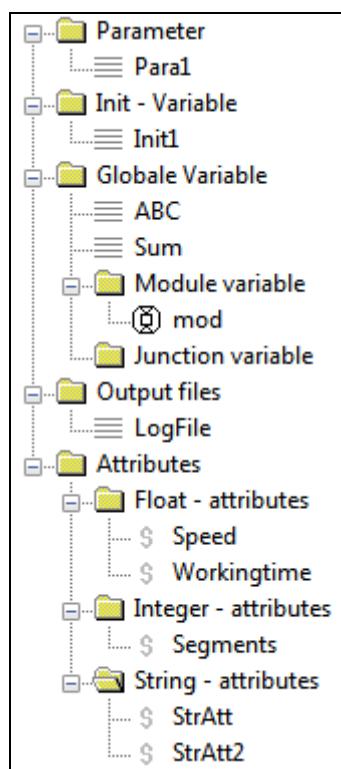


Figure 16.10: Variables

16.7.1 Types of Variables

In a similar manner as programming-languages do, so do decision tables distinguish between local and global variables. The global variables are valid for all the decision tables of a model. The usage of Local variables is restricted to the table that they are defined in.

On the associated context buttons **add** and **delete** existing variables may be created or deleted.



The names of variables are restricted in the selection of the signs. They must begin with an alphabetic character and may exist only of alphabetic characters, digits and the underscore character ('_'). Additional characters (for example ö, ä, ü) are not allowed (they must correspond with the regular expression [a-zA-Z][a-zA-Z0-9_]*).

16.7.1.1 Lokale Variablen

Local variables are also called **Init-Variable**. If these are defined in the main decision table, then these are valid also in all sub-tables. Otherwise their visibility is limited to the table, in which they are defined. They are not even valid for the sub tables of this table.

In some types of decision tables, in addition to the initialization variables and **parameters** can be declared. These can be passed when calling the execution of the decision table. Therefore, only global decision tables and sub-decision tables support parameters.

16.7.1.2 Global Variables

Global variables are available in the simulation model globally, that means in all decision tables. In the folder **Global Variables** of the variable view the currently defined global are shown. Global variables are initialized with the value 0. To change the initial value of the variables, please create a initial decision table. There you can assign a value to the variable in the initialization window (**variable: = 1234**).

Note: When reading old models the net attributes are interpreted as follows. The attribute name is added in the list of global variables. The initialization of the attribute is added to the initial decision table of the first global decision table. If these do not exist, they will be generated automatically.

16.7.1.2.1 Module Variable and Junction Variable

The definition of module variables occurs with the *global DT dialog*. Names are defined for this, which can be used in the definition of decision tables. References to modules may only be assigned to the component variables. For example references can be transferred also on modules to global decision tables. For junction variables above mentioned applies in analogy.

16.7.1.3 Integer-, Float and String Attributes

Attributes are names, which can be assigned to objects and modules. By these names values can be assigned via decision table to the object (module), which is usable in the further simulation process. Attribute names are always defined globally, that means they can be used in any decision table. Integer attribute can take only integral values, floating attributes however real values, and string attributes can take text.

As an example, assume an attribute with the name **processingtime**, which can be used after the definition with objects and / or modules. The following action set the value of the attribute **processingtime** of the first object in the actual module to the value of 60:

`act_module.f_obj.intatt.processingtime := 60.`

One use is the assignment of values to objects or modules that are retained during the simulation. So that information can be managed through meaningful names and are available at any point of the model within decision tables.

A further advantage with the use of attributes consists of the fact that with many parameters in modules these attributes can be referenced directly. This happens, by the prefix \$ followed by



the attribute name. If this parameter must be evaluated now during the simulation, this proceeds as follows. If the last object possesses that indicated attribute, this value is used, otherwise that of the module. If either the inquiries fails or the return value smaller or alike zero, an error message takes place.

If the attribute of the first object should be used for determining, this can be changed in the properties of the [Decision tables](#).

16.7.1.4 Output File

After defining an output file this can be used during the processing of a decision table for text output into this file. Therefore the command **file(name).print** or **datprint** exists. The file is defined by the given *name*. At the end of the simulation the data is placed in that file, which can be found in the project directory and is ending on *name.log*. Output files are always global, so they can be accessed in every decision table.

A typical action for writing an information into a output file *testfile* is:
`file(testfile).print("Destination = ", act_module.l_obj.destparam)`

16.8 References

No.	References
①	act_module
② 8	TRC_8
③ 7	CRS_7
④ 18	TRC_19 -> CRS_4

Figure 16.11: References

In reference to the credentials of the currently selected window decision table are displayed. The following types of references are:

- Activating module
Activating modules are modules that lead to the activation of the respective decision table. These are marked with a red module icon.
- Reference modules
Reference modules are modules that are related to the appropriate decision table, but they are no activating modules. Typically these components, one of which can be interrogated either values such as the actual occupancy or to act the actions, such as setting an attribute for a module. These modules are shown with a blue module icon.
- Reference junction
Reference junction are (corresponding to the reference modules) junctions that are used within the decision table. They are represented by a junction icon.
- Activating timer
Activating timer are timer that lead to the activation of the decision table. They are represented by a red timer icon.
- Reference Timer
Reference timers are (according to the reference module) timers that used in the decision table (referenced). They are represented by a blue timer icon.
- Quick Table, signal display, ...
- All other controls are represented by their symbol. These elements do not lead to activation of the decision table.



The reference window enables to select modules which one would like to access during the work with the decision table. Further on it supports the name and number for every element.

With creating a decision table only the activating modules are found in the reference window. This is always *act_module* and in the case of a GCT decision table the activating modules, which were assigned in the mode *Linking active*.

When in the decision table other modules (or junctions) are to be accessed, then this takes place basically by the number of the module.

The function **module(139).cap** would supply the capacity of the module 139. Since however with complex models the module numbers might hardly be well known, the reference window offers an important assistance, since the names of the modules are to be seen here.

The definition of activating modules for GCT tables can be made by changing to the mode *Linking active*, selecting the GCT module and linking as many as desired activating modules.

16.8.1 Definition of References

Adding references takes place by the menu **Edit/References** or by the context menu in the reference window. The reference dialog, which represents the Dosimis-3 model, appears and offers fundamental view-functions such as *zoom model* or *zoom in*.

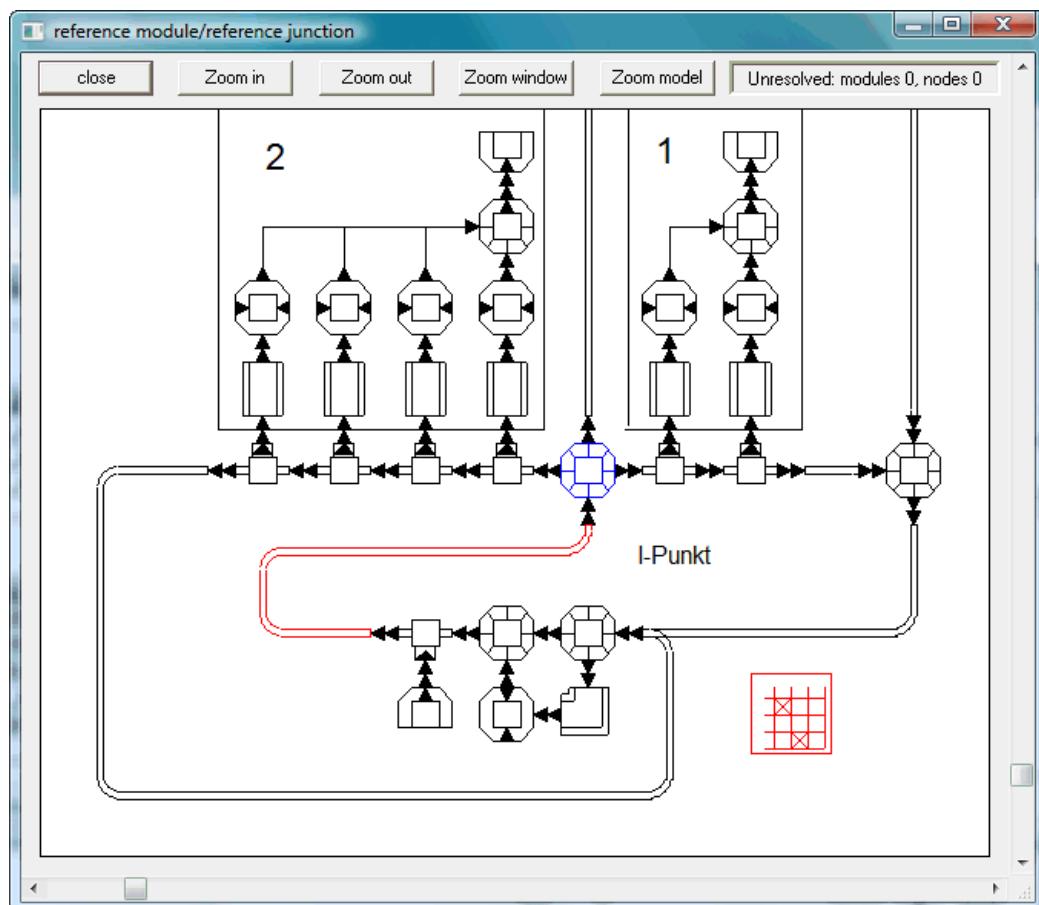


Figure 16.12: Reference Dialog



In the reference dialog the activating elements are represented in red, the reference elements are represented in blue and all other elements are represented with black color.

In reference dialog you can select and unselect DOSIMIS-3 elements and junction which then are represented in blue (is reference) or black (no reference). Here, the tooltips helps to identify the elements. Place the cursor at the element and wait awhile, then the associated information according the element raises up. After pressing *Close* the references appear in the reference window. Activating elements cannot be changed in this dialog.

16.8.2 Deletion of References

References are normally saved together with the decision table. If a reference is to be removed from the list of the reference modules, this happens by clicking on the module. Replacing of the unselected module in the formulas is then no longer possible.

16.8.3 Exchange of References

In order to exchange or subsequently define reference modules (e.g. after deletion of a module), you proceed as follows. First the module to be signed out must be deselected while pressing the shift key. This is confirmed with a message in the field on the upper right.

Unresolved: modules 1, nodes 0

Figure 16.13: Information about missing reference modules

The number of undefined modules increases. Now one selects a new reference module. Now a message appears that the reference module is exchanged.

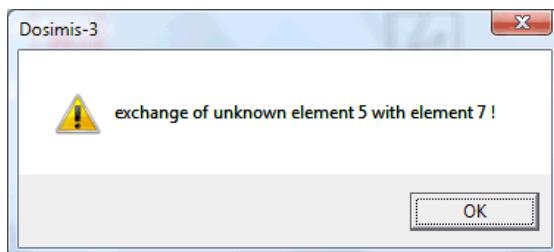


Figure 16.14: Message after the exchange of reference module

All lines of the decision table, which refer to the old reference module, are now updated.

16.8.4 Definition of Activating Modules

The definition of the GC modules (activating module) takes place as in the case of every control elements in the connecting mode. (See chapter: *Linking of controls with elements* in the user manual)

16.8.5 Definition of Activating Timer

The definition of activating timer takes place as in the case of every control elements in the connecting mode. (See chapter: *Linking of controls with Elements* in the user manual)



16.8.6 Converting of Reference Modules to Activating Modules

To convert reference modules into activating modules, it is sufficient to assign these in the connecting mode to the list of GC modules (See chapter: *Linking of controls with elements* in the user manual). This functionality only makes sense in the case of GC decision tables.

16.8.7 Usage of References

During the input of initializations, conditions or actions you can insert these into the formula by a double-click on a reference. Double-click on BLK_8 in Figure 16.11 inserts the text **module(8)**, double-click on a junction (e.g. junction 1) accordingly **junction (1)**.

16.9 Call Stack

The call stack makes it possible to see the call path of the current DT during online simulation. When a breakpoint was triggered in an DT, you can examine all parent DT with their condition and actions.

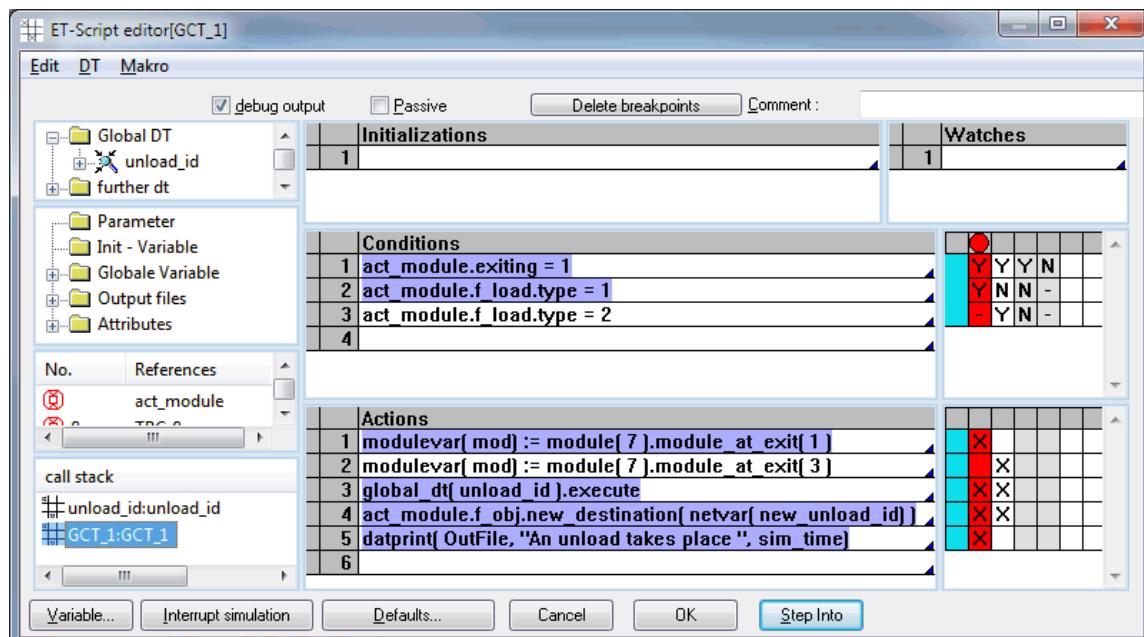


Figure 16.15: Call stack invoked by breakpoint

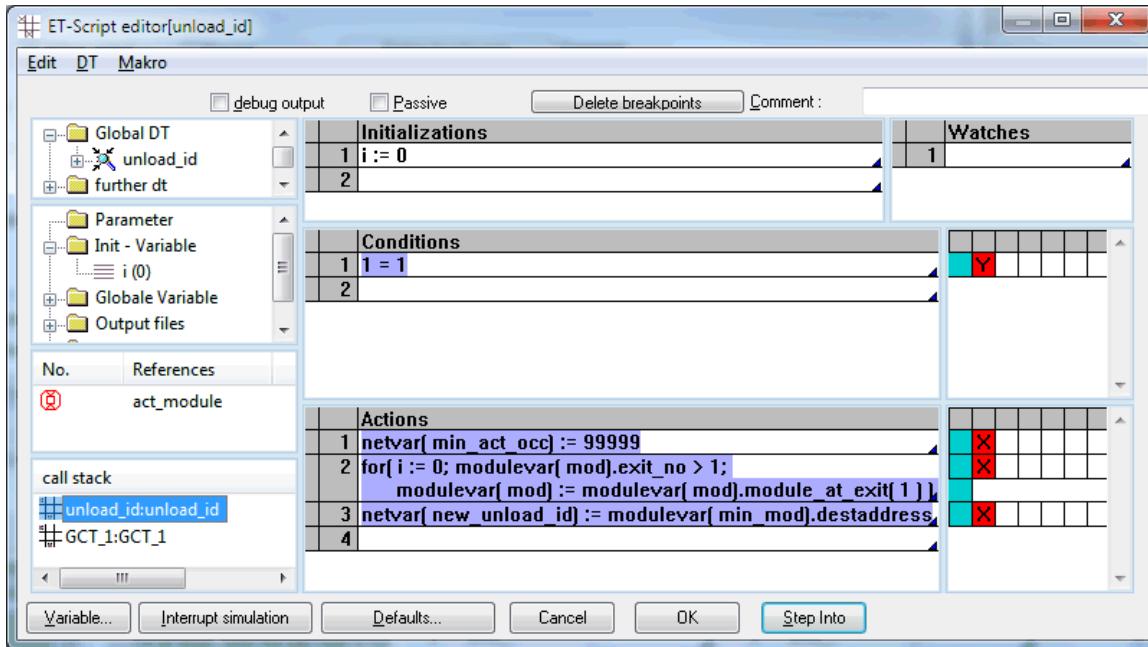


Figure 16.16: Call stack selection by path

The last two figures are to be taken as an example. A DT is interrupted by a breakpoint. The call stack shows the call path. The first entry is always the DT, which was executed last. Directly below is the DT that called the top DT. By double clicking on the list items you can switch between the different DT. The conditions, actions and rules, which led to the call, will be highlighted.

16.10 Macros

Often recurring actions are supported by macros. For this the decision table where the macro is to be implemented, must be selected in the structure view.

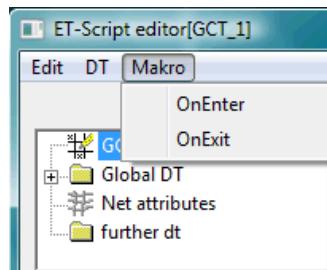


Figure 16.17: Menu macro

The two macros create a new sub - decision table and integrates the call into the main decision table. Thus an additionally new condition is defined, where the event entering/exiting is queried, and an action is defined, in which the decision table call is set.

16.11 Terminating Input

As long as no changes have been made in the decision table, click *cancel* to leave the dialog. Alternatively the Escape key can be pressed.



Through a click on OK in the bottom area of the dialog the handling of the decision table terminates. Now a fundamental *consistency check* takes place. Only if this check runs perfectly, the decision table dialog is closed.

A possible error during the input of decision tables is the use of ambiguous rules. These are two (or more) rules, which lead to same result of the conditions to actions. Such rules are red emphasized after the consistency check and must be corrected.

conditions					
1	act_module.exiting = 1		Y	Y	Y
2	act_module.f_load.type = 1		Y	N	Y
3	act_module.f_load.type = 2		Ø	Y	N
4					

actions					
1	modulevar(mod) := module(7).module_at_exit(1)		X		
2	modulevar(mod) := module(7).module_at_exit(3)			X	
3	global_dt(unload_id).execute				X
4	act_module.f_obj.new_destination(netvar[new_unload_id])				X
5	datprint(OutFile, "An unload takes place ", sim_time)				X
6					

Figure 16.18: Error message with ambiguous rules

If condition 1 AND condition 2 do apply AND condition 3 does not apply, then this leads both to rule 1 as well as to rule 3 (red), which means a contradiction. Such *ambiguous* rules are not admissible in DOSIMIS -3.



17 Further Components

Beside the elements, which can possess a decision table and which there are *global DT data* still further elements exist, with which decision tables can interact.

17.1 Animation text



Animation text allows displaying of text in the model layout. Besides the representation of steady text, the displayed text can be changed during simulation run time by decision tables. In addition, the text can also contain expressions and formulas whose values are dynamically displayed during the simulation run. The animation texts can be faded in or out also with the help of the menu. This takes place by *model/view/animation text* or the keyboard contraction *CTRL+F5*.

To create an animated text, the corresponding icon (symbol ATX) are selected in the Control palette. By clicking the parameter dialog of the new animation text appears. After configuration is complete, the animation text can be placed anywhere in the model layout.

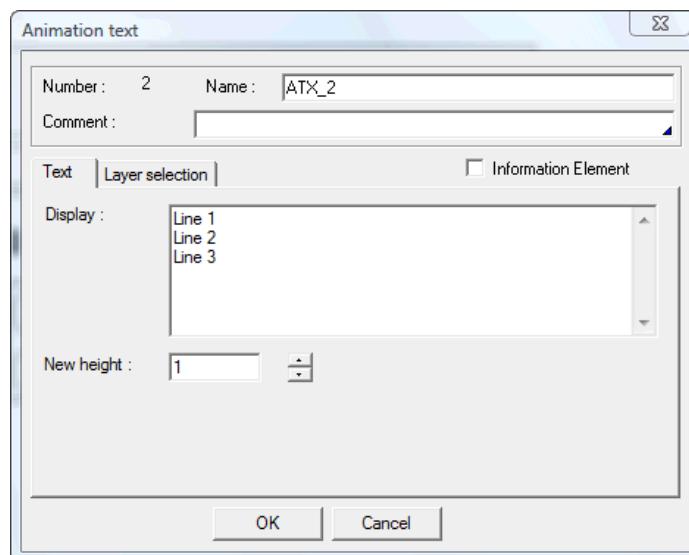


Figure 17.1: Parameter of animation text

- Edit field **Name**

The edit field **Name** allows you to enter an identifier for the animation text. With this name the text can be addressed in a decision table. A typical command might look like this:

```
aniprint(ATX_1, "current occupancy =" act_module.actocc)
```

- Edit field **New height**

Here the size of the font can be changed. They are the values between 1 to 8 allowed.

- Edit field **Display**



In the Edit field **Display** defines what is displayed during the simulation run in the model layout. In the simplest case, this is a constant text that can also span multiple lines. A line break is inserted by pressing Ctrl and Enter. As shown in the screenshot below, you can also formulas or expressions appear.

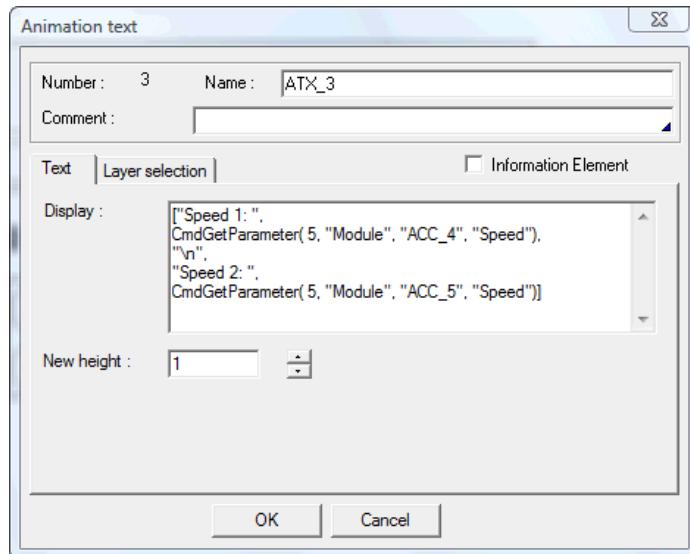


Figure 17.2: Parameter of animation text

In the *display* also formulas or other complex expressions can be printed. For this the text must be set into square brackets ([]). (note: Before the opening bracket and after the right bracket no further indication may be, also no *carriage return / line feed*). Then the following parameters are interpreted similarly as in a print instruction of the decision tables.

Line feeds are however not transferred here directly to the output. In order to let the output text continue at certain places with a new line, the string “\n“ must be entered.

In the display also module parameters can be called up. For this the function *CmdGetParameter*(field, par1, par2,...) is provided.

17.2 Quicktable



A Quick Table is a 2-dimensional table with rows and columns. It enables storage of mass data in tabular form, where in the cells either integers, decimal numbers or strings are stored. Can use a Quick Table:

- External data can be read at the beginning of the simulation, for example, Load data, order data or parameter data
- During the simulation calculated data stored, read out again and changed
- During the simulation calculated data are stored after the end of the simulation in a file, for example Statistics such as type counts

The following figure shows an example.



	0	in Simulation	Aufwand Plan...	Überplan[h]	Aufwand Plan...	Überplan[h]	Aufwand Plan...	Überplan[h]
0	Bremse	2	2	3,33	1,2	1,02	0,85	0
1	EV	2	2	3,33	1,2	1,02	0,85	0
2	Innen	3	3	5	1,8	1,54	1,28	0
3	Klima	1	1	1,67	0,6	0,51	0,43	0
4	Küche	1	1	1,67	0,6	0,51	0,43	0
5	MSPA -Prüf	1	1	1,67	0,6	0,51	0,43	0
6	Türen	1	1	1,67	0,6	0,51	0,43	0
7	Unterbau	3	3	5	1,8	1,54	1,28	0
8	Wasser	1	1	1,67	0,6	0,51	0,43	0
9	Summe:	15	15	25	9	12,8	6,4	9

Figure 17.3: Example of a quicktable

To create a Quicktable, the corresponding icon (shortcut: QTB) has to be selected in the control palette. Click once and move the cursor over the model layout, the Quicktable can be placed anywhere in the layout. Double-clicking the icon in the layout opens the associated parameter dialog.

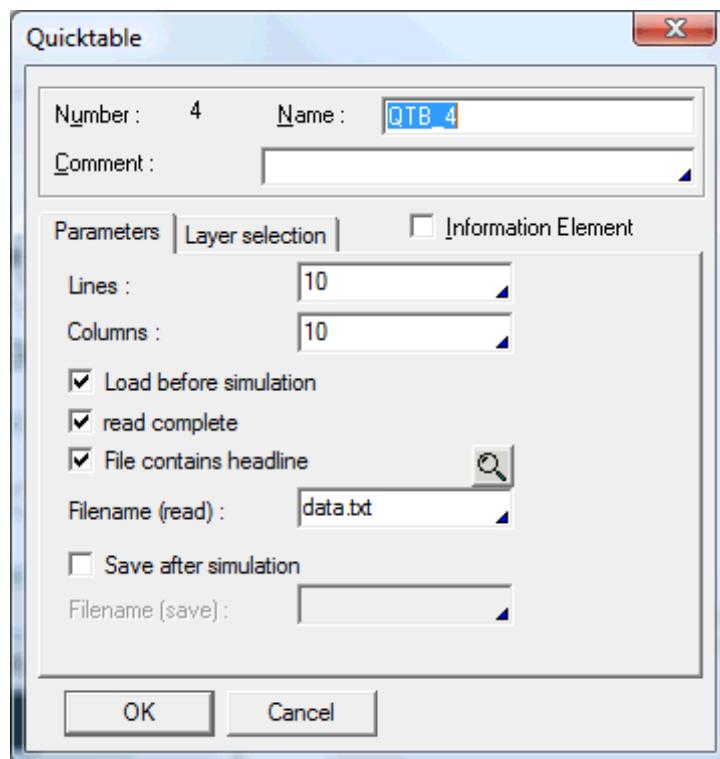


Figure 17.4: Parameter of a quicktable

- Edit field **Name**

In this field the identifier of the Quicktable can be specified. With this name the Quicktable can be addressed from a decision table. By decision table the values of a quicktable can be set and read out and also look for and quicktables can be sorted by line.

- Edit field **Lines** and **Columns**



A quicktable is indicated by the number of **Lines** and **Columns**. The number of lines can be changed however by methods of the quicktable during the simulation.

When the Quicktable is initialized before simulation start with the contents of a file, the **number of lines** can be specified **as 0**. Then the actual number of rows is determined when reading the file dynamically.

- Edit field **Load before simulation, read complete, file contains headline**

If the table is to be initialized before the start of the simulation, this is done by activation of **load before simulation** and specifying the file name in the **Filename (read)**. The file must be in the project directory. If **load before simulation** is enabled, the Quick Table is empty at the start of the simulation. If the switch **read complete** is set, the specified file is completely read into the table at start simulation. If the switch is not set, only the first line of data is read (as header). Reading of these rows can then e.g. be controlled via the command *ReadLine* by decision table. The switch **file contains headline** affects how the first file line is treated. If the switch is not set, all file lines are interpreted as rows of data and is read from row 0 in the Quick Table. If the switch is set, the first line is interpreted as a file header and not as the first data row of the Quicktable. The first line will be used in the header area of Quicktable as column headings. This approach allows accessing the columns of Quicktable not only through the column no. but to by the corresponding column name.

17.2.1 Display

Quicktable can be indicated in DOSIMIS-3. Therefore the entry *execute* is to select from the context menu. If an initialization file is assigned to the table, their display takes place in its own window. The size of the table and the measure can be adapted to the fields.

17.2.2 Data Format

External tables, which are to be read in before beginning of simulation, must have the following characteristics. The data must be present in ASCII - format. The fields must be separated with tabulators. As decimal separators comma or point is permissible. Do not use binary formats like excel sheets or word documents. As a control, the file should be opened with the default Windows Notepad (Notepad).

Operations to the quicktable can be found in the language reference.

17.3 Monitoring



The monitoring module serves an easy way, to see changes of values during simulation. This can be for example the occupancy or throughput of a module, also changes in the value of a module variable. At each change of the value a corresponding output is put into the log window.

To create a monitoring module, the corresponding icon (shortcut: MOR) has to be selected in the control palette. Click once and move the cursor over the model layout, the monitoring module can be placed anywhere in the layout. Double-clicking the icon in the layout opens the associated parameter dialog.

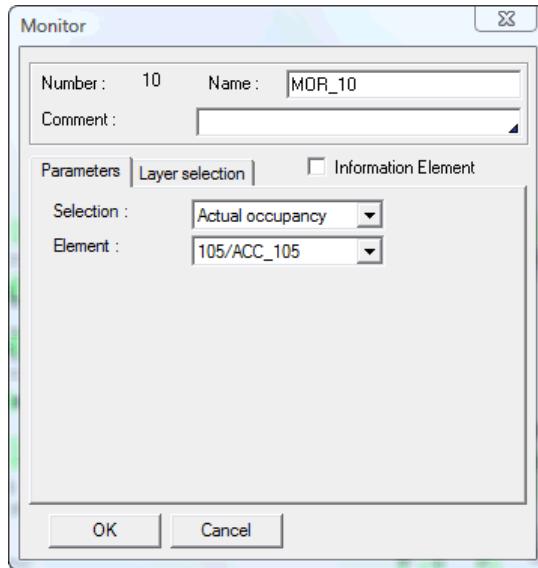


Figure 17.5: Parameter of the monitor module

- **Edit field Name**

The edit field **Name** allows you to enter an identifier for the monitoring module.

- **Selection box Selection**

The value of this field determines the type of value to be controlled. The choices are: **global Variable**, **module variable**, **junction variable**, **occupancy**, **throughput** or **Junction**

- **Selection box Element**

The choices offered depend on the current value of the selection box **Selection**. The following options are available:

Selection = Global variable: List of all global variables

Selection = Module variable: List of all der module variables

Selection = Junction variable: List of all der junction variables

Selection = Occupancy: List of all modules which support occupancy

Selection = Throughput: List of all modules which support throughput

Selection = Junction: List of all junctions in the model

Notes:

The output does not take place during the prerun. Furthermore the output is suppressed if the switch ET trace in the simulation parameters is not activated. To change the switch, first select the item *parameter* from the simulation menu. In the register Action protocol you will find this switch.

17.4 Timer



The timer serves to call the evaluation of e decision table.

To create a timer module, the corresponding icon (shortcut: TIM) has to be selected in the control palette. Click once and move the cursor over the model layout, the timer module can



be placed anywhere in the layout. Double-clicking the icon in the layout opens the associated parameter dialog.

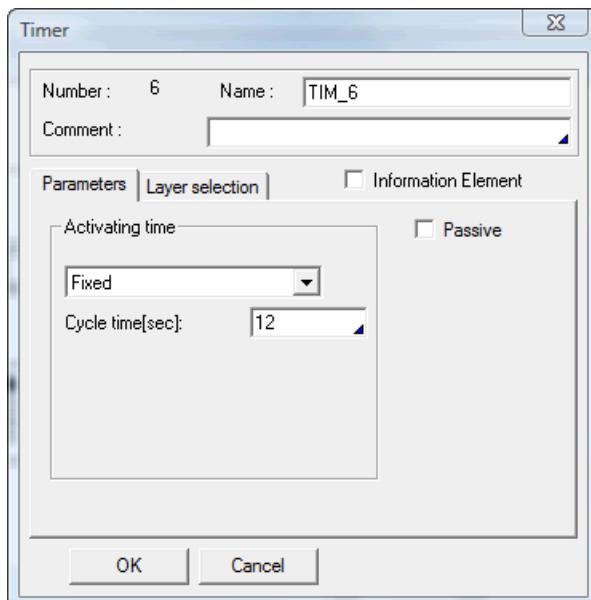


Figure 17.6: Parameter of the timer

- **Edit field Name**

In this field, the identifier of the timer can be specified. About this, the timer can be addressed from a decision table. By decision table can e.g. the next expiration interval of the timer can be set. Or it can be queried whether the timer has expired.

- **Edit area Activating time**

The activation time defines the time period (= the distance) between two activations of the timer. To choose from in addition to a variety of stochastic functions is also a fixed-timed function. Once the specified period elapses, all decision tables are activated, which are connected to the timer.

Is the time of activation parameters with **none**, the timer will turn off activations. This kind of parameterization is provided to activate the timer by actions in a decision table Command: Timer (TIM_X) activate (Time_in_SEC)

- **Switch Passive**

When the switch **Passive** is set, the timer module will not trigger any activation, that means no evaluation of decision tables.

Operations and methods of the timer are described in the language reference.

17.5 Signal Display



The signal display serves to emphasize situations colored in the layout. For this 4 circular fields are available, which can be filled by operations with individual colors.



To create a signal display, the corresponding icon (shortcut: SIG) has to be selected in the control palette. Click once and move the cursor over the model layout, the signal display can be placed anywhere in the layout. Double-clicking the icon in the layout opens the associated parameter dialog.

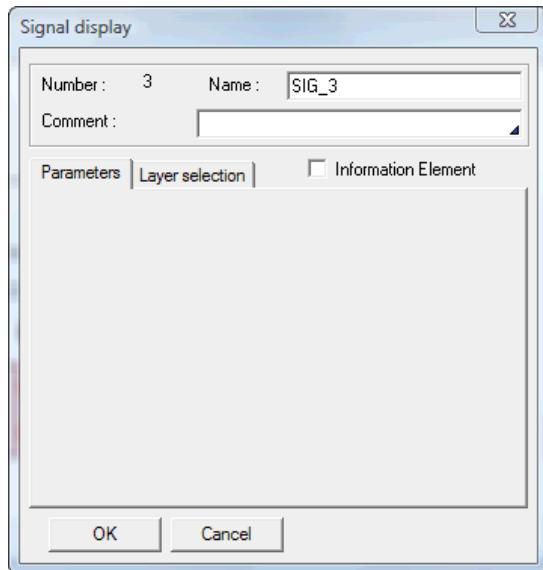


Figure 17.7: Parameter of the signal display

- **Edit field Name**

In this field the identifier of the signal can be specified. With this name the signal can be addressed from a decision table.

The command is: Signal(SIG_X, Circle-No., red, green, blue)

The description of the operations to the signal display is in the language reference.

17.6 Progress indicator



The progress indicator serves to point out progress or filling degrees in the layout by colors. For this a rectangular drawing area is available, which can be filled with colors by methods of the decision tables.

To create a progress indicator, the corresponding icon (shortcut: PGI) has to be selected in the control palette. Click once and move the cursor over the model layout, the progress indicator can be placed anywhere in the layout. By the position of the second mouse clicks can be determined how big (long), the progress indicator should be and whether it should be placed horizontally or vertically. Double-clicking the icon in the layout opens the associated parameter dialog.

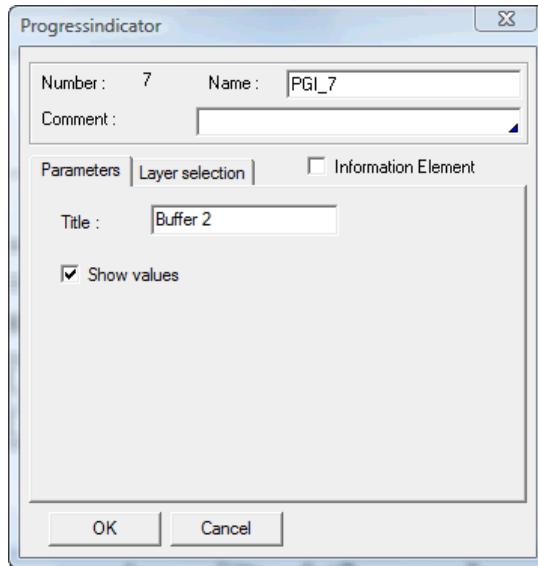


Figure 17.8: Parameter of the progress indicator

- Edit field **Name**

In this field the identifier of the progress indicator can be specified. With this name the progress indicator can be addressed from a decision table.

The command to change the represented value of the progress indicator is:

Signal(PGI_X, Perc.-Value, red, green, blue)

- Edit field **Title**

Text entered here is added to the progress indicator as the title.

- Switch **Show values**

If the switch **show values** is set, the represented values are indicated additionally in text form.

The description of the operations to the progress indicator can be found in the language reference.





18 Language Reference

Keywords are usually used in selector chains. Such a chain is e.g. **act_module.f_obj.real.a**. The components of such a selector chain are called also selector. Not each keyword is automatically a selector. Example: The keyword **round**.

Each selector defines a quantity of keywords, which may be attached to it. These keywords are the *attributes* and the *operations* of the selector and can be again selectors. For the simplification in the following one the word *attributes* is used.

This structure may appear complicated, but it is however relatively simple: A module selector leads to module attributes, an object selector to object attributes,

Many keywords can be represented either like a selector (act_module.actocc) or a function call (actocc(act_module)). This is admissible however only for compatibility reasons and should not be used. The first way of writing is to be preferred not at least because of the context sensitive editing support.

First some examples of admissible inputs:

- **act_module.actocc**
Selects the activating module and supplies its current allocation
- **module(15).f_obj.type**
Selects the first object in the module 15 and supplies its type. Here one sees, how selections can be combined. First one selects a module, then an object in this module, then a characteristic of the object **f_obj** is thus an attribute of the module, **type** an attribute of the object. The keyword **module** expects the module number as parameter. One could use this e.g. in a condition. This could read:
- **module(15).f_obj.type=70**
With this condition it is checked whether the type of the first object in the module 15 has the value 70. By the rule matrix it could be determined, which actions are supposed to be executed in this case.
- **module(2).module_at_exit(1).junction_at_exit(2).lock**
this is an example of an action. First module 2 is selected. Subsequently, the module, which is placed at its output 1. Thus one can move in this way through the model. From this module we select the junction at its output 2 and lock it.

Essentially a partitioning can be made into 4 areas:

- Basic elements
- Module attributes
- Object attribute
- Junction attribute



18.1 Types

In the following reference these types are used:

int	an integer
real	a real number: Only dots are valid as a decimal separator.
bool	boolean value (true/false). Boolean keywords can be used directly (without comparison operation) as condition.
String	a text - Strings are indicated in quotes.
par	Any, representable parameter, thus a value of the type int , real or String .
Work	Reference to a task in DOSIMIS-3. Work is equivalent to the call for a worker for a task in a station or clearing a failure. - If a keyword is of this type, it selects a junction.
Module	a reference to a module - If a keyword is of this type, it selects a module.
Junction	a reference to a junction - If a keyword is of this type, it selects a junction.
Object	a reference to a DOSIMIS-3 Object - If a keyword is of this type, it selects an object.
Quicktable	Reference to a DOSIMIS-3 Quicktable - If a keyword is of this type, it selects a Quicktable.
Worker	A reference to a DOSIMIS-3 worker- If a keyword is of this type, it selects a worker.
Work step	A reference to a DOSIMIS-3 work step.
Work list	A reference to a DOSIMIS-3 work list
Order	A reference to a DOSIMIS-3 order
Timer	A reference to a DOSIMIS-3 Timer
Anitext	A reference to an animation text
File	A reference to an output (log-) file
Transport-system	A reference to a transport system.
tsorder	A reference to an (transport-) order of the transport system.
Sensor	A reference to a DOSIMIS-3 Sensor, e.g. at an accumulating conveyor2
Var	a variable name - This can be the name of an initialization variable, a net attribute, an animation text, etc., depending upon the context. Variables are specified without quotation marks.
Operation	An operation - This type is to be regarded more than note. Operations do not return the values, with exception of the operation call .



18.2 Operator

•

Function	Operator	Example	Usage
Less than	<	<i>Act_module.actocc < 5</i>	<i>Condition</i>
Equal	=	<i>Act_module.entering = 1</i>	<i>Condition</i>
String compare (case sensitive)	=	„abc“ = „ABC“ (result: FALSE)	<i>Condition</i>
String compare	~=	„abc“ ~= „ABC“ (result: TRUE)	<i>Condition</i>
Greater than	>	<i>Act_module. actocc > 12</i>	<i>Condition</i>
Less equal	<=	<i>initvar(a) <= initvar(b)</i>	<i>Condition</i>
Greater equal	>=	<i>netvar(a) >= netvar(b)</i>	<i>Condition</i>
Not equal	!=, <>	<i>Act_module.integer.a != 1</i>	<i>Condition</i>
Relation	??	<i>A ?? b</i>	<i>Condition</i>
Logical and	&&	<i>A<5 && b>12</i>	<i>Condition</i>
Logical or		<i>A<5 b<12</i>	<i>Condition</i>
Logical not	!	<i>! a<5</i>	<i>Condition</i>
Assignment	:=	<i>Var := 15</i>	<i>Initializing/Action</i>
Assignment	:=, +=, -=, *=-, /=, %=	<i>var := 15</i> <i>var += 15 (var := var + 5)</i>	<i>Initializing/ Action</i>
Increment/ Decrement	++, --	<i>var++ (var := var + 1)</i>	<i>Initializing/ Action</i>
Addition	+	<i>X:= 5+27+ act_module. actocc</i>	<i>All, also text</i>
Subtraction	-	<i>Act_module.integer.a:= 19-</i> <i>act_module.cap</i>	<i>all</i>
Modulus operator	%	<i>X := a % b (reminder of a division)</i>	<i>all</i>
Multiplication	*	<i>X := a * b</i>	<i>all</i>
Division	/	<i>X := act_module.cap / act_module.actocc</i>	<i>all</i>
Priority	(...)	<i>X:=(5+4)*20</i>	<i>all</i>

Explanation: The comparison operators (=,<, >, <=, >=, <>, ~=, ??) supply logical values (Y/N), according to the result of the comparison. A special case represents the relation operator, which can be on behalf for any comparison. Only in the rule matrix one determines, which comparison it concerns. It is recommended not to use this operator. It is only available for reasons of compatibility.

The logical operators (**&&**, **||** and **!**) serve the linkage of predicates. The **&&**-operator supplies only true (Y), if both predicates are true. The **||**-operator supplies true if one of the predicates applies. The **|**-character is read "Pipe". The **!**-operator supplies true only if the following predicate is false.

The assignment operator (**:** =) assigns a value to a variable.

The basic arithmetic function operates as usual. It applies for "point before line", which can be removed by the use of brackets. Ebenso gilt für die logischen Operatoren „**&&** vor **||**“.

The addition works also with strings. 2 variants are to be differentiated.



- If 2 strings are added, then the strings are concatenated.
Example: “Text1“+“Text2“(=“Text1Text2“)
- If a string and a number is added, then a string is created, which is the original string followed by the text that represents the number.
Example: “Text1“+ 5 (=“Text15“)

18.3 String

The decision table supports also operations on string - variables. On the one hand the addition of strings is supported. 2 variants are to be differentiated.

- If 2 strings are added, then the strings are concatenated with one another.
Example: „Text1“ + „Text2“ (=„Text1Text2“)
- If to a string and a number is added, then a strings results, which exists followed from the original string from a text, which represents the number.
- Example: „Text1“ + 5 (=„Text15“)

Additionally there are still methods, in order to manipulate strings.

- **Find(String text, String search, int start)**

Type: *int*

Gets the position, at which the text *text* was found in the string *text*. Searching starts at the position *start*. The return value is -1, if the text is not found.

- **Left(String text, int cnt)**

Type: *String*

The function returns a string that contains a specified number of characters from the left side of the string *text*.

Example 1: str1 := “ABCDEFG“; str2 :=Left(str1, 3). Str2 contains „ABC“

- **Length (String text)**

Type: *int*

The function returns the number of characters of the string *text*.

- **Mid (String text, int start, int cnt)**

Type: *String*

The function returns a string containing a specified number of characters from a string.

- **Replace(String text, String old, String new)**

Type: *int*

Replaces the text *old* was by the text *new* in the string *text*. The function returns the number of replacements.

- **Right (String text, int cnt)**

Type: *String*

The function returns a string, which consists of the last *cnt* characters of the original string *text*.



18.4 Basic Elements

18.4.1 Selectors

The keywords specified here select a module, a junction or an object. These are however not the only possible selectors, since e.g. it can also be selected by variables.

- **act_work**

Type: work

Restrictions: Only in restrictions-DT of work areas

Results in the actual work to be assigned.

- **act_module**

Type: Module

Results in the activating module. In case of transport - DTs is this the module containing the vehicle (agv).

- **act_object**

Type: Object

Restrictions: Only in distribution DT

Results in the actual object, for which an exit is looked for. If in the module *stopping forbidden* is parametrized, the object has not entered the module yet.

- **act_worker**

Type: worker

Restrictions: Only in restrictions-DT of work areas

Results in the actual worker to be assigned.

- **module(int no)**

- **module (string name)**

- **module (var name)**

Type: Module

Results in the module with the number *no* or with the name *name*.

- **file (string name)**

- **file (var name)**

Type: File

Selects the output file, which was passed to the decision table by the parameter *para*. If the decision table has no parameter *para*, the output file will be searched by the name *para*.

- **anitext (string name)**

- **anitext (var name)**

Type: Animation text

Selects the animation, text, which was passed to the decision table by the parameter *para*. If the decision table has no parameter *para*, the animation, text will be searched by the name *para*.



- **act_agv**
Type: Object
Restrictions: Only valid in Transport - DTs
 Results in the actual transport vehicle.
- **junction(int no)**
Type: Junction
 Results in the junction with the number *no*.
- **object(var para)**
Type: Object
 Selects the objects that was passed by the parameter *para* to the decision table.
- **quicktable(int nr)**
- **quicktable(string name)**
- **quicktable(var name)**
Type: Quicktable
 Select a Quicktable with the number *nr* resp. name *name*.
- **timer(var name)**
Type: Timer
 Select a timer with the name *name*
- **act_timer**
Type: Timer
 Selects the timer that was responsible for the call of the decision table. *act_timer.no* is 0, if the DT were not called by a timer.
- **act_tsorder**
Type: tsorder
 Selects the current transport order that is to be scheduled.
- **act_sensor**
Type: Sensor
 Selects the sensor that was responsible for the call of the decision table. *act_sensor.no* is 0, if the DT were not called by a timer.
- **transportsystem (int nr)**
Type: Transportsystem
 Selects the transport system with the number *nr*. If *nr* = 0 is, the global transport control is selected.

18.4.2 Variables

Changeable values. There are different types of variables. Variables can be selectors. Not all types of variables can be assigned a value in a decision table (e.g. **intpar** and **floatpar**).

- **destinationpar(var name)**
Type: int
 Results in the value of the destination constant *name* from the Dosimis-3 default values.



- **floatpar(*var name*)**

Type: real

Restrictions: Read Only

Results in the value of the float constant *name* from the Dosimis-3 default values.

- **initvar(*var name*)**

Type: real

Results in the value of the initialization variables *name*. The value can be also assigned. Initialization variables are valid local variables, thus only within a DT. The initialization variables of the main decision table are well-known also in the Sub decision tables.

- **intpar(*var name*)**

Type: int

Restrictions: Read Only

Results in the value of the integer constant *name* from the Dosimis-3 default values.

- **junctionvar(*var name*)**

Type: Junction

Selected and results in the junction, to which the junction variable *name* refers. Also a junction can be assigned to the variable by this keyword. For this one uses the assignment operator. A junction must be assigned to a junction variable, before it is used for the first times as a selector.

Example: **junctionvar (abc) := act_module.junction_at_exit (1)**

- **modulevar(*var name*)**

Type: Module

Selected and results in the module, to which the module variable *name* refers. Also a module can be assigned to the variable by this keyword. For this one uses the assignment operator. A module must be assigned to a module variable, before it is used for the first times as a selector.

Example: **modulevar (xyz) := act_module**

- **globalvar(*var name*) (former netvar(*var name*))**

Type: real

Restrictions: Read Only

Results in the value of the global variable *name* (former net attribute). The value can be also assigned. Global variables are variables, which are global for all decision tables.

- **objecttypepar(*var name*)**

Type: int

Restrictions: Read Only

Results in the value of the object type constant *name* from the Dosimis-3 default values.

- **Selfdef (out of date: please use the operation *return*)**

Type: int / real

Restrictions: Not valid in GCT-DTs

One can assign a return value of the decision table to this variable. This is e.g. necessary with a distribution DT, which must determine an exit (self defined distribution strategy).



One deposits the selected exit here. By *selfdef:= 3* the exit 3 will be selected (please use the equivalent operation *return (3)*).

- **stringpar(*var name*)**

Type: real

Restrictions: Read Only

Results in the value of the string constant *name* from the Dosimis-3 default values.

- **NULL**

Type: various

Restrictions: Read Only

Several variables (modulevar, junctionvar, act_object etc.) can be reinitialized by assignment of this value.

18.4.3 Random Function

- **dice(*int min, int max*)**

Type: int

Results in an integer random value uniformly distributed with lower limit *min* and upper limit *max*. The limits belong to the value range of the return value.

- **erlang_dist (*real average, int k*)**

Type: real

Results in a random value erlang distributed with mean value *average* and k-parameter *k*.

- **exponentially_dist (*real value*)**

Type: real

Results in a random value exponentially distributed with mean value *value*.

- **normally_dist (*real average, real dev*)**

Type: real

Results in a random value normally distributed with mean value *average* and deviation *dev*.

- **triangular_dist (*real min, real max, real top*)**

Type: real

Results in a random value triangular distributed with lower limit *min*, upper limit *max* and top value *top*.

- **uniformly_dist (*real min, real max*)**

Type: real

Results in a random value uniformly distributed with lower limit *min* and upper limit *max*.



18.4.4 Mathematical Function

For the calculation of more complex procedures a set from mathematical functions is provided. All trigonometric functions (sin, asin...) work with radian measure.

- **abs(*real value*)**

Type: *real*

Calculates the absolute value of *value*.

- **acos(*real value*)**

Type: *real*

Calculates the absolute value of *value*.

- **asin(*real value*)**

Type: *real*

Calculates the arcus sinus of *value*.

- **atan(*real value*)**

Type: *real*

Calculates the arcus tangent value of *value*.

- **ceil(*real value*)**

Type: *real*

Calculates the integral value greater or equal to *value*.

- **cos(*real value*)**

Type: *real*

Calculates the absolute value of *value*.

- **floor(*real value*)**

Type: *real*

Calculates the integral value less or equal to *value*.

- **ln(*real value*)**

Type: *real*

Calculates the natural logarithm of *value*.

- **log(*real value*)**

Type: *real*

Calculates the logarithm to the basis of 10 of *value*.

- **pi()**

Type: *real*

Calculates the value of Pi (3.14159...).

- **pow(*real basis, real exponent*)**

Type: *real*

Raises *basis* to the power of *exponent*.



- **round(*real value*)**

Type: *real*

Rounds the value of *value*.

- **sgn(*real value*)**

Type: *real*

Calculates the sign of *value*. The result is -1, if *value* < 0. It is 0, if *value* is 0. It is +1, if *value* > 0.

- **sin(*real value*)**

Type: *real*

Calculates the sinus of *value*.

- **sqrt(*real value*)**

Type: *real*

Calculates the square root of *value*.

- **tan(*real value*)**

Type: *real*

Calculates the tangent of *value*.

18.4.5 Decision Tables

subtable and **global_dt** are selectors. Since however only the keyword **execute** can be applied, these are summarized here accordingly. If in the action a return value is specified (*return* - action), the call supplies this value. This can be used then with an assignment.

- **subtable(*var name*).execute**

Type: *real*

Execute the sub table *name*.

- **global_dt(*var name*).execute**

Type: *real*

Execute the global decision table *name*.

- **dt.execute(*para1*, [*para2*, ...])**

Type: *real*

Executes a decision table and passes the parameter to the decision table.

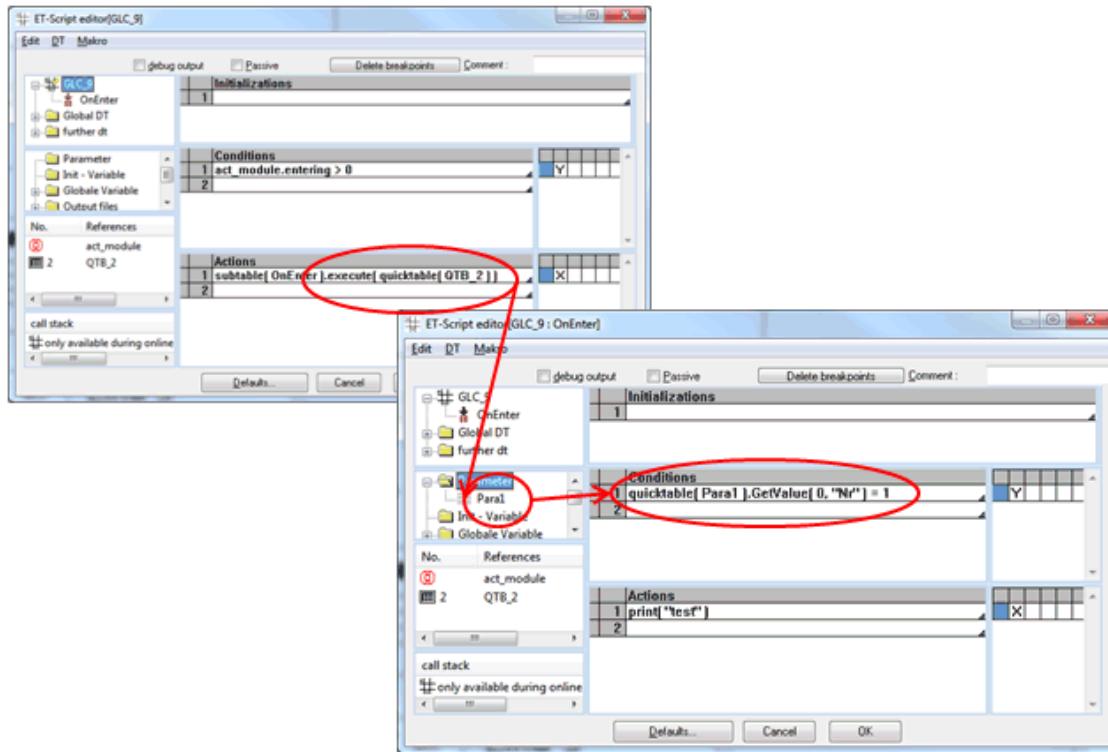


Figure 18.1: Call of a sub table with parameter

The assignment of the call parameters in the order from left to right on the declared parameters (from top to bottom). That is, the first call parameter is mapped to the first parameter in the decision table, the second to the second parameter, etc.

Since the parameters do not have a type, they must be qualified in the use (converted in type). Any use without qualification converts a parameter to type *Int* or *Real*.

Example:

`Quicktable(para1)` changes the first parameter to the type *Quicktable*.
`Object(para3)` changes the third parameter to the type *Object*.

If another type than expected was passed in the decision table, the evaluation raises an error message. The number of parameters is checked in the consistency check.

18.4.6 Text Output

All text output has a unified output format. By default, the output real valued numbers uses the decimal separator from the system settings. This can be changed in the properties of the simulator.

- **print(par param, ...)**

Type: *Operation*

Writes text into the simulation log file. *param* can be an arbitrarily, representable value, thus all values of the types *int* and *real*, as well as text constants. Texts are set in quotes. As many as desired parameters, by commas separated, can be set consecutively.

The output does not take place during the prerun Furthermore the output is suppressed if the switch DT trace in the simulation parameters is deactivated.



Example: **print(„Time :“, sim_time, „ - actual occupancy: “, act_module.actocc)**

- **anitext(var atx).print(par param, ...)** new spelling
- **aniprint(var atx, par param, ...)** old spelling

Type: Operation

Writes a text into an animation text. Animation texts can be placed in the Dosimis-3 model with a name assigned. In order to display texts in the animation text field, one uses this operation. *atx* is the name of the animation text, *param* contains output data (s. **print**).

The output does not take place during the prerun Furthermore the output is suppressed if the switch DT trace in the simulation parameters is deactivated.

- **datei(var dat).print(par param, ...)** new spelling
- **datprint(var dat, par param, ...)** old spelling

Type: Operation

Writes a text into a DT output file. Output files are defined in the variable view of the decision table editor. *dat* is the name of an output file defined before, *param* corresponds to the output text (s. **print**).

The decimal and field separators for the output can be adapted by the configuration of the decision tables.

- **errprint(var dat, par param, ...)**

Type: Operation

Writes a text into the error file (*modelname.err*). Output files are defined in the Dosimis-3 menu *model/global DT data*. *dat* refers the name of an output file defined before, *param* corresponds to the output text (s. **print**). The output can be redirected additionally in the trace file by the configuration of the simulator, so that an this information can be seen in the log window during animation.

18.4.7 Transport System (TS)

All following keywords correspond to automated transport systems.

- **agv_order (int primdest)**

Type: Operation

Create an order with this *primary destination*.

- **agv_order2(int primdest, int secdest)**

Type: Operation

Create an order with primary destination *prim_dest* and secondary destination *secdest*.

- **order_generationtime**

Type: real

Restrictions: Only Transport-DTs

Point of time of the creation of the actual order.

- **order_primdestination**

Type: real

Restrictions: Only Transport-DTs

The primary destination of the actual order.



- **order_prio_primedest**

Type: int

Restrictions: Only Transport -DTs

The priority of the primary destination of the actual order.

- **transportation_time**

Type: real

Restrictions: Only Transport-DTs

The calculated transportation time of the actual vehicle. This is defined by the way matrix.

- **transportsystem.schedule**

- **transportsystem(int no).schedule**

- **transportsystem(string name).schedule**

Type: Operation

The order scheduling of the appropriate transport control is initiated. Neither if name nor number are indicated, the scheduling of the global transport control is activated.

18.4.8 Miscellaneous

- **act_entrance**

Type: int

Restrictions: Only distribution -DT

Results in the entrance, by which the object enters or wants to enter the module.

- **act_exit**

Type: int

Restrictions: Shuttle -DT

Results in the exit, by which the object leaves or wants to leave the module.

- **callfor_wk(int min, int max, int qual)**

Type: Operation

Restrictions: Only working time-DT

Call for at least *min* and maximum *max* worker of the qualification *qual*.

- **call(string function, par param, ...)**

Type: int / Operation

The C-function *function* will be executed. The parameters *param* are transferred. Here is referred to the documentation of the [programming interface](#), where this operation is described. **call** is valid as an operation in an action, and as function - e.g. in comparisons or assignments. The return value is the return of the called C-function.

- **CmdGetParameter(int feld, par param1, par param2, ...)**

Type: int / real / string

Restriction: Not available during simulation.

This function works like the *get instruction* of the excel data transfer. Here the first parameter determines, which field from the result line is to be displayed. This index is 1-based, corresponds thus to the column number in the excel sheet. Further information according the excel interface you find [here](#). The usage is limited on the display of module parameters and simulation results in combination with animation texts.



Example:

1) CmdGetParameter(5, "Module", "ACC_7", "Speed")

returns the fifth field from the result line of the evaluation instruction, thus the value of the speed of component ACC_7.

The result line would be:

"Get", "Module", "ACC_7", "Speed", "0.3"

2) CmdGetParameter(6, "Throughput", "ACC_7", 0, 4)

returns the sixths field from the result line of the evaluation instruction, thus the value of the throughput of component ACC_7 on the last statistics.

The result line would be:

"Get", "Throughput", "ACC_7", "0", "4", "123"

- **for(Assignment; Condition; Counting -Action) main action**
- **for Assignment to Limit step step do main action**

Type: Operation

By this control structure it is possible to execute an operation several times. The Dosimis-3 syntax is near to the C-syntax. First uniquely the *assignment* is executed. Subsequently, the *condition* is checked. If this is true, then only the *main action* and afterwards the *counting action* are executed. Now again the condition is checked. The loop ends only if the *condition* is no longer fulfilled.

Example 1: **for(a:=1; A < 5; a:=a+1) print (a)**

Prints the numbers from 1 to 4 into the simulation log. First *a* obtains the value 1. This is smaller than 5, therefore the print command is executed. Subsequently, *a* is increased by **(a:=a+1)** and obtains so the value 2. This meets still the condition **a<5**. Thus **print(a)** outputs now 2 and *a* is increased, etc.. Only if the value 4 is output and *a* attains afterwards the value 5, the condition does not apply. The loop is terminated. It must be guaranteed that the loop is finite.

Example 2: **for(a:=junctionvar(xyz).blockings; a>0; a:=a-1) junctionvar(xyz).deblock**

releases the junction the junction variable xyz refers to completely.

Example 3: **for a:=1 to 5 step 2 do print(a)**

Print numbers 1, 3 and 5.

- **foreach(Variable in Set) Action**

Type: Operation

By this control structure it is possible to execute an operation several times. This method will be called for each element in the set. This method has two variants. The first one allows to iterate through each module in the decision table. Variable must be of type *module variable* and *set* is of kind decision table.

Example 1: **foreach(modulevar (mod1) in subtable(GSZ_1)) print(modulevar (mod1). no)**



The second variant serves to iterate for each object in a module. Variable is to set to act_object and the set must be a module. The action will be called for each object in the module. By the value of act_object each object can be analyzed.

Example 2: **foreach(act_object in act_module) print(act_object.type)**

- **return (real value)**

Type: *Operation*

Set the return value of a decision table. This is necessary in a distribution decision table with e self defined distribution strategy, where the exit must be determined. The exit 3 will be selected by **return(3)**. The processing of the actions will continue. The return operation is similar to the old style, an assignment to the selfdef variable.

- **round(real value)**

Type: *real*

Rounds the value *value*.

- **signal(var name).display(int field, int red, int green, int blue)** new spelling

- **signal(var name, int field, int red, int green, int blue)** old spelling

Type: *Operation*

If *name* is the name of a signal display, the color of the field *field* of this signal display is set. The value of field must be in the range of 0 to 4. 0 means, that all the 4 fields will be set to the color simultaneously.

If *name* is the name of a progress indicator, the progress bar of this progress indicator is set. Field may take only values between 0 and 100. The bar is filled out according to this percentage in the indicated color.

The color is by the RGB - values defines. The three fields red, blue and green define the color portion of the appropriate basic color, which may have values from 0 to 255. The indicated color results then as mixture of these chromas.

The output does not take place during the prerun. Furthermore the output is suppressed if the switch ET trace in the simulation parameters is deactivated.

- **sim_time**

Type: *real*

Results in the current simulation time.

- **abort(par param, ...)**

Type: *Operation*

Aborts the simulation run. *Param* will be written into the err-file.

- **Condition ? valke1 : value2**

Type: *int/real*

Results to *wert1*, if *condition* applies, otherwise *value2*.

$a := (\text{act_module.entering} = 1) ? 4 : 3$

The variable becomes to 4, if an object has entered the module from entrance 1. Otherwise the value will be 3.



18.5 Module attribute

18.5.1 Values (Read Only)

Values are all data representing the module. Also the actual simulation values belong to these values.

- **actocc**

Type: int

The current allocation of the module, thus the number of objects, which are in the module.

- **cap**

Type: int

The capacity of the module.

- **dt_failsem**

Type: int

Like **failsem**, however the number of disturbances is supplied, which were caused by decision tables.

- **entr_no**

Type: int

The count of entrances of the module.

- **exit_no**

Type: int

The count of exits of the module.

- **failsem**

Type: int

Results in the number of (parametrized) failures, which affect the module at this time.

- **name**

Type: String

The name of the module.

- **no**

Type: int

The number of the module.

- **throughput**

Type: int

Show the throughput of the module, thus the number of objects, which left the module in the past simulation process. The value is reset at the end of the pre-run.

- **maxocc**

Type: int

The max. occupancy of the module, thus the largest number of objects, which were at the same time in the module during the last simulation period. The value is reset at the end of the pre run.



- **maintenancesem**

Type: int

Results in the number of maintenances, which affect the module at this time.

- **pausesem**

Type: int

Results in the number of pauses, which affect the module at this time.

- **sum_failsem**

Type: int

This is the total of all interruptions, which affect the module. It counts together from **failsem**, **dt_failsem**, **maintenancesem** and **pausesem**.

Further values can be found in the section event and state.

18.5.2 Values (Write Only)

Values are all data representing the module. Also the actual simulation values belong to these values.

- **basicposition**

Type: Operation

Restriction: Only shuttle and turntable

Set the basic position of the module to the given value. This marks the entrance, that will be reached, when no further transport is to be executed. When in the moment, the assignment was made, no transport takes place, the change of the position will start automatically.

18.5.3 Values (Read/Write)

Values are all data representing the module. Also the actual simulation values belong to these values.

- **integer.A ... integer.J**

Type: int

These values can be assigned and queried freely.

- **real.A ... real.J**

Type: real

These values can be assigned and queried freely.

- **intatt.name**

Type: int

Gives the value of the integer - attribute with the identifier *name* of the module. When the attribute is not yet defined, an error occurs in case of a read access. In case of write access the attribute will be created.

- **floatatt.name**

Type: real

Gives the value of the float - attribute with the identifier *name* of the module. When the attribute is not yet defined, an error occurs in case of a read access. In case of write access the attribute will be created.



18.5.4 Event and State

Under statuses all attributes are summarized, which determine the work routine of a module. These methods work only, if the querying module has activated the processing of the decision table.

- **exit_pos(int no)**

Type: real

Restriction: Shuttle

Results in the position (in meters) of the exit number *no*.

- **movement**

Type: int

Restrictions: Shuttle, Turntable or Crossing

Supplies the movement status of the shuttle. -1:undefined, 0: unknown (it does not support this function), 1: Beginning of movement, 2: In motion, 3: Movement end, 4: not in motion.

- **entrance_pos(int no)**

Type: real

Restrictions: Shuttle

Results in the position (in meters) of the entrance number *no*.

- **free_segments(junction jun, int direction)**

Type: int

Restrictions: conveying circuit

The number of unoccupied segments, counted from the given junction. If direction is equal 1 it is counted according the transfer direction. If direction is equal 0 it is counted in the opposite direction.

- **direction**

Type: int

Restriction: shuttle

Returns -1, if the shuttle runs to a lower position, results 1, if the shuttle is running to a higher position and 0, if it is not moving.

- **destination**

Type: int

Restrictions: shuttle

Returns the destination of the shuttle. Entrances are numbered continuously starting at 1 followed by the exits The number for the first exit of a shuttle with 5 entrances is 6.

- **entering**

Type: int

Results in the number of the entrances, by which an object has entered the module. The value is 0, if the decision table was not released by an entrance of an object.

- **exiting**

Type: int

Results in the number of the exit, by which an object has left the module. The value is 0, if the decision table was not released by an exit of an object.



- **failed**

Type: int

Return 1 with the start of the failure, 2 during the failure, 3 with the end of the failure and und 4 otherwise. With overlaying failures the return value of 1 takes place only with the first failure. Analogous the return value of 3 applies only with the end of the last failure.

- **sensor**

Type: int

Restrictions: Only with accumulating conveyor of type 2

Results in the number of the affecting sensor.

- **sensor**

Type: int

Restrictions: Sensors exists only with accumulating conveyors 2.

The number of the activating sensor of a module. The functions is available only for compatibility reasons and should not be used. Use instead: act_sensor.no

- **sensor(int no)**

Type: Sensor

Restrictions: Sensors exists only with accumulating conveyors 2.

Results in the sensor with number *no*.

- **working**

Type: int

Restrictions: Only valid for working modules

1 supplies with beginning of work, 2 during the handling, 3 with end of work and 4 if the module waits. If nothing of that applies, then 0 is return.

18.5.5 Objects

- **f_obj**

Type: Object

This is the first object that is in the module. The first object is the next object to leave the module.

- **l_obj**

Type: Object

This is the last object, which is in the module.



18.5.6 Strategies

The right of way and distribution strategies, which do not need further parameters, can be called directly from the decision tables. If a computation has been made and the object has not been taken over/exiting yet, these strategies return the input/output calculated last. If the calculation is unsuccessful, the return value is 0. If the module does not possess the appropriate strategy, 1 is returned.

- **distribution.exit**

Type: int

Returns the exit, which was determined with the last distribution strategy. This is 0, if this were not evaluated yet and 1, if the component does not possess a distribution strategy.

- **distribution.shortestpath**

Type: int

Restriction: only with shuttle.

Returns the exit in accordance with the strategy *shortest path*.

- **distribution.maxfreecap**

Type: int

Returns the exit in accordance with the strategy *maximum free capacity*.

- **distribution.minocc**

Type: int

Returns the exit in accordance with the strategy *minimum occupation*.

- **right_of_way.entrance**

Type: int

Returns the entrance, which was determined with the last right of way strategy. This is 0, if this were not evaluated yet and 1, if the component does not possess a distribution strategy.

- **right_of_way fifo**

Type: int

Returns the entrance in accordance with the right of way strategy *First In First Out*.

- **right_of_way.shortestpath**

Type: int

Restriction: only with shuttle, turn table or crossing.

Returns the entrance in accordance with the right of way strategy *shortest path/ shortest turning time*.

- **right_of_way.maxabsocc**

Type: int

Returns the entrance in accordance with the right of way strategy *maximum absolute occupation*.

- **right_of_way.maxrelocc**

Type: int

Returns the entrance in accordance with the right of way strategy *maximum relative occupation*.



18.5.7 Transport Systems (TS)

All these attributes refer to automated transport systems.

- **destaddress**

Type: int

The destination code of the module.

- **f_load**

Type: Object

Restrictions: The first object in the module has to be a vehicle.

Results in the object, which is transported by the first vehicle in the module.

- **l_load**

Type: Object

Restrictions: The last object in the module has to be a vehicle.

Results in the object, which is transported by the last vehicle in the module.

18.5.8 Storage

The following attributes are applicable only in storages.

The methods for retrieval work even the object cannot be found in the storage. Then an anonymous order is generated, which is activated when an object with the respective criteria is arrives in the storage.

After the call of the retrieval methods further attributes can be assigned to the object which should be released. This is done via assignment to the basic element *act_object*, which refers after the call to the object which can be released. This is also possible, if the object is not yet at all in the store. Then these data is stored with the waiting queue.

- **warehouse_area**

Type: int

Restrictions: Only storage modules

The number of the storing area with the smallest occupancy.

- **no_obj_in_warehouse**

Type: int

Restrictions: Only storage modules

The count of objects in the storage

- **no_obj_in_warehouse(int type, int area)**

Type: int

Restrictions: Only storage module

Returns the number of objects of the type *type*, which are in the area *area* of the storage.

- **retrieval_att (var name, int value [, int dest])**

Type: Operation

Restrictions: Only storage modules

An object, which contains the value *value* of the integer attribute *name*, is delivered from the store. If *dest* is set, the value of *dest* is assigned to the destination parameter of the object.



- **retrieval_obj (int type [, int dest])**

Type: Operation

Restrictions: Only storage modules

An object of type *type* is delivered from the store. If *dest* is set, the value of *dest* is assigned to the destination parameter of the object.

- **retrieval_obj_no (int nr [, int type] [, int dest])**

Type: Operation

Restrictions: Only storage modules

The object with the unique number *nr* is delivered. If the type of object is known, this function works more efficiently. If the type of object is unknown, then second parameter should be set to 0. The value of *dest* is assigned to the destination parameter of the object.

18.5.9 Model

The following module attributes refer in the structure of the model

- **junction_at_entrance(int no)**

Type: Junction

Restrictions: Not valid with sources

Results and selects the junction, which is at the entrance with the number *no* of the module selected before.

- **junction_at_exit(int no)**

Type: Junction

Restrictions: Not valid with sinks

Results and selects the junction, which is at the exit with the number *no* of the module selected before.

- **module_at_entrance (int no)**

Type: Module

Restrictions: Not valid with sources

Results and selects the module, which is at the entrance with the number *no* of the module selected before.

- **module_at_exit (int no)**

Type: Module

Restrictions: Not valid with sinks

Results and selects the module, which is at the exit with the number *no* of the module selected before.

18.5.10 Operations

All further operations.

- **clear([int part])**

Type: Operation

Clears the module, the status value of **dt_failsem** is decreased by 1. If **dt_failsem** is greater than zero, the module remains disturbed. The parameter **part** is optional. Only the modules of type shuttle and storage support this parameter:



In case of a shuttle it denotes the number of the table. In case of a storage it is the number of the area (single lane). The sub-elements are addressed by numbers from 1 to n.

- **clear_time(real time, [int part])**

Type: Operation

Clears the block for the specified duration (in seconds). For the specified period, the state value reduces **dt_failsem** of the module by 1. If the module is not disturbed at call time, nothing happens. The meaning of the optional parameter **part** corresponds to the description in **clear**.

- **disturb([int part])**

Type: Operation

Disturbs the module. Increases the status value **dt_failsem** by 1. The break will get active as soon as the current process has finished. The meaning of the optional parameter **part** corresponds to the description in **clear**.

- **disturb_time(real time, [int part])**

Type: Operation

Disturbs the module for the referenced time (in seconds). The break will get active as soon as the current process has finished. The meaning of the optional parameter **part** corresponds to the description in **clear**.

- **gen_object (int type)**

Type: Operation

Restrictions: Only valid with sources

An object is produced by this statement of the type *type* in the source selected before.

- **Stop_work (int type)**

Type: Operation

Stops the work in the module. The value of type determines, how to do this. When the value is set to 0, the current work will be interrupted (e.g. work in a work procedure in a working station). When the value is set to 1, the whole work will be interrupted a the object is ready to leave the module

18.6 Object attribute

18.6.1 Values (Read Only)

Values are all data, which represent the object. To this also the current simulation values belong.

- **input_time**

Type: real

The entering time of the object. The point of time, the object started to enter the module.

- **no**

Type: int

The unique number of the object. All Dosimis-3 objects have a number, which is unique within the material flow system.



- **order**
Type: order (only objects with a work plan)
 Current order of the object.
- **output_time**
Type: real
 The time lag of the object. That is the point in time, when an object announces itself to the exit from a module. This point in time can be situated also in the future ($> sim_time$). It is possibly already beforehand well-known, when an object will announce itself.
- **workingstep**
Type: working step (only objects with a work plan)
 Current working step of the object.
- **working plan**
Type: • working plan (only objects with a work plan)
 Current working plan of the objct.

18.6.2 Values (Read/Write)

Values are all data, which represent the object. To this also the current simulation values belong.

- **type**
Type: int
 Results in the type of the object. With the assignment operator := the type can be changed.
 Example: **act_module.e_obj.typ :=15**
- **no**
Type: int
 The number of the object.
- **destparam**
Type: int
 This is the destination of the object for the distribution strategy *distribution by destination parameter*. With this keyword also a destination parameter can be assigned to the object.
- **delivery_destination**
Type: int
 This is the destination code of the secondary destination, which the object should reach. This is meaningful only for loads in a transport system. This value is used with loading instead of the destination code indicated in that loading station.
- **integer.A ... integer.J**
Type: int
 These values can be assigned and queried freely.
- **real.A ... real.J**
Type: real
 These values can be assigned and queried freely.



- **intatt.name**

Type: int

Gives the value of the integer - attribute with the identifier *name* of the object. When the attribute is not yet defined, an error occurs in case of a read access. In case of write access the attribute will be created.

- **floatatt.name**

Type: real

Gives the value of the float - attribute with the identifier *name* of the object. When the attribute is not yet defined, an error occurs in case of a read access. In case of write access the attribute will be created.

18.6.3 Transport Vehicles(TV)

The following keywords are only valid with transport vehicles.

- **agv_no**

Type: int

Restrictions: Only valid with vehicle-objects

The vehicle-id of the vehicle.

- **batt_load**

Type: int

Restrictions: Only valid with vehicle -objects

The battery loading status of the vehicle.

- **set_order(int primdest)**

Type: Operation

Restrictions: Only valid with vehicle -objects

Assigns an order with the indicated *primary destination* to a free vehicle. Possibly an order job can be assigned to the vehicle, which was already assigned to another vehicle.

With call of this routine the scheduling tries to force a doubles match. In detail one the proceeding is as follows:

(1) an order is looked for with *primdest* as a primary destination. First one tries to find an

order which is still not assigned to a vehicle. If there is no such order, an order is looked for, which is assigned already to another vehicle. This is then taken away from the second vehicle.

(2) that momentarily assigned order of the vehicle is taken away likewise. If it an order concerning a transportation (no travel into the waiting position, direct arrangement...), these is released for scheduling. Instead the first order is assigned.

(3) the second vehicle is scheduled again.

In the case that there is no order with the indicated primary destination, it does not happen anything. In the case that the order was not assigned to any vehicle, only step 2 is executed.



- **set_free_order(int primdest)**

Type: Operation

Restrictions: Only valid with vehicle -objects

Assigns an order with the indicated *primary destination* to a free vehicle. This function works essentially like the function *set_order*, but one searches only within the orders which are not assigned yet.

- **prim_destination**

Type: int

Restrictions: Only valid with vehicle -objects

The primary destination of the vehicle.

- **sec_destination**

Type: int

Restrictions: Only valid with vehicle -objects

The secondary destination of the vehicle.

- **new_destination(int dest)**

Type: Operation

Restrictions: Only valid with vehicle -objects

Assigns the destination of a vehicle to the value of *dest* and starts it. There must be a module in the model, which matches the given destination.

- **batt_loading**

Type: Operation

Restrictions: Only valid with vehicle -objects

Instructs the selected vehicle to drive to a battery loading station.

18.7 Junction attribute

18.7.1 Value (Read Only)

The values represent the data of the junction. All these values cannot be changed.

- **no**

Type: int

The number of the junction.

- **blockings**

Type: int

The count of blockings of that junction.

- **occupied**

Type: bool

The junction is occupied.

- **waiting**

Type: bool

An object is waiting at the junction.



- **ready_to_leave**

Type: *bool*

An object is ready to leave at the junction (announced).

18.7.2 Operation

The transfer of objects from one module to his successor can be prevented by disabling the junction that connects the two modules. It should be noted that a junction can be blocked several times. Then the node must also be unblocked several times. Alternatively, this can be achieved by the function *open*.

- **block**

Type: *Operation*

The junction will be blocked. The count of blockings of that junction is increased by 1.

- **block_time(*real duration*)**

Type: *Operation*

The junction will be blocked for a given **duration**.

- **deblock([*real time*])**

Type: *Operation*

The junction will be deblocked. The count of blockings of that junction is decreased by 1. If the optional parameter **time** is specified, the output time of a waiting object is corrected according this value.

- **deblock_time(*real duration*, [*real time*])**

Type: *Operation*

The junction will be deblocked for a given **duration**. The count of blockings of that junction is decreased by 1. If the optional parameter **time** is specified, the output time of a waiting object is corrected according this value.

- **open([*real time*])**

Type: *Operation*

The junction will be totally deblocked. The count of blockings of that junction is set to 0. If the optional parameter **time** is specified, the output time of a waiting object is corrected according this value.

- **open_time (*real duration* [*real time*])**

Type: *Operation*

The junction will be totally deblocked for a given **duration**. If the optional parameter **time** is specified, the output time of a waiting object is corrected according this value.



18.8 Quicktable

This section describes the functions, which are available with quicktables. Calls to the quicktable frequently possess 2 versions. So in all cases, where the column is referenced, this can be done by a number as well as by a name. For the second type the quicktable must contain a headline.

18.8.1 Values

- **FirstEmptyLine()**
- **FirstEmptyLine(*int col*)**
- **FirstEmptyLine(*string col*)**

Type: int

Serve the first line with empty entry in columns *col*. When no such entry exist, the return value is -1. The first variant examines only whether for line an entry does not exist and works faster. For marking it is not sufficient however to put an empty text ("") to the first field. The line must be deleted (*DeleteRows*).

- **GetColumn()**

Type: int

Returns the total number of the columns.

- **GetColumn(*string col*)**

Type: int

Returns the number of the column with the title *col*.

- **GetString(*int line, int col*)**
- **GetString(*int line, string col*)**

Type: String

Serve the text of the table from line *line* and column *col*. The return value can only be used with text output. When the indexed filed does not exist, an empty string is returned.

- **GetValue(*int line, int col*)**
- **GetValue(*int line, string col*)**

Type: double

Results in the value of the table from line *line* and column *col*. When the indexed field does not exist, the return value is 0.

- **LastLine()**

Type: int

Return the index of the last line in the table, which possesses still another entry. (see also FirstEmptyLine).

- **no**

Type: int

The number of the quicktable.

- **ReadLine (*int count = 1*)**

Type: int

Reads *count* lines of the table. The return value is the number of the last read in line. The return value is -1, if no line could be read in and thus the end of the file is reached.



- **SearchTable (int start, int col, int mode, double value)**
- **SearchTable (int start, string col, int mode, double value)**
- **SearchTable (int start, int col, string mode, double value)**
- **SearchTable (int start, string col, string mode, double value)**
- **SearchTable (int start, int col, int mode, string value)**
- **SearchTable (int start, string col, int mode, string value)**
- **SearchTable (int start, int col, string mode, string value)**
- **SearchTable (int start, string col, string mode, string value)**

Type: int

Search in the table up from line *start* in column *col* for the value *value*. *mode* determines, when the search has to stop.. 0 mean equality, 1 inequality, 2 less than, 3 greater, 4 less or equal and 5 greater or equal. The search will be stopped, when the comparison is true for the first time. (mode: 0:==; 1:!=; 2:<; 3:>; 4:<=; 5:>=). Instead of the value the corresponding text (“==“, “!=“, ...) can be used. The return value is the number of the line, found according the comparison. When no such entry exist, the return value is -1.

- **SubTotal (int/string mode, int/string col[, int start [, int end]])**

Type: int

The function performs the calculation defined by the parameter **mode**. The calculation is made according the column **col** from line **start** to line **end**. If end is not specified or negative, end is set to last line. If start is omitted, start is set to the first line.

The following *mode* parameter values are allowed:

- 1 or „avg“: Returns the average
- 2 or “cnt”: Returns the number of fields
- 4 or “min”: Returns the minimum value within the range
- 5 or “max”: Returns the maximum value within the range
- 6 or “sum”: Returns the sum of all values within the range

ATENTION: the values 3, 6, 7, 8 are not used.

18.8.2 Operation

- **CopyLine (int lineTo, quicktable (NameFrom), int linefrom)**

Type: Operation

Copies the line *linefrom* of the table *nameFrom* into the line *lineTo* of the quicktable.

- **DeleteRows (int line, int anz)**

Type: Operation

Delete *anz* lines in the table on from line *line*.

- **InsertRows (int line, int anz)**

Type: Operation

Add *anz* lines after line *line*.

- **MoveLine (int lineTo, quicktable (NameFrom), int linefrom)**

Type: Operation

Moves the line *linefrom* of the table *nameFrom* into the line *lineTo* of the quicktable.



- **SetValue (int line, int col, double value)**
- **SetValue (int line, string col, double value)**

Type: *Operation*

Set the value in the table in line *line* and column *col* to the value of *value*.

- **SetString(int line, int col, string value)**
- **SetString(int line, string col, string value)**

Type: *Operation*

Set the value in the table in line *line* and column *col* to the value of *value*.

- **SortTable (int col, int mode)**
- **SortTable (int col, string mode)**

Type: *Operation*

Sort table according to column *col*. The Value of *mode* determines the search direction: 1 means forward, 0 backward. The sorting is stable. This means that lines, which are identical in accordance with sorting criterion, remain in the same sequence in the received arrangement.

18.9 Timer

18.9.1 Values

- **activetime ()**

Type: *real*

Provides the activation time of a timer. If the timer is not enabled, -1.0 is returned.

- **name ()**

Type: *string*

Provides the name of the timer.

- **Calculate ()**

Type: *real*

Activate the random process of the timer and returns the value in accordance with the parameterized distribution.

18.9.2 Operations

- **activate**
- **activate(*real time*)**

Type: *Operation*

Activates the timer and with this all decision tables, connected with this timer, after the indicated time (in seconds). With indication of a time the activation takes place only after indicated the time (in seconds). Note: For the timer "no distribution" is to be selected as point of activating time!



- **no**

Type: int

The number of the timer supplies. Returns 0, if the timer is not existed (e.g. if act_timer.no was queried but the DT was not activated by a timer).

- **passivate**

Type: Operation

Disables a timer and therefore all decision tables associated with this timer. If the timer was not active before, nothing happens.

18.10 Worker

18.10.1 Values

- **qualification**

Type: int

Get the qualification of the current worker.

18.11 Transport order

18.11.1 Values

- **generationtime**

Type: real

Returns the date on which the order was created.

- **primdestination**

Type: int

Supplies the primary destination of the transport order.

- **sec_destination**

Type: int

Supplies the secondary destination of the transport order.

- **prio_primdestination**

Type: int

Supplies the priority of the module at the primary destination of the transport order.

- **prio_secdestination**

Type: int

Supplies the priority of the module at the secondary destination of the transport order.

- **module_primdestination**

Type: module

Supplies the module at the primary destination of the transport order.



- **module_secdestination**

Type: *module*

Supplies the module at the secondary destination of the transport order.

18.12 Work

18.12.1 Values

- **qualification**

Type: *int*

Get the qualification, needed for the current work

18.13 Working Step

18.13.1 Values

- **processtime**

Type: *real*

Returns the duration of the work of the working step.

- **module (int no)**

Type: *module*

Returns the nth module from the list of alternating module of the working step.

- **name**

Type: *String*

Returns the name of the working step.

- **no**

Type *int*

Returns the number of the working step.

- **setuptime**

Type: *real*

Returns the duration of the setup of the working step.

18.14 Working plan

18.14.1 Values

- **name**

Type: *String*

Returns the name of the working plan.



- **no**

Type: *int*

Returns the number of the working plan.

18.15 Order

18.15.1 Values

- **name**

Type: *String*

Returns the name of the working plan.

- **lotsize**

Type: *int*

Returns the lotsize of the order.

- **lotno**

Type: *int*

Returns not lot number of the actual objects within the order.

- **no**

Type: *int*

Returns the number of the order

18.16 Sensor

18.16.1 Values

- **no**

Type: *int*

Returns the number of the sensor within the module.

- **pos**

Type: *real*

Returns the position (in meters) of the sensor within the module.

18.17 Transport system

18.17.1 Operations

- **disposition**

Type: *Operation*

Activates the scheduling of the transport system. One tries to assign driving orders to the free vehicles.



18.18 File

18.18.1 Operations

- **print(*par param, ...*)**

Type: Operation

Specifies text to an output file. *Param* corresponds to the output text (see [print](#)).

18.19 Animation text

18.19.1 Operations

- **print(*par param, ...*)**

Type: Operation

Specifies text to an output file. *Param* corresponds to the output text (see [print](#)).





19 Error Situations

If during the processing of the decision tables an error is detected, a message occurs. This is to be confirmed normally only with *cancel*. *Retry* leads to the start of the Visual C++ Debugger if you own a Debug Version of Dosimis-3 (only possible in connection with the programming interface).

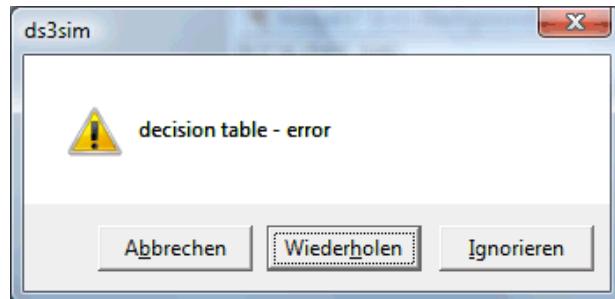


Figure 19.1: Error message of the simulator

Following output takes place into the *err-file*, which one sees by the menu **Simulation/Errorfile**.

Example:

```
Runtimeerror in (decisiontable-) execution of
Exitdecisiontable KKN_7
entrance not defined

modulevar( ele ) := module_at_entrance( 5 , module( 55 ) )
executing action no. 5 with rule 4

called by
Elem no: 7 Exit-DT

List of netattributes:
break1      :    99.00
new_ent_id   :    99.00
min_real_no  :    99.00

List of modulevariables:
ele : undefined
merk : undefined
sst : undefined
```

Figure 19.2: Error file





20 Log Window

The text output of the decision tables are shown in an own window so that all information can be seen compactly. The debug output is redirected into this window and colored according to the information level.

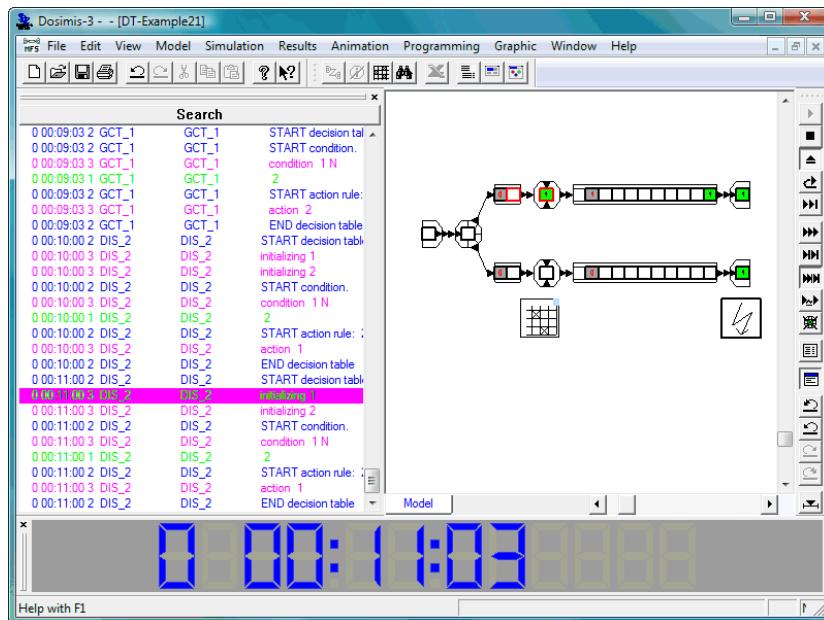


Figure 20.1: Log window

The information of the debug output is the following. In the first column the simulation time stands in the format Day Hour:Minute:Second. Behind that the output of the name of the main- followed by the name of the sub decision table occurs. In the fourth column the actual information takes place. The standard text output is colored in black.

20.1 Searching in the Log window

It is the possibility, to search for text in the log window. By the button **Search** a dialog offers several variants searching for a text.

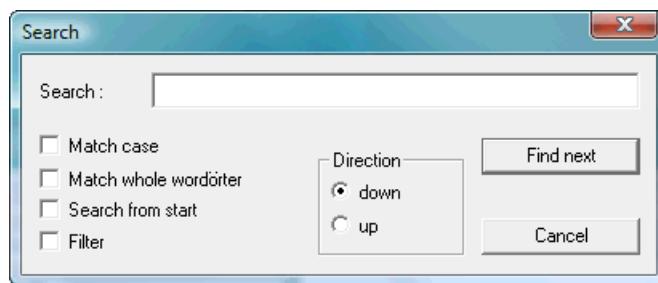


Figure 20.2: Searching in the log window

Beneath the known options there is a switch called **Filter**. When this is activated, only these output in done in the log window, which refers to the searching text.





21 Animation of Decision Tables

For the validation of decision tables the possibility exists to carry out an animation.

For this purpose in the dialog *Global-DT-Data* the mode *Animation* is to be selected in the field *DT-Debug*.

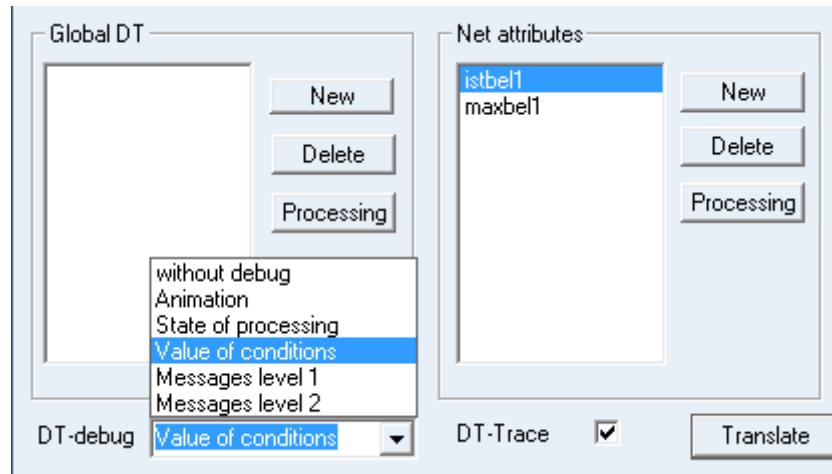


Figure 21.1: End of transaction debug selection

Further on in the decision table, which is to be analyzed, the check box *debug output* is to be activated.

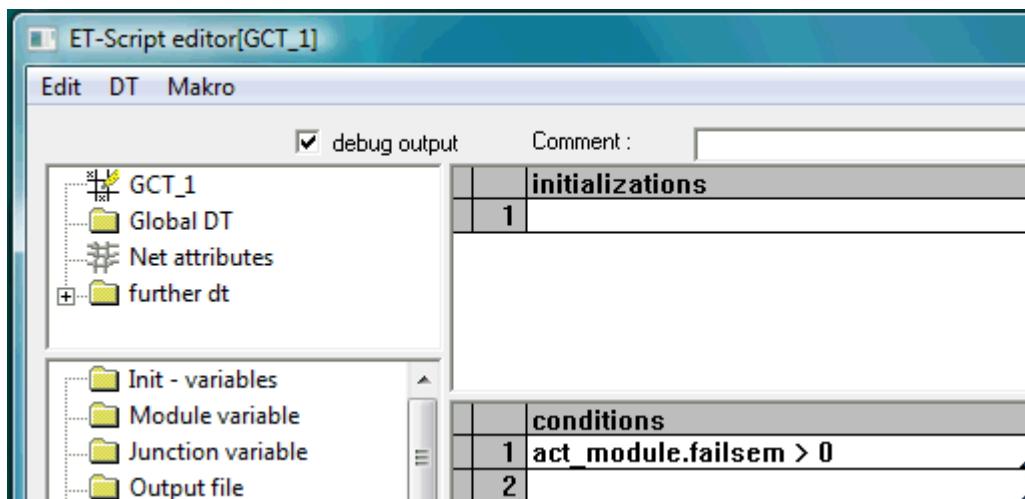


Figure 21.2: Selection with debug output

During the simulation in the trace file additional information is logged, depending upon stage of detail. This provides during the animation the visualization of activations and analyses of the decision table.



This occurs with the *single step animation*, as after analysis of the conditions the decision table dialog is inserted and the relevant rule (column) is marked red. Additionally fulfilled conditions and actions, which are to be executed, marked in light blue.

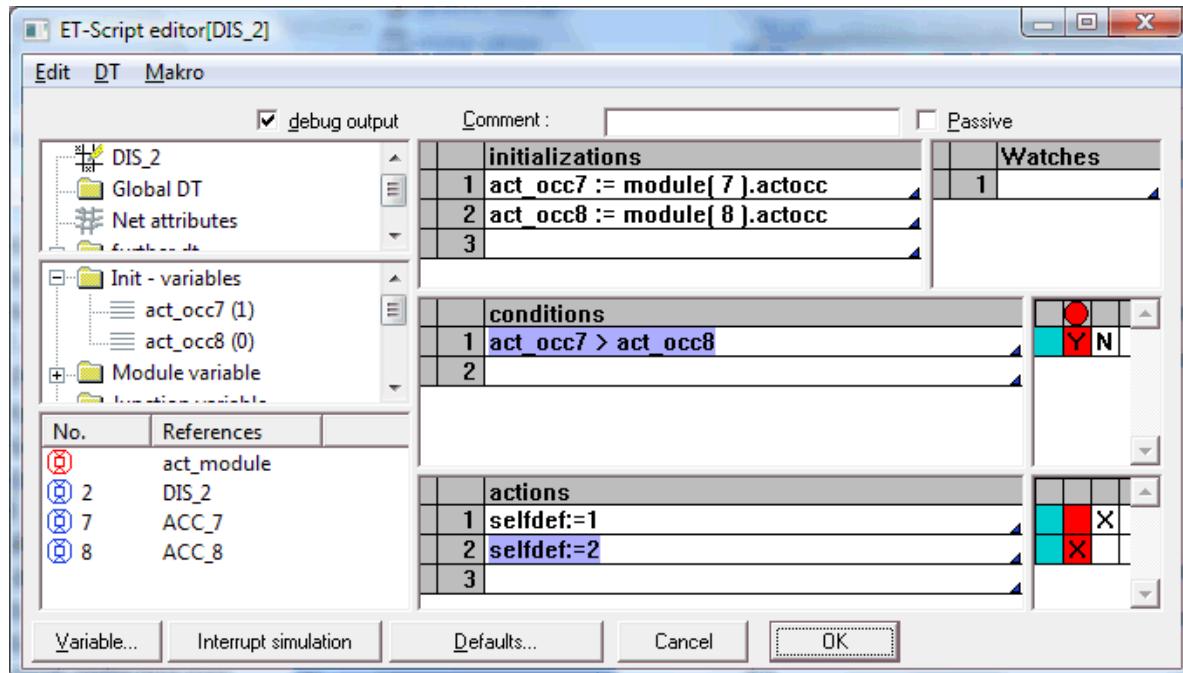


Figure 21.3: Animation of decision tables

With the click on the button **Interrupt animation** the animation will set into the state pause.

All further output is indicated in the log window.

All debug output in the trace file is equipped with the *ED* key and can be searched so easily.

Table top from the trace file during the evaluation of a decision table:

ED 120.7 2 DIS_2	DIS_2	START decisiontable
ED 120.7 2 DIS_2	DIS_2	START condition.
ED 120.7 3 DIS_2	DIS_2	condition 1 =
ED 120.7 1 DIS_2	DIS_2	1
ED 120.7 2 DIS_2	DIS_2	START action rule: 1
ED 120.7 3 DIS_2	DIS_2	action 1
ED 120.7 2 DIS_2	DIS_2	END decisiontable

Next to the key it is to take from every line:

Simulation time in second, debug-level (1-5), name of the main decision table, name of the sub decision table, message.

For the animation of decision tables the names of the main decision table and the respective sub decision tables must be unambiguous. If this is not the case, a fault report occurs in the consistency check if this is not valid for a table in which debug output is activated.



22 Online - Simulation

Next to the decision table animation also the methods for validation are available to the online simulation. Especially for the analysis of global controls the simulation can be interrupted purposefully during the evaluation of a main decision table. For this purpose from the context menu of the Global Control that rule is to be selected, whose evaluation should suspend the online simulation. Alternatively this can be done by a defining breakpoint in the decision table dialog.

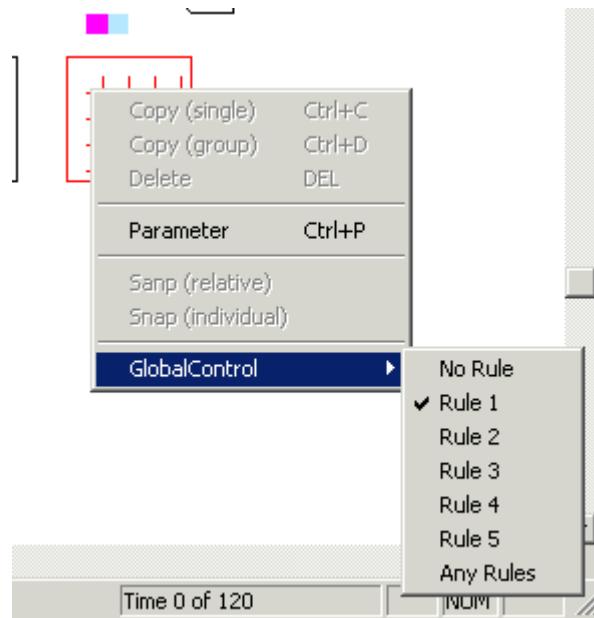


Figure 22.1: Definition of the “Stop” - Rule in the online - simulation

Analogue to the DT - animation the debug output is to be activated. Supplementary at least the decision table animation (Global DT-data - dialog) must be selected. The activation is characterized by a violet square. After attaining the state (in this case, that the rule 1 comes for the execution), the simulation is suspended and the parameter mask of the decision table is opened.

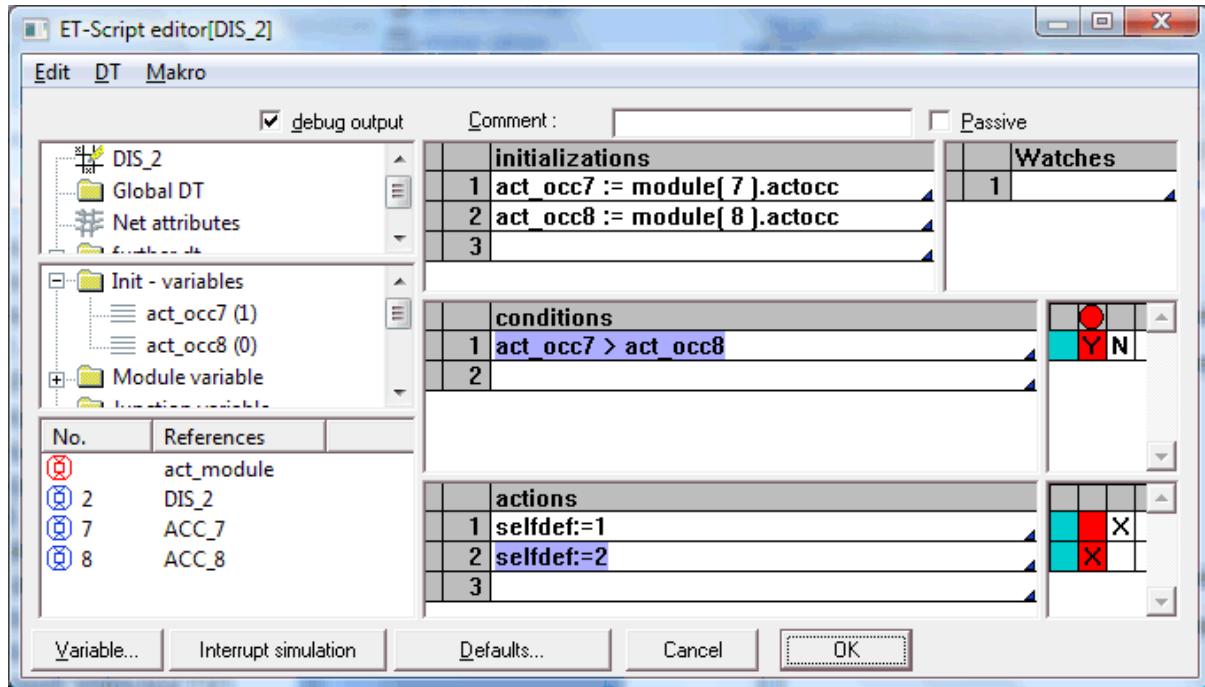


Figure 22.2: selection during DT-debug

During online simulation the values of the init variables are shown in the variable window behind the declaration. Thus central values, which determine the conditions or actions of the decision table, can be seen quite simple.

If the dialog is left via **OK**, the initializations are calculated and the conditions are evaluated again, so that changes become valid and can be examined immediately.

ATTENTION:

The recalculation of the init variables can lead to unintended side effects. Therefore the dialog should be left with **Cancel**, if no changes have been made.

With the click on the button **Delete breakpoints** all breakpoints of the decision table can be removed at one time. When pressing the Ctrl-key simultaneously, the debug output will be removed too.

With the click on the button **Interrupt simulation** the simulation will set into the state pause.

Next to the standard processing of the decision table the most important **Variables** of the decision table can be analyzed in addition.



Variable		Value	New ...
:Module 2		2	DIS_2
Number of entrances		1	1
Number of exits		2	2
Entrance Junction		{...}	
Exit Junction		{...}	
Capacity		1	1
Reservation		0	0
next exit		0	0
Breakpoint		0	0
Occupancy		0	0
Failsemaphor		0	0
Maintenancesemaphor		0	0
Breaksemaphor		0	0
DT-failsemaphor		0	0
Throughput		121	121
Mod-attributes		NULL	
Net attributes		{...}	
Module variable		{...}	
??act_module		2	DIS_2
Number of entrances		1	1
Number of exits		2	2
Entrance Junction		{...}	
Exit Junction		{...}	
Capacity		1	1
Reservation		0	0
next exit		0	0
Delete immediately			
Delete breakpoint			
OK			

Figure 22.3: End of transaction debug selection

In addition to the activating module these are also the net attributes and all components of the decision table. All values in a red color can be changed. This happens by a double-click on the parameter and the change of the value in the dialog following then.

22.1 Profiling

In online - simulation one can be examined the performance of the decision table more exactly. If the Profiling is activated in the properties of the decision tables, decision table calls are seized quantitatively and temporally. After an online simulation run these information can be represented in the layout. Additionally individual analyses are logged in the log file. Similarly operations of the quicktables are logged.

In the percentage occupation the relation of the total time of all calls in relation to the total time of the calls for the indicated quicktable/decision table is represented.

In the throughput diagram the number of calls of the Quicktable/decision table is represented.



In the actual occupation the absolute time is represented, which was needed for the evaluation of the Quicktable/decision table (in seconds).

With operations with large Quicktables (more than 5000 lines) additionally a warning takes place if one of the functions *SearchTable*, *LastLine* or *FirstEmptyLine* analyzes more than 20% of the lines.



23 Examples

23.1 Example of a Target and GC Decision Tables

23.1.1 Description of the Example

The necessity, to take influence to system performance by the possibilities described above, becomes conscious the user, if he must solve problems, which cannot be realized any longer with the usual strategies or parameterization. It may happen that the strategies for out- and input fail because only the states of the very next modules are checked. These local strategies can be expanded into global strategies utilizing decision tables by using any modules for the respective strategy.

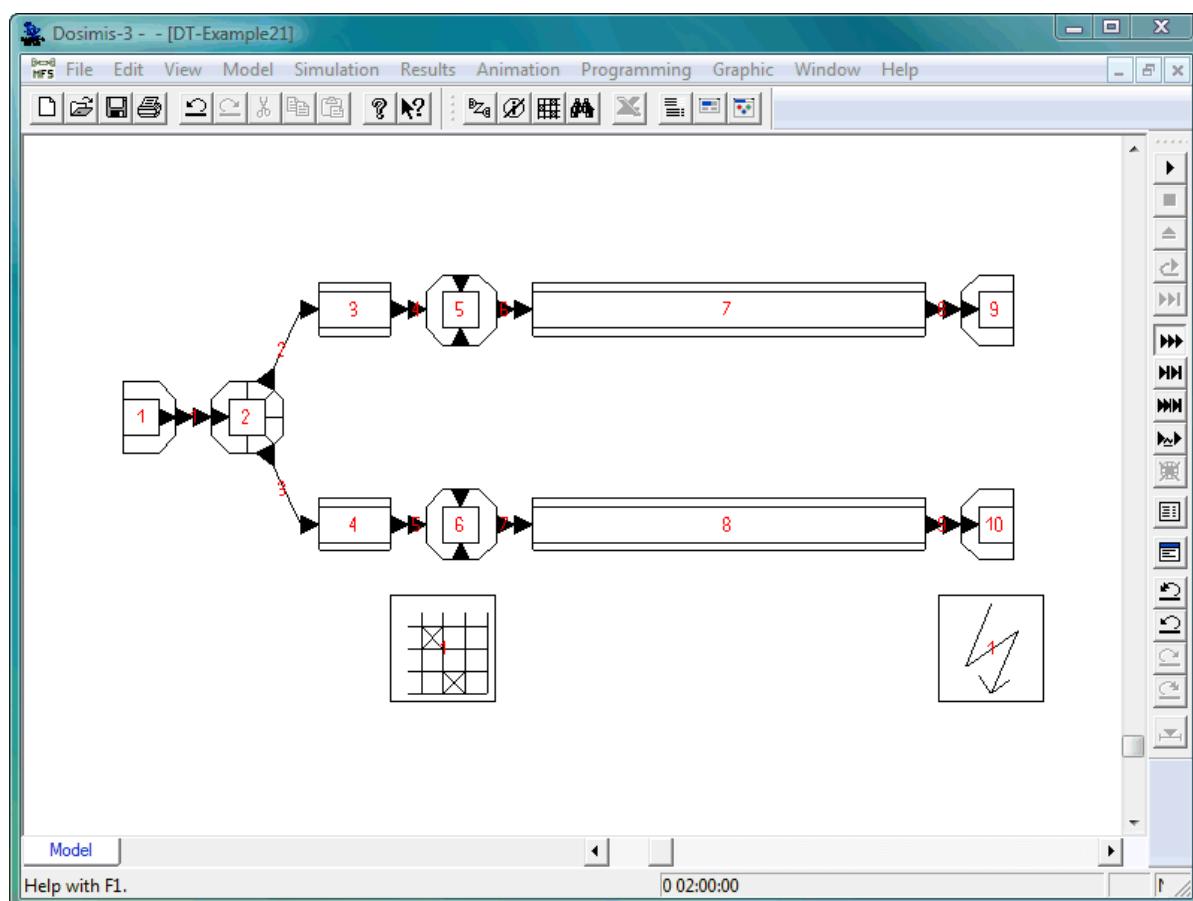


Figure 23.1: Example Layout

The source generates two different types of objects that the distributor DIS_2 sends off to two workstations. From there they pass through a FIFO storage on their way to the sinks.

If a disturbance occurs in workstation 6, then sometimes the distributor DIS_2 can be blocked by the developing tailback in accumulate conveyor 4. This effect is to be prevented by a suitable GCT. In order to keep the text readable, the usual Dosimis-3 module names are used. Workstation 6 is called thus WST_6, distributor 2 accordingly DIS_2.



Basically problems described here can be solved by different approach. We show the solution of the problem by the use of a GCT-DT.

The methodology is the following: The distributor DIS_2 is parameterized in such a way that it uses the two outputs alternating. Additionally the switch *lock* must be activated for this strategy. This ensures for the fact that the distributor reacts dynamically to blocked output junctions. Thus if the junction between DIS_2 and ACC_4 is blocked, the distributor uses only the output to ACC_3.

We use a decision table, in order to lock this junction whenever WST_6 is disturbed. The junction is again released at the disturbances end.

First a GCT module is to be placed into the model. In the mode *linking active* you select the activating modules for this GCT. We remember that only such modules should be activating modules, whose change in status causes an action of the GCT. Connecting activating modules to a GCT functions behaves in the same principle as a connection of two modules. Change into the mode *linking active*, select the GCT and afterwards click onto the activating modules (only WST_6 in this case). Then the mode *linking active* can be left.

Now to GCT DT itself: In the decision table editor the references should be selected. These are all modules and junctions, to which one would like to refer within the decision table (and those not activating). In our example we want to change the status of the junction 3, therefore this is a reference.

Now we determine the conditions, on which an action is executed. For this example the breakdown status of WST_6 is relevant. One contains these by the module attribute **failsem**. If **failsem** is larger than 0, then the module is disturbed. Is **failsem** equal to 0, then no disturbance is present. The condition reads thus

act_module.failsem > 0

We remember: **act_module** is the selector for the current activating module. Since there is only one activating module (WST_6), is *act_module* unique.

Now we determine actions, which are to be executed. Interesting is that we will determine two actions. One is executed, if the condition is *fulfilled*, the other is executed, if the condition is *not fulfilled*.



In the action junction 3 is to be blocked. For this we use the junction operation **block** and **deblock**. The actions read thus:

junction(3).block
junction(3).deblock

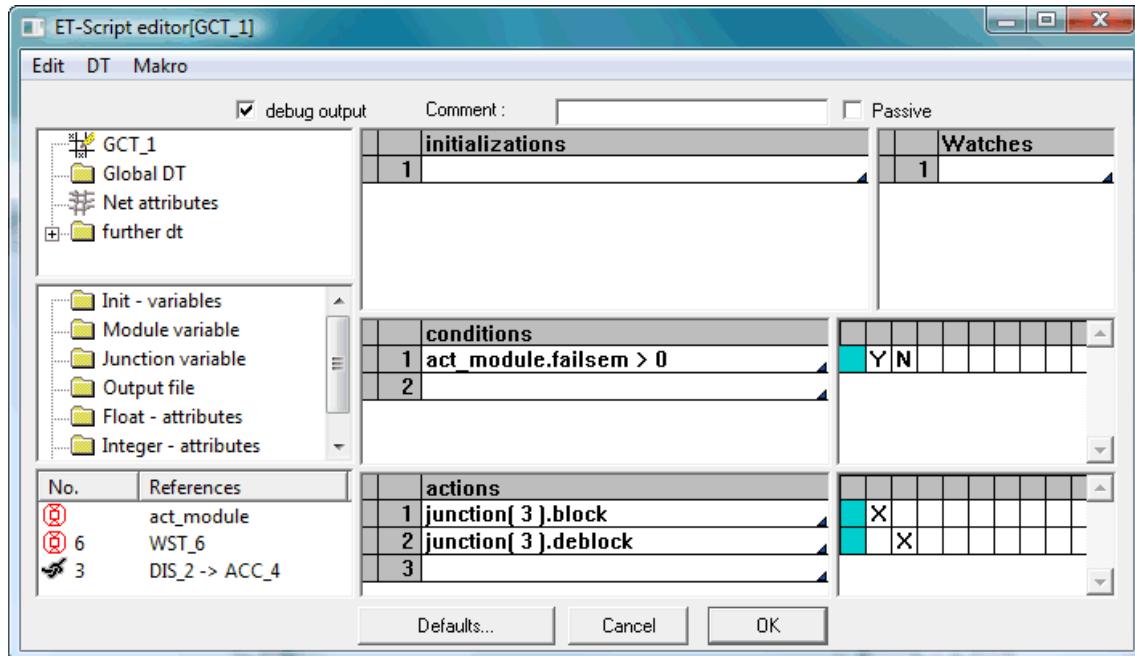


Figure 23.2: The decision table

Finally one determines, which action is to execute. For this the rule matrix serves. In the matrix line apart from the condition the entries *Y* in first and *N* in the second column are produced. These entries are for "YES" (fulfilled) and "NO" (does not fulfill). Now one follows the *Y-column* perpendicularly downward, until the suitable action line is met. Here a click into the matrix produces a marking in form of a cross. With the *N-column* exactly the same way is to proceed.

Now the decision table is instructed to do the following:

If the activating module is disturbed, then lock junction 3. Otherwise unlock junction 3.

Now all necessary is done. With each event in WST_6 the decision table checks whether the station is disturbed and locks/unlocks if necessary junction 3. The distribution module is parameterized in such a way that it does not consider outputs, which are blocked (junction 3 is blocked), any longer.

Note: Those GCT DT is called with each event in WST_6. In our example between disturbing and screening this workstation no further event will take place. Should this occur, by further modeling, but still another event, then junction 3 becomes closed several times. In this case it is advisable to introduced a net attribute, by which is noted, whether the junction already is blocked.



23.2 Example of a Distribution-DT

23.2.1 Description on the Examples

If the given distribution or right of way strategies should lead to a result any longer, then you can solve this problems also by a decision tables. The Dosimis-3 standard strategies considers only the statuses of neighboring modules. Such local strategies can be extended to global strategies using decision tables. Then any modules for the respective strategy formation can be consulted.

The use of decision tables for the solution of this problems is explained with the model from the previous example. Here we proceed from previous problems, but not using GCT DT and no disturbance.

The problem, which we would like to solve in this example, turns out as follows: If the cycle times are different in both sinks, then this will lead to a tailback in one of the branches. A distribution strategy, which reacts to the utilization of the two accumulation conveyors ACC_7 and ACC_8 would be ideal. Objects are distributed on that conveyor, which is less occupied.

To produce this behavior there are naturally several approaches. One could e.g. act with a GCT DT and lock the output junction of the distributor accordingly. Dosimis-3 offers however the possibility of defining a distribution strategy in form of a decision table.

To achieve this in the distributor, you select the strategy self-defined. At the bottom of the strategy selection now a button *Edit DT* appears. By this you reach the decision table editor and determine the strategy.

To determine the selection of an output you have to use the variable **selfdef**. You assign the number of the appropriate output to it.

We determine the number of the output on the basic of the occupancy of the two accumulation conveyors ACC_7 and ACC_8. If ACC_7 is more occupied than ACC_8, then we distribute to ACC_8 by output 2, in the other case we pass the objects to ACC_7 by output 1. When both are equal occupied, we use output 1 likewise.

The status of ACC_7 and ACC_8 is thus interesting for us. These are the referencing modules. There are not activating modules with distributing DTs. The distributing DT is activated, whenever Dosimis-3 must examine an output for the next object in a distributor.

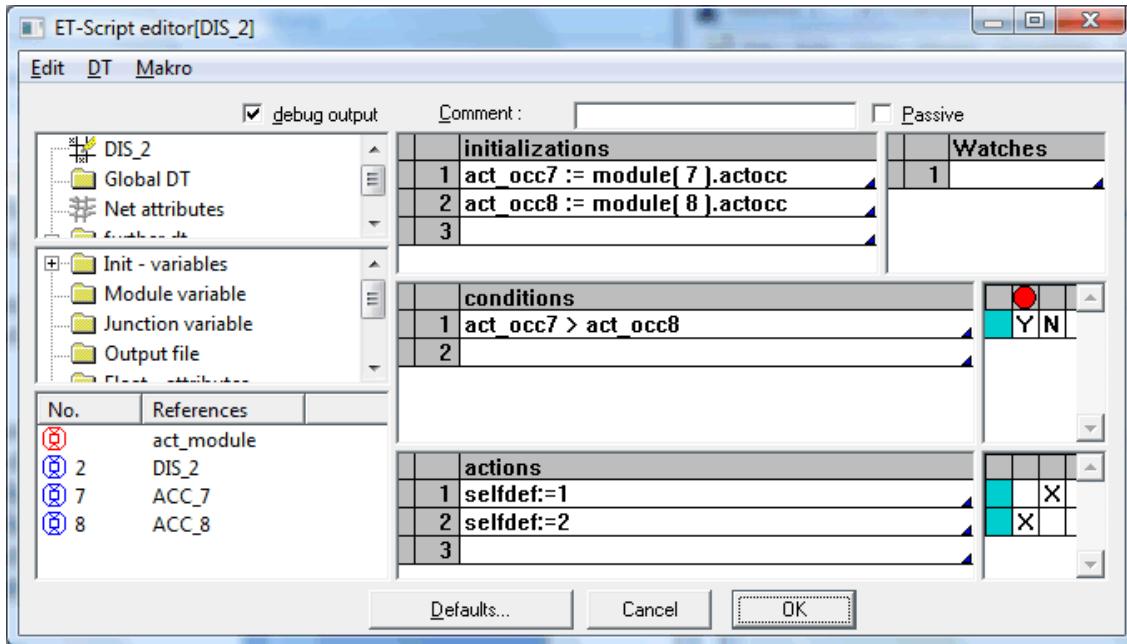


Figure 23.3: the distributing decision table

To find out the occupancy of a module you can use the attribute **actocc**. From this the following condition results:

module(7).actocc > module (8).actocc

As an actions we assign the number of the appropriate output to the variable **selfdef**. When the condition is fulfilled, the next object is to be led by output 2:

selfdef:= 2

otherwise output 1 is used

selfdef:= 1.

23.3 ATS-Model with global DT-Data

Figure 23.4 shows a simple material flow system, including two separate production areas (1 and 2) of the same type, provides together by one ATS-system. The distribution strategy for both areas is an occupation-oriented providing of the single machines. It is slightly to recognize, that the control strategy must be based on the same principle for both areas. Without global DT's for the control strategy each area must have its own decision table. This two decision tables would be very similar.

The setting of the system process can be described as follows:

The AGV-vehicle reaches with its loading the point of information. At this point, two important decisions are to be taken:

- Selection of an area (1 or 2).
- Selection the machine with the lowest occupancy.



The ranges differ concerning the number of the machines. Area 1 contains two, area 2 contains four machines.

At this point now, the priorities of the **global decision tables** should be presented by making use of global module- and junction variables.

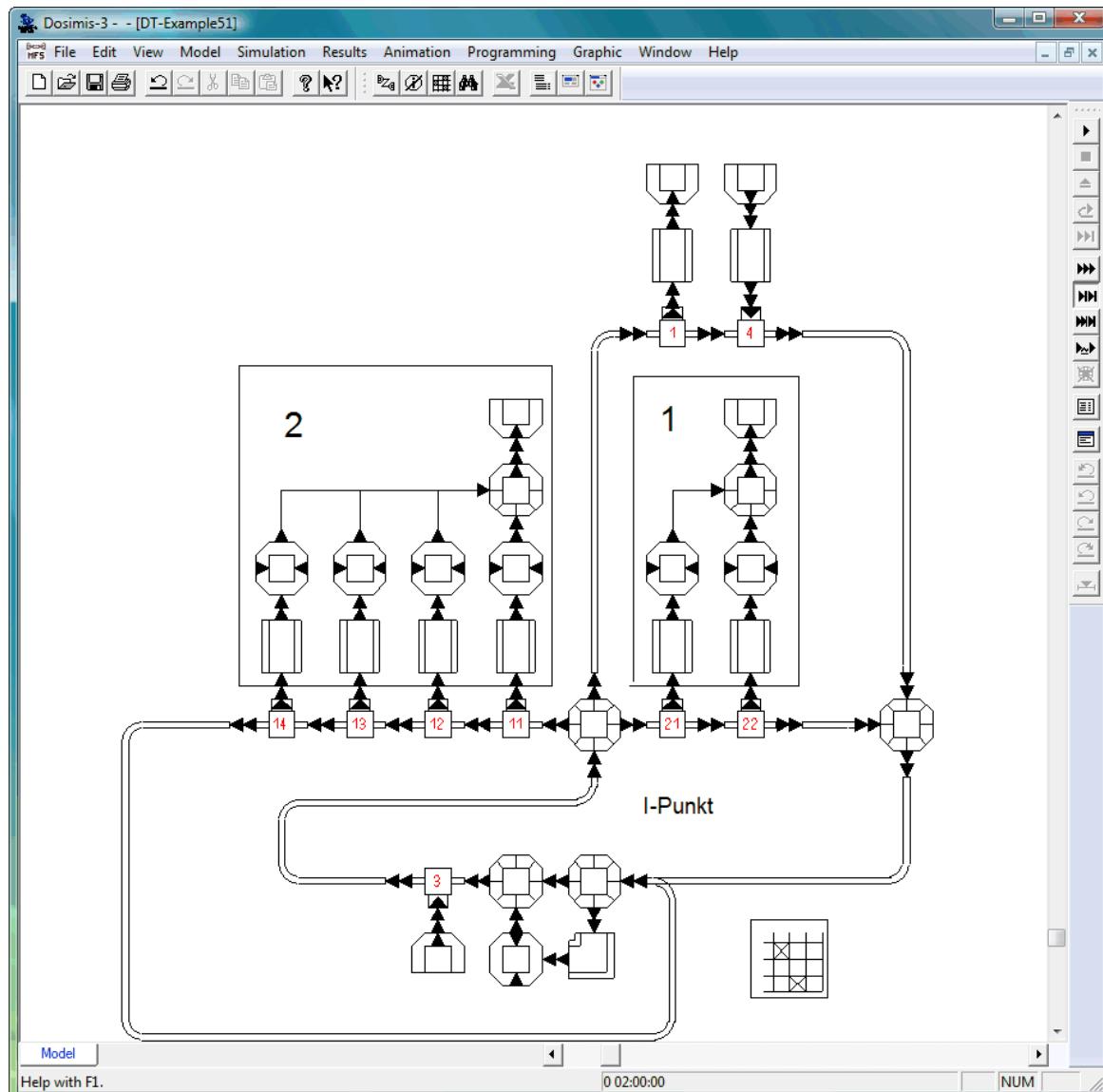


Figure 23.4: MFS of the example

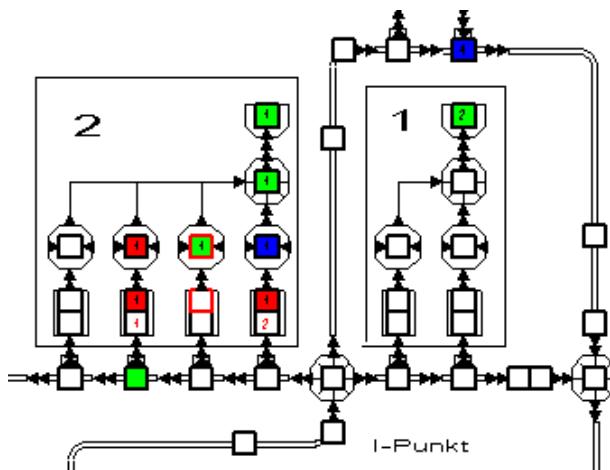


Figure 23.5: Overview about the production areas

23.3.1 Solution by using a global DT

The solution of that problem solution pursues the goal to implement the allocation-oriented supply of the two different areas by one algorithm for both areas.

The track before the crossing (point of information) becomes the active GCT module, which calls a global DT for each AGV reaching the point of information. The GCT decides already whether the load of the AGV is disposed for the left or right area. If the AGV has the *destination code* 1, then it is intended for the left area, with the *destination code* 2 it is to turn on the right.

In order to determine now the exact destination, the occupancy of all accumulation conveyors of one area must be compared. This ensures a global DT. The trick is as followed: By means of module variable (*mod*) it is stepped from unloading station to unloading station. If that is found, whose attached accumulate conveyor contains the smallest occupancy , then the destination code of this station is stored in the net attribute *new_unload_id*.



The GCT DT must initialize this module variable *mod* thus with the correct (left or right) unloading station. From the model one infers that the unloading station on the left is linked by output 1 of the crossing. The unloading station on the right linked to output 3. The crossing has the module number 7 (and is reference module). Thus we define GCT DT as follows:

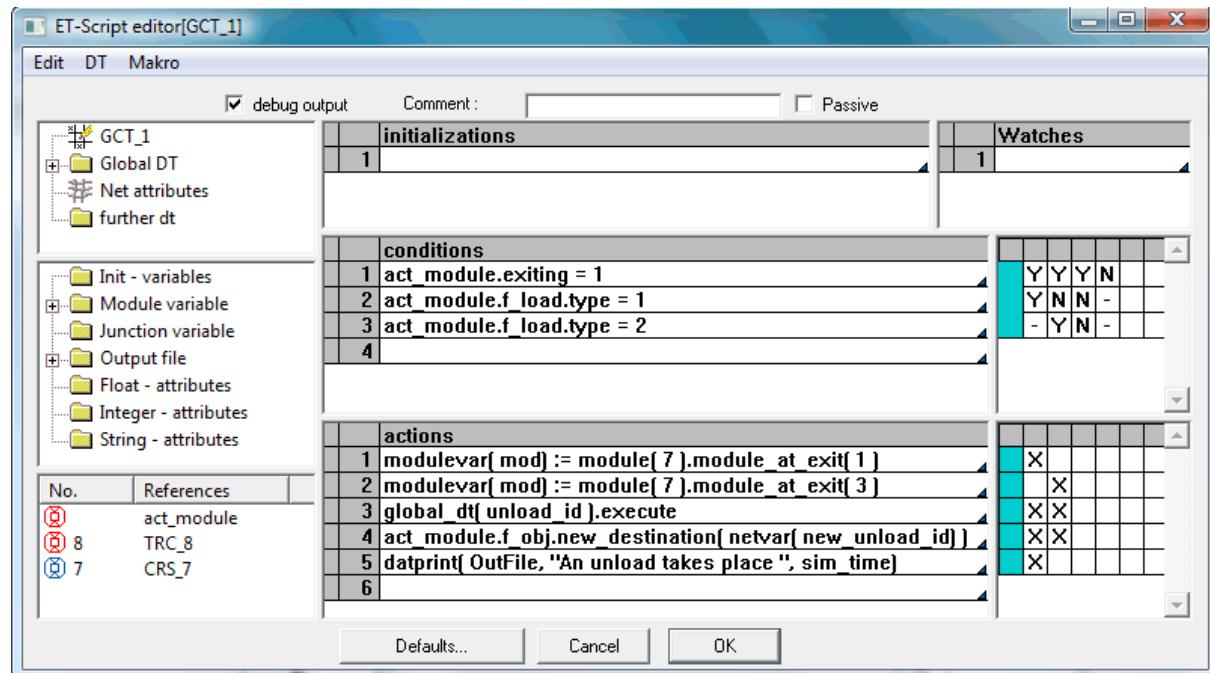


Figure 23.6: Die GCT-Decision Table

condition 1: **act_module.exiting = 1**

condition 2: **act_module.f_load.type = 1**

condition 3: **act_module.f_load.type = 2**

action 1: **modulevar(mod):= module(7).module_at_exit(1)** [with condition 1 and 2 true]
 action 2: **modulevar(mod):= module(7).module_at_exit(1)** [with condition 1 and 3 true]
 action 3: **global_dt(unload_id).execute** [every time act. 1 or 2]
 action 4: **act_module.f_obj.new_destination(netvar(new_unload_id))** [every time act. 1 or 2]

The global DT should start with the module, which is referenced by the module variable *mod*. A sub-table of these global DT is provided, to find the accumulation conveyor with the lowest current occupancy and to assign this module to the module variable. If it is smaller, the net attribute *min_act_occ* is set to this value. Additionally we notice the appropriate unloading station in the module variable *min_mod*.



The global DT thus has nothing further to do as to call the sub-table in a loop and afterwards to find the next unloading station. Finally the destination of the AGV is set on the destination parameter, which is available in the noticed (module variable *min_mod*) module.

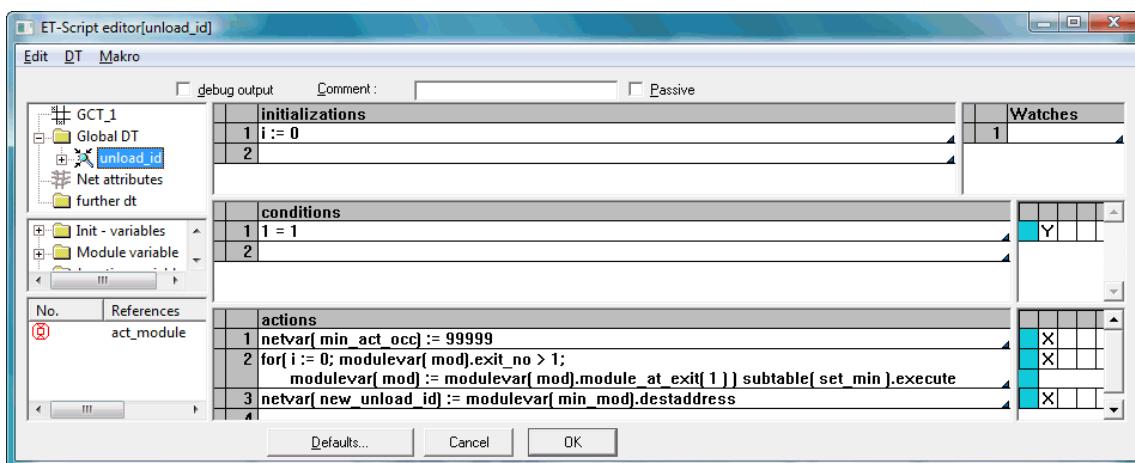


Figure 23.7: The global decision table

The global DT has the following structure:

initialization 1: **i := 0**

condition 1: **1=i**

action 1: **netvar(min_act_occ):= 99999**

action 2: **for(i := 0;modulevar(mod).exit_no > 1;**

**modulevar(mod):= modulevar(mod).module_at_exit(1))
 subtable(set_min).execute**

action 3: **netvar(new_unload_id) := modulevar(min_mod).destaddress**

Assertion: The initialization variable *i* is used as control variable for the for loop in action 2. There it is meaningless, but syntactically necessary. The condition is always true and therefore all actions are always executed. As the first action we set the variable *min_act_occ* to a very high value. Thus we guarantee that we always find an accumulation conveyor with a smaller occupancy .

Attention: The **for**-loop checks whether the module of the variable *mod* has more than one exit. If this is the case, then it concerns to an unloading station (according to the model, only unloading stations are applicable) and we execute the sub-table *set_min*. If the module has only one exit, then it cannot concern an unloading station. In this case we finish. After each run we set the module variable *mod* again: It is referencing the module, which is at output 1 of the unloading station. This should be either a further unloading station, or however a module at the end of the chain of unloading stations, for which the loop condition (**modulevar(mod).exit_no > 1**) becomes false. *set_min* is executed for each unloading station and notes that accumulation conveyor with the minimum occupancy in the variable *min_mod*.



Subsequently, at the complete operational sequence of the loop, in action 3 the AGV destination is set on the destination code of the found unloading station.

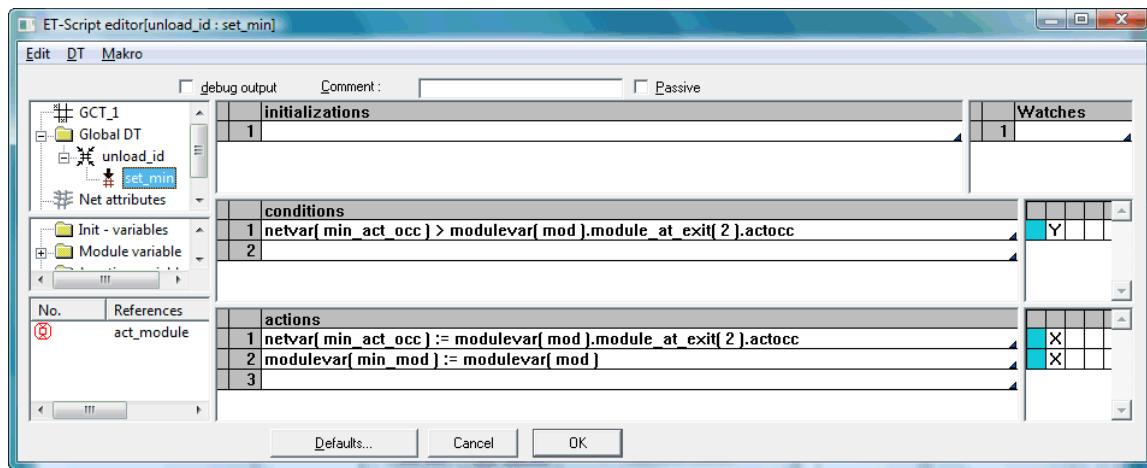


Figure 23.8: The sub decision table

The sub-table `set_min` executes as followed:

condition 1: `netvar(min_act_occ) > modulevar(bs).module_at_exit(2).actocc`

action 1: `netvar(min_act_occ) := modulevar(bs).module_at_exit(2).actocc`

action 2: `modulevar(min_mod) := modulevar(bs)`

Both actions are executed, when the condition is true. The condition checks thus the actual occupancy of the module *mod* at exit 2. This is in accordance with the model an accumulation conveyor. If the actual occupancy is smaller than found so far (stored in *min_act_occ*), then we save this new minimum occupancy back in *min_act_occ* (action 1) and note the unloading station the accumulation conveyor leads to (action 2).

This example demonstrates impressively the use of global variables (net attributes and module variables) for the transfer of information between decision tables. Additionally one detects here the possibilities of **for** loops and the solve of sub-problems in different decision tables.



24 Shop Floor Control

24.1 Introduction (Shop Floor Control)

In Dosimis-3 - Models objects can be processed according to work plans. In addition production orders are provided, which produce several objects. These objects are processed afterwards according to a work plan. During the treatment of work plans by the simulator all the following things are of interest:

- Producing objects in accordance to order dates
- Working on these objects in several work stations in accordance with a work plan
- The indication of setup and operating time in these work stations
- The automatic path finding for objects in accordance with the sequence of operations
- The work plan-referred determination of object turn-around times.

24.2 Work plan

The term of work plans is doubly occupied (unfortunately). On the one hand it designates a collection of operations, which provide for the processing of objects in work stations, on the other hand work plans means also the quantity of all orders and work plans.

The work plan is divided into as many as desired operations. Each of these operations takes place in a work station. The objects are transported automatically by an automatic path finding to the correct work station. There the object is processed, whereby work and setup behavior is defined by the work plan (or more exactly: by the operation). The parameters of the work station have no meaning for work plan objects thus. So that the objects achieve their destination work station it is necessarily to switch on the *automatic pathfinding for work plans* in all modules with distribution strategy which can be passed by objects with work plans.

After end of processing the next operation is selected. For this each operation has as much as desired operation references, which define the probability, with which a subsequent operation is accomplished.

The processing of a work plan always begins with an order. This produces a given quantity of objects in a source and sets these objects under the administration of the work plan.

The destinations `::Ready::` and `::Reject::` require special attention. These terminate the processing of an object by work plan. The object moves thus from now on as "normal" Dosimis-3 object through the material flow system. Its own work plan statistics shows the kind of the completion, thus whether an object with "finished" or "reject" was terminated.



24.3 The Shop Floor Control Editor

24.3.1 Overview

By means of the menu of "model/work plans" one opens the work plan editor. This is divided into two sections: A structure view on the left and a data view on the right.

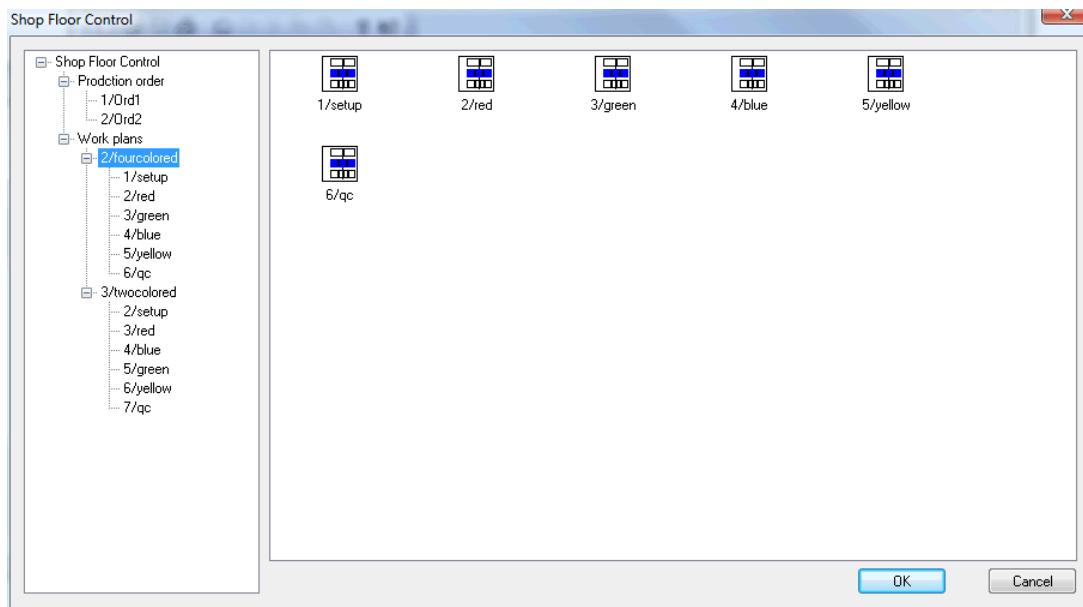


Figure 24.1:Editor shop floor control

The structure view always contains the folders of *Production order* and *work plans*. By opening these folder (with a double click, operation as in the Windows Explorer) one reaches the appropriate data.

A right-click in the structure view opens a context menu, which offers the possibilities for creation and deletion of orders, work plans and operations.

If one leaves the dialog within the right area (by click on OK or selection of another object in the structure view), the entered data are fundamentally examined. Only if all data is valid, one can actually leave the dialog.



24.3.2 Work Plans and Operations

A work plan contains a sequence of operations. Work plans and operations can be created in two different ways: By opening the context menu (right-click) within the left area of the editor or by referencing when defining an order (and/or by a reference to a non existing operation).

The work plan does not carry own information except its name. All processing is described in the operations.

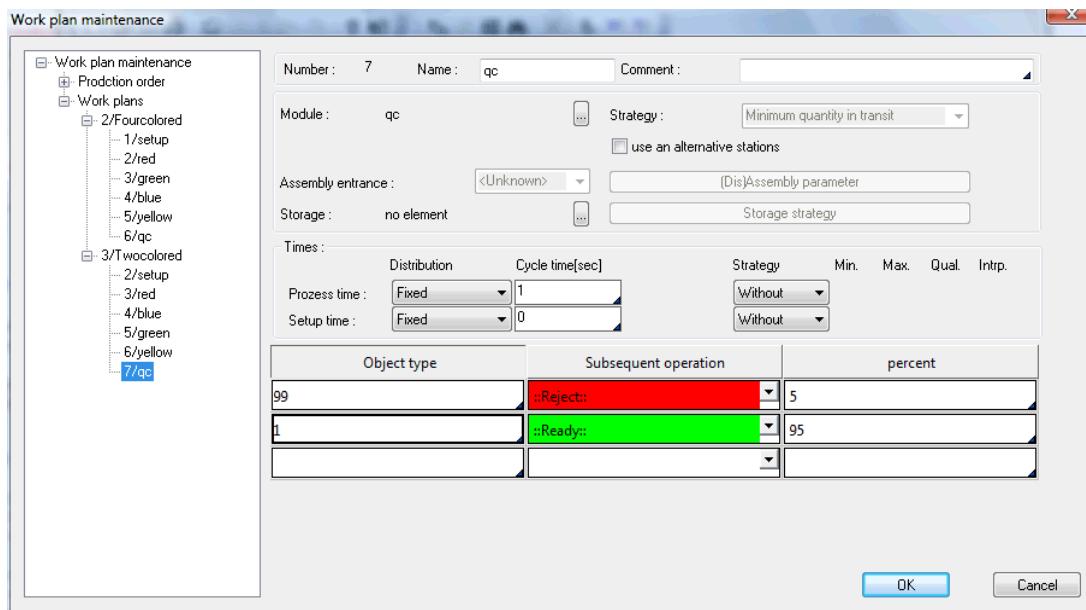


Figure 24.2: Operation

If one opens an operation by selection in the structure view, then one can specify the name of the operation within the right area. By this name the operation is referenced.

The selection of modules is made by the button (…). With the operations work stations, assemblies or disassemblies can be selected. Here it is also possible to select several modules. Then however all modules must be of the same type. If an object reaches one of the declared work stations, then the operation is accomplished at the work station reached. This concept is called *alternative stations*.

Process time and setup time define the appropriate times during processing of objects. Here also strategies for the worker employment can be defined. The parameterizing follows the proceeding usual for Dosimis-3.

Within the lower area references to operations can be specified. An operation reference declarexxs, with which probability (uniform distributed) a following operation is accomplished, and/or which is terminated processing by the work plan.

The first column declares the type of object, which the lot is to receive, if it follows the reference. The indication is optional. If a wildcard (*) is used, then the lot keeps its type of object. The possibility to change object types is in particular interesting for visualization and the error tracing, in addition, for the correct subsequent treatment of lots, which refer to



::Ready:: and ::Reject::. These can be identified by different object types after leaving the work plan for further treatment in the model.

The second column of a reference marks an operation. Here one can select from all existing operations of the current plan. Additionally the possibility exists of creating a new operation. The appropriate operation is then created automatically. As a special case the references ::Ready:: and ::Reject:: exist. A lot that references one of these destinations leaves control by work plan. It is led in the work plan statistics accordingly as *Ready* or a *Reject* and moves in accordance with the parameters of the model.

The third column defines the probability (in per cent) the operation follows.

24.3.2.1 Selection of Modules

Click the selection button (…) to select a module. Then a dialog opens where you can select one or more modules, depending on the context, in which a selection takes place. The modules must be however all of the same type. Subsequently, one clicks *close*.

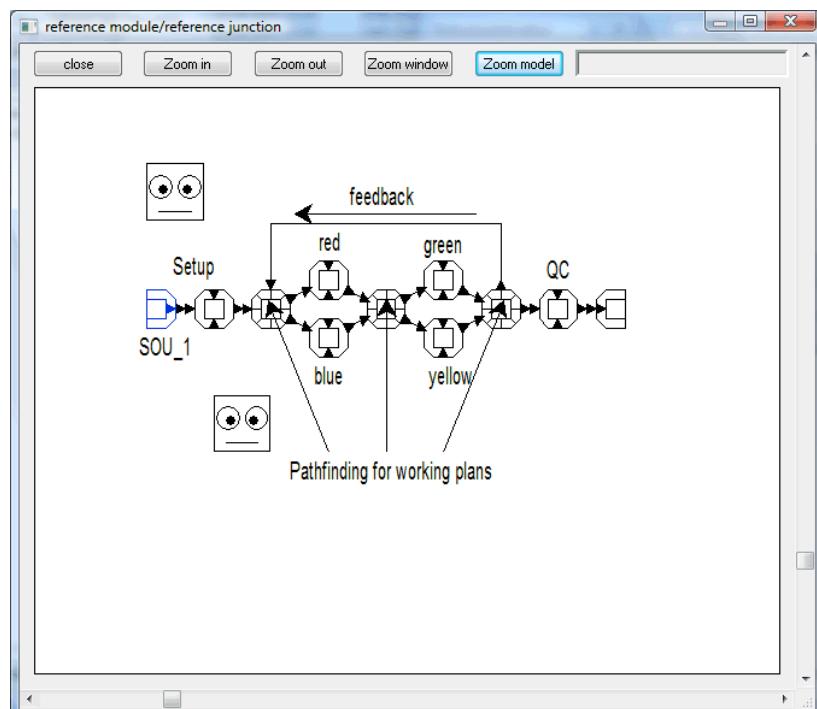


Figure 24.3: Bausteinauswahl

24.3.2.2 Alternative Stations

If several stations belong to an operation, then the strategy, which of the possible alternatives is selected, can be specified. The following strategies are available:

Module :	WST_2, ...	Strategy :	Minimum quantity in transit
		<input type="checkbox"/> use an alternative	First module Minimum quantity in transit
Assembly entrance :	<Unknown>	(Dis)Assembly parameter	
Storage :	no element	Storage strategy	

Figure 24.4: Alternative Stations



- **First Module:**

All objects are scheduled to the first module. The following is to be considered with this strategy: The mechanism „automatic pathfinding for work plans“ in the distribution strategies ensures now that an object arrives at the first work station defined here. If one wants to process the operation at one of the other stations, then the automatic pathfinding has to be switched off and be replaced by a strategy, which leads the object (guaranteed) to one of the stations referenced in the operation.

- **Minimum Quantity in Transit:**

The object is scheduled to the station, to which few objects are bounded for.

24.3.2.3 Assembly Operation

If the station is an assembly, then an entrance is to be selected at **Assembly entrance**.

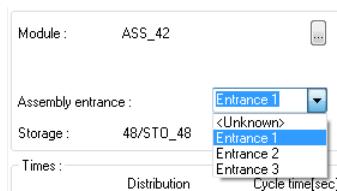


Figure 24.5:Assembly Entrance

2 alternatives are to be distinguished. If an entrance > 1 is selected, the object is navigated to the appropriate entrance. The operation must be the last one of the work plan. If the entrance 1 is selected, then it is to be specified which objects are to be assembled with the object. This takes place by the switch (**De)Assembly parameter**.

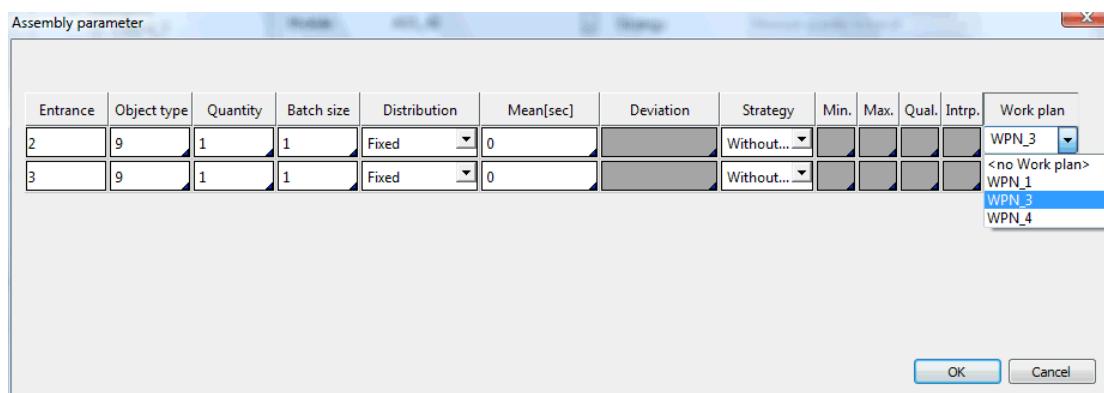


Figure 24.6: Assembly Operations

The list corresponds to an assembly list of the assembly module. Additionally the work plan, which the object has to possess, can be still defined. If **<no work plan>** is selected in the last field, only the type of object at the assembly entrance is examined.



24.3.2.4 Disassembly Operations

If the station is a disassembly, then in the **(De)Assembly parameter** it can be specified, how the disassembly has to take place.

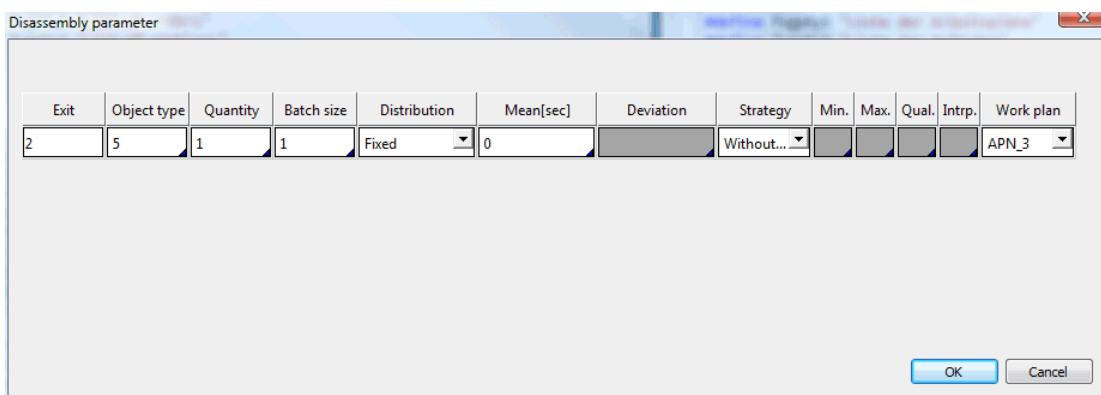


Figure 24.7: Disassembly Operations

A work plan can be assigned to every disassembled object in the last input field. If one selects **<no work plan>** there, a “normal” object is disassembled.

24.3.2.5 Warehousing

A storage can be assigned to each operation. The use of the storage can be specified by the switch **Storage strategy** more exactly.

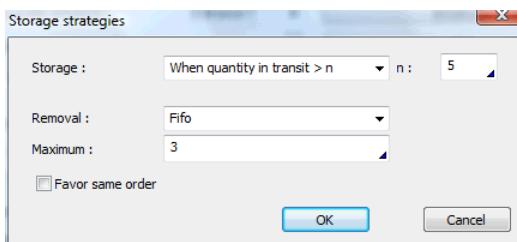


Figure 24.8: Storage Strategies

On the one hand it has to be specified, when an object is to be stored. For this the following strategies are available:

- **Always:**
Each object is stored in the storage.
- **When quantity in Transit n:**
The object is stored, if too many objects are in transit for a station. The value of n can be defined in the subsequent field more exactly.
- **Never:**
The object is never stored (to test purposes).

In order to remove the stored object again from stock, can be specified under **removal**, when and according to which criteria an object is removed from stock. It is made certain that the number of removed objects does not exceed the value of **Maximum**. The strategies are:

- **Fifo:**
The object, which lies longest in the storage, is outsourced.
- **Priority of Order:**
A stored object of the order, which possesses the highest priority, is removed.



This strategy can be overruled with the switch **Favor same order**. Then first an object is removed, which belongs to the same order as the object, which left the station. Only if such an object is not found, the removal takes place in accordance with the strategy.

24.3.3 Production Order

If one selects an order in the left structure view, then the associated order data appear on the right side of the editor.

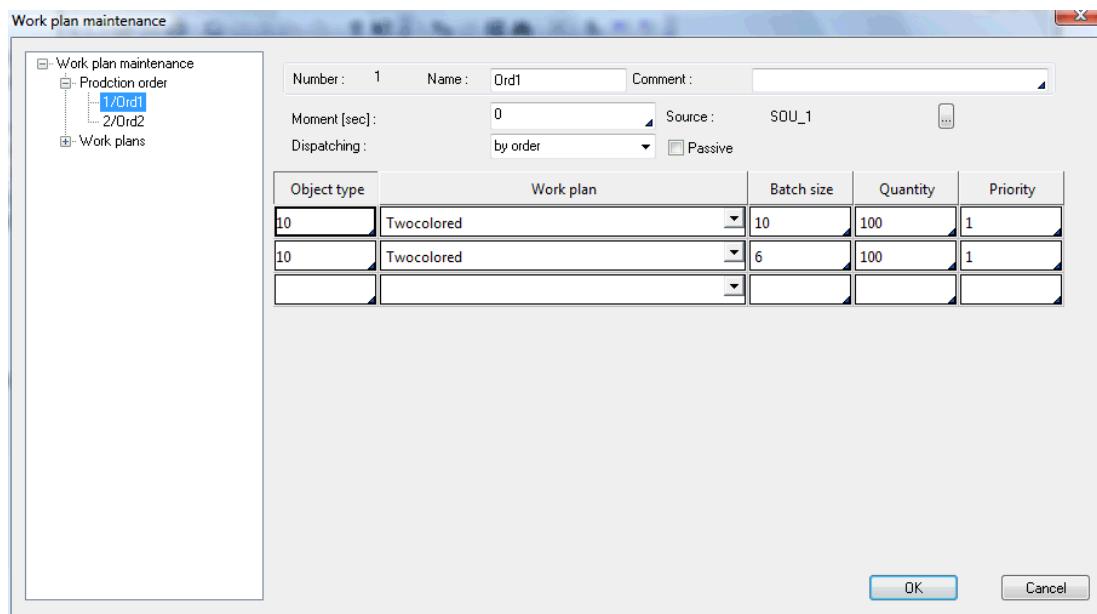


Figure 24.9: Display of order

Within the upper area one can specify an arbitrary name for the order. The name does not have any further meaning except the information for the user. Further one can define the source of an order, the creation time and the work plan which should be used. If one selects a work plan, which does not exist yet, then these is created when leaving the order dialog.

At Dispatching one can define, how the objects are created:

- **Single:**
One object of the order will be created. When the source is empty, the next object will be created.
- **By partial order:**
All objects of a partial order (one line in the table of objects) will be created at once. When the source is empty, all objects of the next partial order will be created.
- **Complete:**
All objects of the order will be created.

In the lower area is the table of objects. This defines which objects are created by the order. In the first column one defines the type of object. The second column defines the **work plan** of the object. The third column defines the size of the lot which can be created, in the fourth column contains the **quantity** of the order and the last column the **priority**.

A lot with quantity N is created in Dosimis-3 as a single object. It is processed in the work stations however as N parts.



Example: A lot with lot size of 10 and type 40 is treated in Dosimis-3 as a single object of the type 40. In a work station with work time of 5 seconds the lot however needs 50 seconds for the processing (5 seconds * lot size).

24.4 Result Graphic

24.4.1 Diagrams

The statistics can be plotted in the Dosimis-3 editor by the menu of *Results/Shop Floor Control/Throughput-time statistics*. This representation is a bar chart with the lots ready/rejected of each work plan on the x axis and the throughput times on the y axis. Here minimum (blue), average (green) and maximum throughput time (red) as bars are put one behind the other. The values are represented separately according to the shop floor control, order number and lot size (**Sfc/Ord/Lot**). Under the number 0 the global shop floor control, which can be reached from the menu *Model/Shop Floor Control*, is provided.

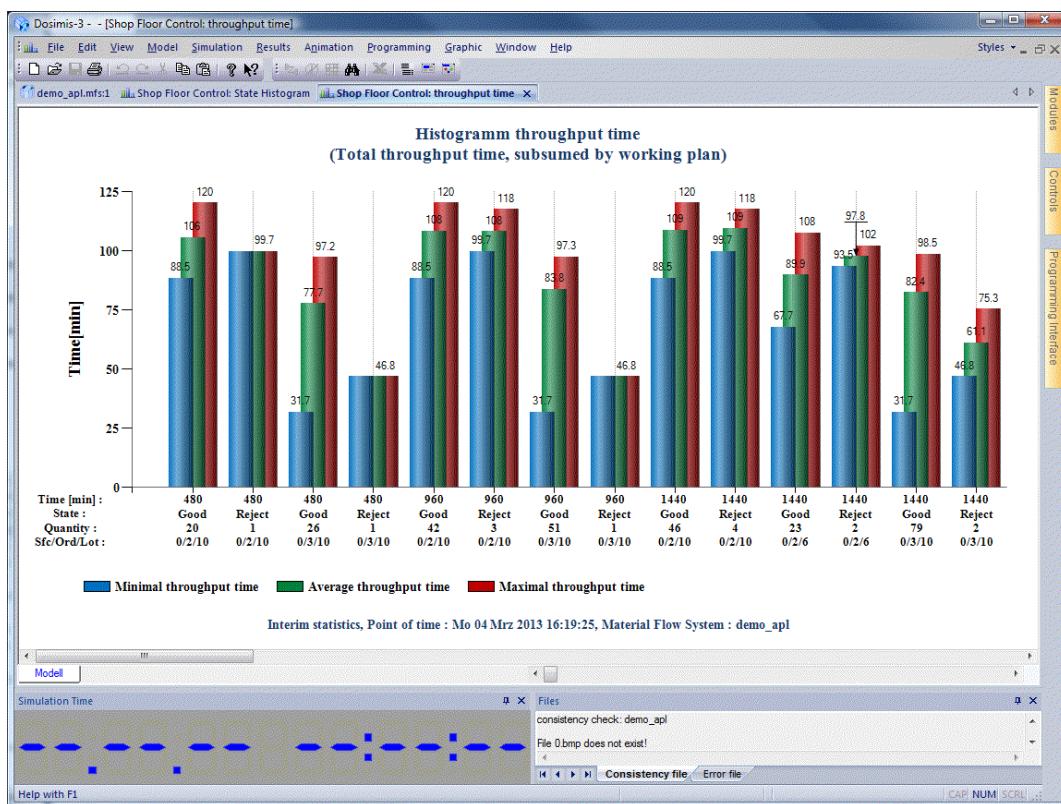


Figure 24.10: Shop Floor Control –Throughput time diagram



Additionally the states of the production orders can be shown in a state diagram.

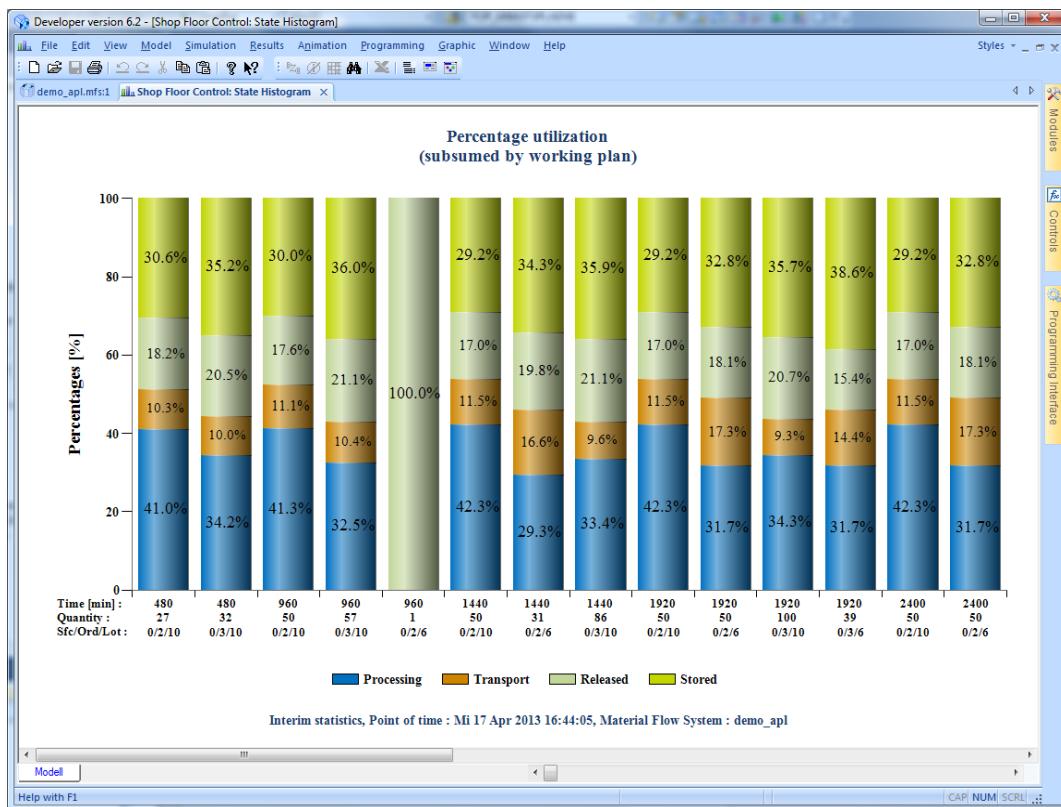


Figure 24.11: Shop Floor Control-State diagram

The diagram can be adjusted with the result parameters of the tab **Shop Floor Control**.

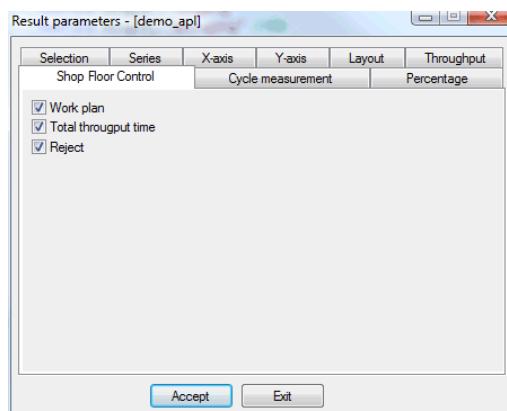


Figure 24.12:Shop Floor Control-Result parameter

Work Plan

The statistics of the work plan are presented. Otherwise, production order information is displayed.

Total Throughput time

In the throughput time diagram the total throughput times are represented. Otherwise, only the throughput time from first operation to the last operation is drawn.

Reject

With this switch the information of the rejected - objects can be faded out.

Additionally the representation can be affected by the following result parameters:



- From the tab *Series* different mean values can be activated:
 - Mean value sub area: Mean value of ready and reject parts.
 - Mean value selection: Mean value of all selected shop floor controls.
- The representation can be limited by changing the *limits* in the tab *selection*. The only the orders/work plans with their numbers inside these limits are shown.

24.5 Result Table

The statistics of the shop floor control can be found in the statistics file *modelname.slg*.

24.5.1 Statistic File

Several different analysis of the shop floor control can be found in the statistics file. These are concentrated according work plan and production order. The throughput time statistics are conform to the standard throughput time statistics. These are differentiated according several criteria: work plan, lot size and process result.

Shop floor-control	Name of-workingplan	Lotsize	State	Count of Objects	Average tpt	Minimal tpt	Maximal tpt
0 Global	2 fourcolored	10	Good	46	108.77	88.52	120.35
0 Global	2 fourcolored	10	Reject	4	109.47	99.68	117.52
0 Global	2 fourcolored	6	Good	44	87.65	65.52	107.68
0 Global	2 fourcolored	6	Reject	6	94.02	85.35	102.02
0 Global	3 twocolored	10	Good	98	80.95	31.68	98.52
0 Global	3 twocolored	10	Reject	2	61.10	46.85	75.35
0 Global	3 twocolored	6	Good	65	46.70	37.68	65.68
0 Global	3 twocolored	6	Reject	5	45.68	40.85	52.85

Figure 24.13: Work plan statistics: Duration from release to finished

The total process time is the duration from the release of the order until end of the last operation.

Shop floor-control	Name of-workingplan	Lotsize	State	Count of Objects	Average tpt	Minimal tpt	Maximal tpt
0 Global	2 fourcolored	10	Good	46	62.30	51.52	68.35
0 Global	2 fourcolored	10	Reject	4	63.06	58.52	66.02
0 Global	2 fourcolored	6	Good	44	47.89	36.85	57.02
0 Global	2 fourcolored	6	Reject	6	53.02	47.18	56.35
0 Global	3 twocolored	10	Good	98	37.65	26.18	50.35
0 Global	3 twocolored	10	Reject	2	36.60	32.18	41.02
0 Global	3 twocolored	6	Good	65	25.54	16.02	34.18
0 Global	3 twocolored	6	Reject	5	24.95	19.18	33.85

Figure 24.14: Work plan statistics: Duration from first operation until finished

The work plan process time is the time from start of the first operation until the end of the last operation.



```
*****
Final statistics          : Working plan state
Material flow system     : demo_apl
Statistics for time      : 4320.0
*****
Shop floor-control       Name of- Lotsize State   Count Working Trans- Released Stored
                           workingplan           [%] port[%] [%] [%]
-----
0 Global                 2 fourcolored    10    50  1.26  42.27  11.50  17.00  29.23
0 Global                 2 fourcolored    6     50  1.02  31.71  17.34  18.13  32.81
0 Global                 3 twocolored     10    100 1.86  34.33  9.31   20.72  35.65
0 Global                 3 twocolored     6     70  0.76  35.67  15.16  12.36  36.81
*****
```

Figure 24.15: Work plan statistics: States

The state statistics supports 4 different states

- **Released:** The order is created in a source and waits for leaving the module.
- **Transport:** the objects has left a module or processing is finished. The time contains blockings, tranposrt (by agv), ...
- **Stored:** Two situations are considered as stored. The object has reached a storage module or the object enters a module directly in front of the next station (exit 1 is connected with the station of the next operation, receipt buffer of the station).
- **Working:** the object is processed at a station. This is the time from start of processing until end of processing, including waiting for worker, setup time,

The statistics are provided in a second variant. Instead of work plans the values are collected according the production order.

```
*****
Final statistics          : Order-Throughput-time total (data in minutes)
Material flow system     : demo_apl
Statistics for time      : 4320.0
*****
Shop floor-control       Name of order State   Count of Average Minimal Maximal
                           objects      tpt      tpt      tpt
-----
0 Global                 1 Ord1        Good    163    67.29  31.68  98.52
0 Global                 1 Ord1        Reject   7     50.09  40.85  75.35
0 Global                 2 Ord2        Good    90     98.45  65.52  120.35
0 Global                 2 Ord2        Reject   10    100.20  85.35  117.52
*****
```

Figure 24.16: Statistics of production order: Duration from release until finished

```
*****
Final statistics          : Order-Throughput-time order (data in minutes)
Material flow system     : demo_apl
Statistics for time      : 4320.0
*****
Shop floor-control       Name of order State   Count of Average Minimal Maximal
                           objects      tpt      tpt      tpt
-----
0 Global                 1 Ord1        Good    163    32.82  16.02  50.35
0 Global                 1 Ord1        Reject   7     28.28  19.18  41.02
0 Global                 2 Ord2        Good    90     55.25  36.85  68.35
0 Global                 2 Ord2        Reject   10    57.03  47.18  66.02
*****
```

Figure 24.17: Statistics of production order: Duration from first operation until finished



```
*****
Final statistics          : Production order state
Material flow system     : demo_apl
Statistics for time      : 4320.0
*****
Shop floor-control       Name of order   Count Active Working Trans- Released Stored
                           [ ]           [%]       [%]       [%]       [%]
-----
0 Global                 1 Ord1          170  2.62   34.71  10.99  18.31  35.98
0 Global                 2 Ord2          100  2.28   37.54  14.12  17.51  30.84
*****
```

Figure 24.18: Statistics of production order: States

```
*****
Final statistics          : Order statistics (data in minutes)
Material flow system     : demo_apl
Statistics for time      : 4320.0
*****
Shop floor-control       Name of order   Gen.- Start    End     Target
                           order        time   time     time    count
-----
0 Global                 1 Ord1          0.000  0.667  2113.517 170
0 Global                 2 Ord2          20.000 41.833  1812.517 100
*****
```

Figure 24.19: Statistics of production order: Points of time

This statistic contains the exact moment in time when the first object of a production order has been created (*Gen.-time*), when the first object of an order starts the first operation (*Start time*) and when the last object of an order finished the last operation (*End time*). Additionally the target count can be seen.

24.6 Old File Formats of Work Plans

In Dosimis-3 versions before release 4.1 work plans have been stored in the model directory in a file ending on APL. Starting from version 4.1 all work plan data as well as the material system flow are stored of in the MFS file. If still another APL file should be present in the model directory, then this is read when opening the flow of material system in the editor and it is renamed before storing the data (ending: APL.BAK). This APL.BAK file serves only as backup and does not continue to be relevant for the model, since now all work plan data is in the MFS file.

With the use of old work plans the following is to be considered. Alternative work stations were specified by same names. This proceeding is changed. The ambiguous naming is not necessarily any longer, since there are strategies, from which an alternative station can be selected. Additionally in many cases the distribution strategy *minimum allocation* was selected, in order to achieve an alternative station. Also this does not lead now no more to the desired behavior. In the distribution before the station now the *Pathfinding for work plan* should be activated.



24.7 Example

24.7.1 Model

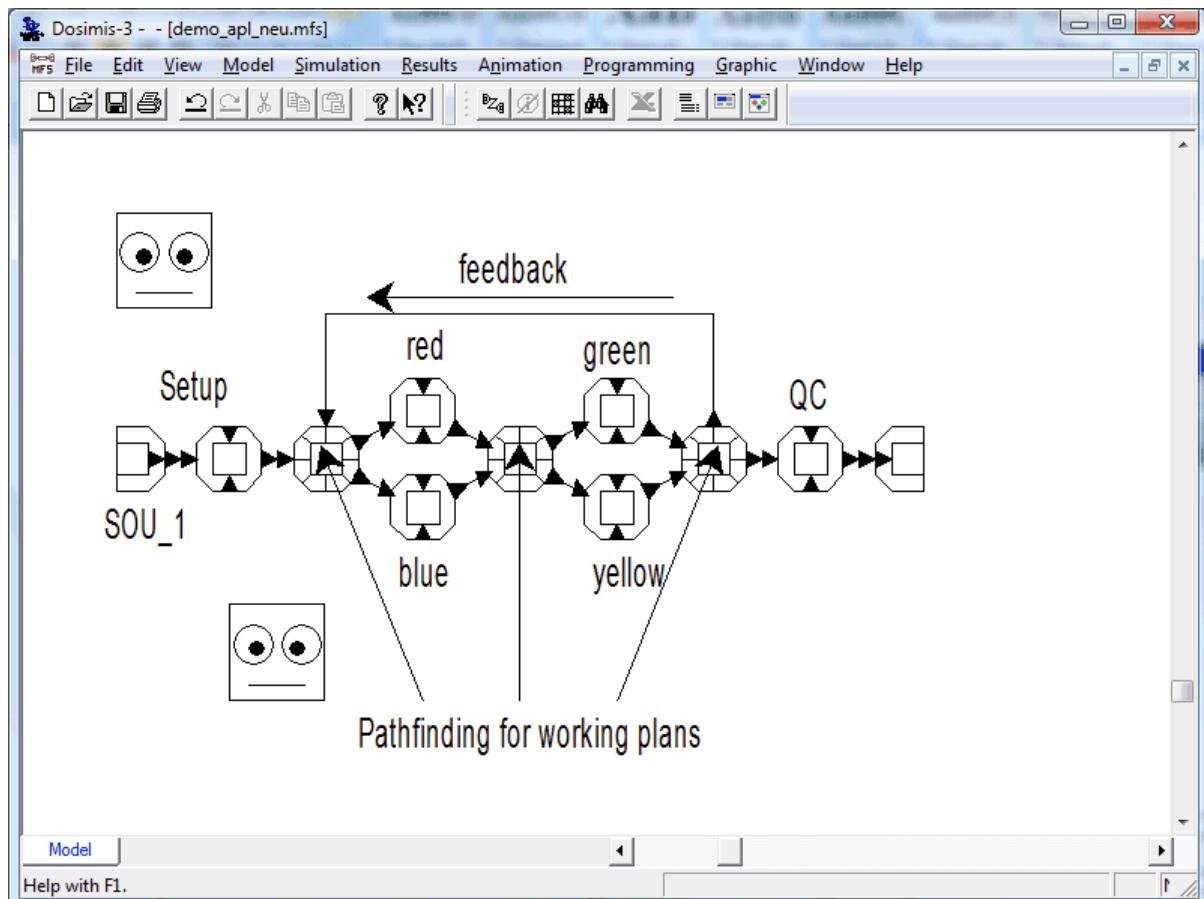


Figure 24.20: Example

24.7.2 Problem

To model is a manufacturing (lacquer finish) of parts. It exists a work station "setup" (general setup for all objects), one work station for every color which can be created (red, green, blue and yellow), as well as a finally quality control (rejects: 5%).

The colors red and blue as well as green and yellow are never used in sequence, therefore the appropriate painting stations can be attained by means of a common way.

There are four product types: two-colored and four-colored products in each case in lots of 6 and 10 parts. The two-colored variant receives its color coincidentally. With 50% probability it is only painted either red or blue, selected afterwards with 50% probability between green and yellow. The four-colored products always go through the cycle red, green, blue, yellow.

24.7.3 Modeling

The modeling was consciously selected simply. Each painting station, the setup and quality control were converted in each case as work stations. These were connected by crossings. It exists a way back, which is necessary for the four-colored products (way from green to blue). Since from this feedback deadlocks can result, a simple deadlock control was selected here:



The control with capacity 4 supervises all modules inside the feedback - thus all crossings and the painting stations. Consideration here: The shortest way by the modules contains five buffer places (first crossing, a painting station red/blue, middle crossing, a painting station yellow/green, rear crossing). If one permits only an allocation with 4 objects to 5 buffer places, then one place always remains for circling.

24.7.4 Order

Two orders exist. In the first order two-colored products are createxxd, 100 lots each with 10 parts and 100 lots each with 6 parts. This order is created at the time 0 in source 0.

In the second order four-colored one createxxs, ever 50 lots to 10 parts, further 50 to 6 parts. Reading in time is after 1100 seconds.

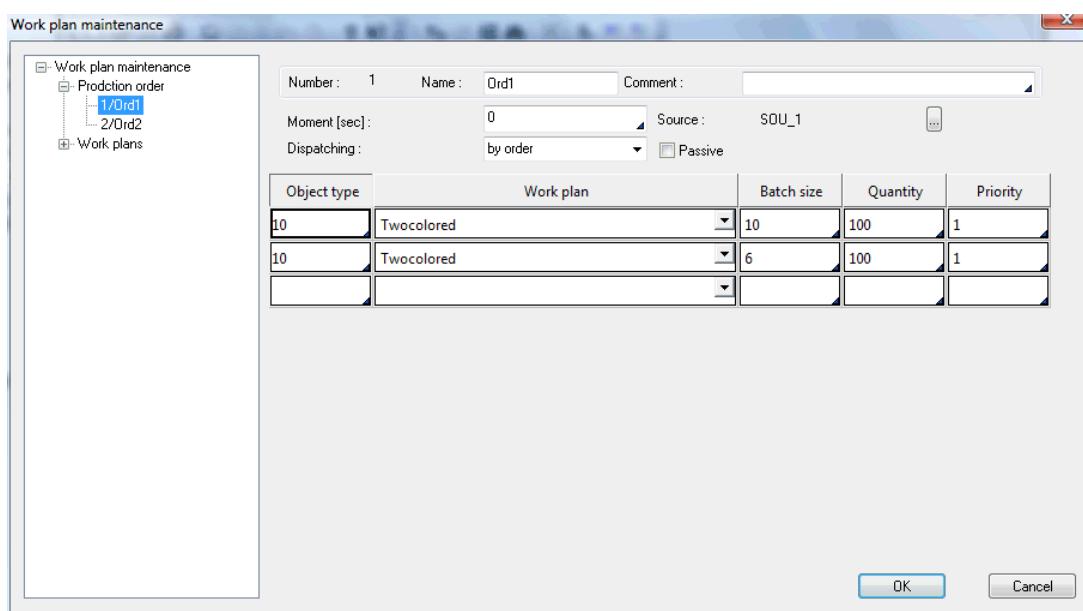


Figure 24.21: Order „two-colored“



24.7.5 Work Plan

Two work plans are put on: "two-colored" and "four-colored". The work plan "two-colored" begins with processing in quality control. From there a 50/50 probability exists, in order to branch out after red or blue.

From the operations red/blue in each case 50% exist after probability green and to 50% a probability for operation yellow as a successor.

These two operations have quality control in each case as successors (100%).

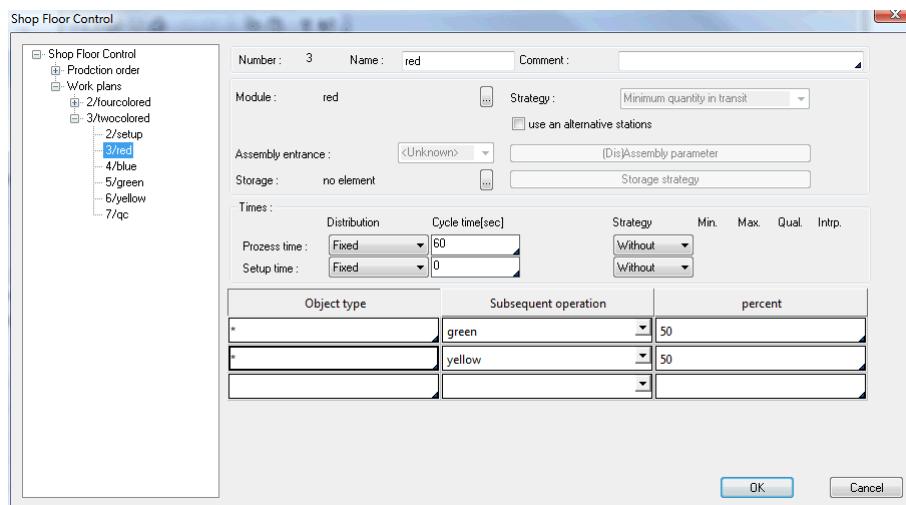


Figure 24.22:50% distribution after red painting

Quality control creates parts for 95% "::Ready::" and to 5% parts "::Reject::"

The work plan "four-colored" branches from red to green, to blue, to yellow in each case with 100% probability. Setup and quality control are parameterized as with two-colored.

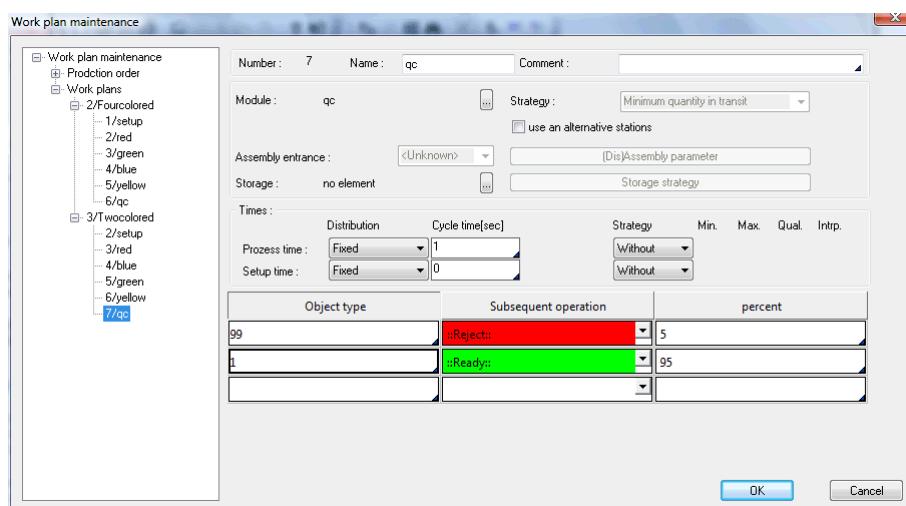


Figure 24.23:Distribution in the quality control



24.7.6 Statistic

The work plan statistics shows the turn-around times of the finished and rejected products separately according to their lot size and their work plan.

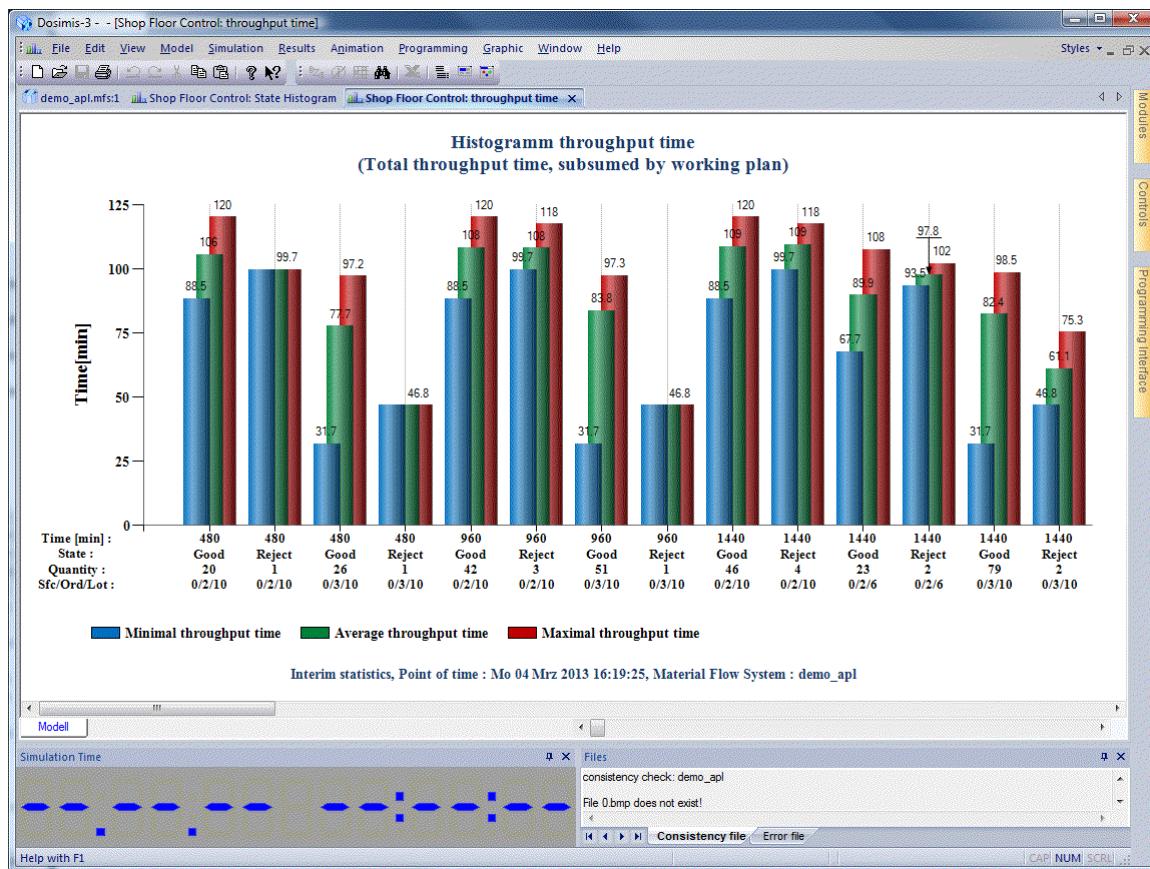


Figure 24.24: Work plan statistic





25 Graphic Comments

25.1 The Menu „Graphic“

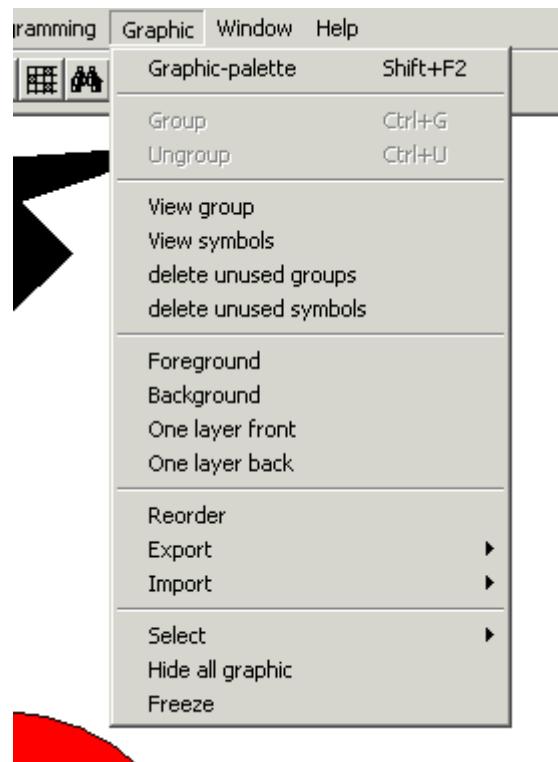


Figure 25.1: The menu „Graphic“

25.1.1 Graphic Palette

Use this instruction, in order to fade the graphic palette, with which you can insert graphic objects into your document.

Shortcuts

Keyboard: Shift-F2

This can be attained alternatively by means of the context menu of the toolbars.

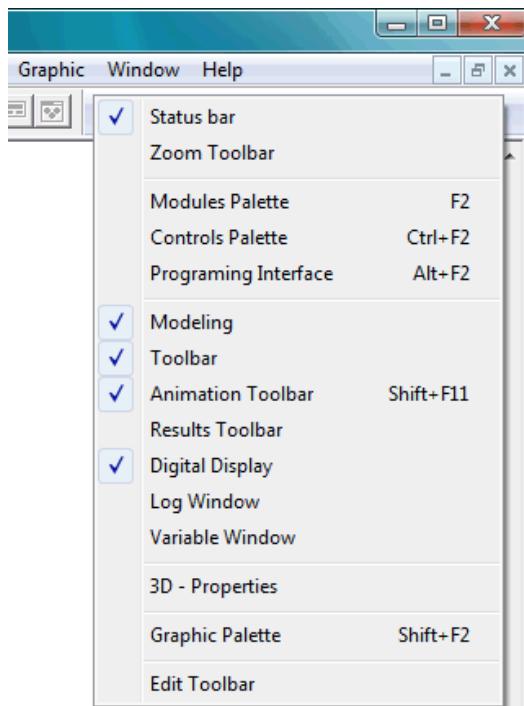


Figure 25.2: Activation of Graphic palette

25.1.2 Group

With this function you can join several selected graphic objects to a group. The objects themselves are then removed and replaced by a reference on the group (at least this is optically identical). The grouped objects behave henceforth as an individual graphic object.

If the selection contains references, these are resolved and their individual components into the new reference are inserted. There is no group of groups.

Shortcuts

Keyboard: CTRL+G

Select a number of graphic objects before and select then this command.

25.1.3 Ungroup

With this command you can separate before grouped objects. The reference on the group, as well as the group itself, is removed and replaced by the individual objects of the group. The objects now behave as before grouping again. If the original selection contained a group, then this is resolved also into its individual components.

Shortcuts

Keyboard: CTRL+U

Select first a group of objects, afterwards select you this command.



25.1.4 View Group

To get an overview of the available groups, one can call a group preview here. The [group-selection-dialog](#) will be opened.

One has the possibility of regarding groups, of deleting them and of assigning a name.

Caution: The function *delete* deletes also all occurrences of the group in the Dosimis-3-model.

25.1.5 View Symbols

Here one can regard the available symbols, assigning them names and delete them. To the deleting process applies: All elements, which are already represented with the symbol that should be deleted, appear after the deleting process again in their original representation.

25.1.6 Delete unused Groups

This function offers the possibility to delete groups which are not used in the Dosimis-3-model.

25.1.7 Delete unused Symbols

This function offers the possibility to delete symbols which are not used in the Dosimis-3-model.

25.1.8 Foreground

All selected objects are set into the foreground (Z-Order 1) and represented thus completely up most.

25.1.9 Background

All selected objects are set into the foreground (Z-Order MAX) and represented thus completely at the bottom.

25.1.10 One Layer front

With this command all selected objects are set forward one level. Therefore the Z-Order is decreased by one.

25.1.11 One Layer back

With this command all selected objects are set to the rear one level. Therefore the Z-Order is increased by one.

25.1.12 Reorder

Redraws all graphic objects. All objects are reordered according to their z-order. The z-order is a number between 1 and the number of graphic objects. The z-order is unique for every object.

25.1.13 Export/DXF- Export/DXG-Export

Stores all graphic objects in the AutoCAD-format (*.dxf) or Dosimis-3 graphic-format (*.dxg). Dosimis-3 specific information's are lost in DXF-format.



25.1.14 Import/DXF- Import/DXG-Import

Reads a file in the DXF format (AutoCAD) and DXG-format (Dosimis-3 specific).

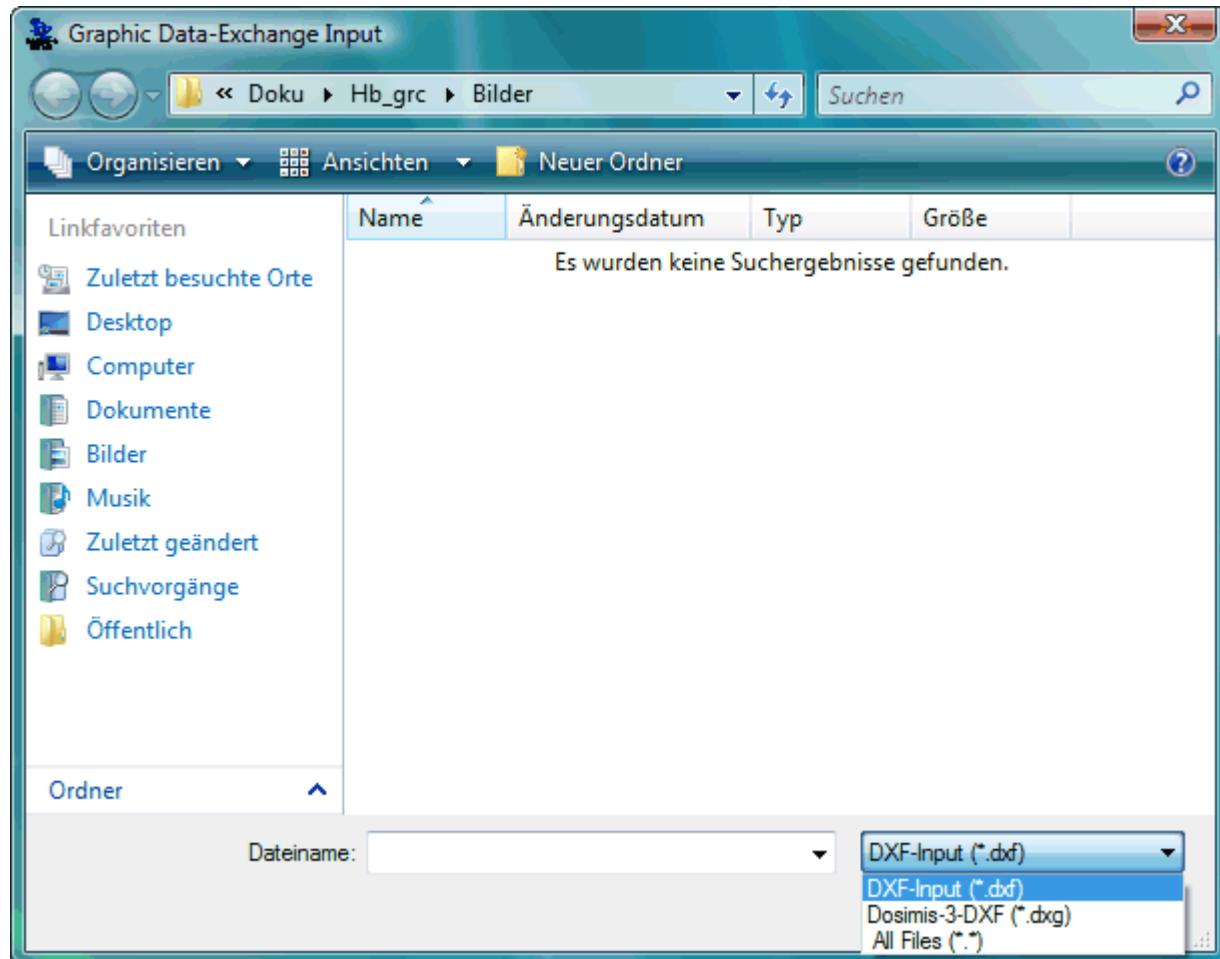


Figure 25.3: Graphic - Import

The new read graphic items are available as a reference on a group. The group possesses the name of the imported file and is inserted into the list of the groups. Therefore all items can be scaled immediately, as the reference is adapted to the layout.

Since Dosimis-3 does not administer references of references, these are resolved. See also *group*.

Dosimis-3 does not support each element according the DXF-specification. Therefore all drawings should be reduced to the elementary components. At least the level of detail should not be too high.

When inserting models it is to be noted that the graphic information must be added separately by the menu **Graphic/Import/DXF import**.

25.1.15 Select (all Graphic)

With this command all graphic elements can be selected. These can be used for further manipulations.



25.1.16 Hide all Graphic

This command hides all graphic elements (see also *Edit/Freeze*).

25.1.17 Freeze

With this command all graphic elements will be frozen (see also *edit/Freeze*).

25.2 Graphic Palette

Use the graphic palette, in order to insert and manipulate graphic objects into your document.



Figure 25.4: Graphic palette

The graphic palette offers the possibility of determining or subsequently of manipulating colors, line types, line weight, arrow at the line start as well as at the line end. If you select a graphic object in the model, then the palette reflects the adjustments (type of line, thick, heads of the arrow) for this object again. A modification of the adjustments in the palette affects all selected objects immediately. If no object is selected, then the adjustments apply to all objects, which are drawn from now on.

If several objects with different adjustments were selected, then the palette for these adjustments displays white (type of line, line weight) or light-gray (heads of the arrow) fields.

25.2.1 Drawing with the Graphic Palette

Use the graphic palette to insert graphic objects into your document. The following objects are available:

	Creates a <u>line</u>
	Creates a <u>rectangle</u>
	Creates a <u>circle</u>
	Creates a <u>polyline</u>
	Creates a closed <u>polygon</u>
	Creates a <u>reference</u>
	Creates a <u>text</u>
	Create a <u>bitmap</u>



25.2.2 Manipulating Graphic Objects with the Graphic Pallet

The following aids can be used, in order to manipulate objects:

25.2.2.1 Line Type Selection



Figure 25.5: selection of line type

This selection box lets you select different line types (broken, scored, point line etc.).

25.2.2.2 Line Thickness Selection



Figure 25.6: selection of line thickness

This selection box lets you select between different line thickness.

25.2.2.3 Set Arrow Heads



With these switches you can switch on and off heads of the arrow at the line start (arrow left) or at the line end (arrow on the right).

25.2.2.4 Line- and Border Color



This button calls a Windows standard dialog to determine the color for lines, heads of the arrows and borders.

25.2.2.5 Fill Color



This button calls a Windows standard dialog, to select the color for object fillings, which can apply to rectangles, circles and polygons.



25.3 Graphic Elements

25.3.1 Line

For creating a line you select the palette icon "line". Then you move the mouse to the starting point of your line and hold the left mouse button pressed. Now draw the line and then release the mouse button.

A selected line in the layout can be manipulated directly by dragging the track-points.

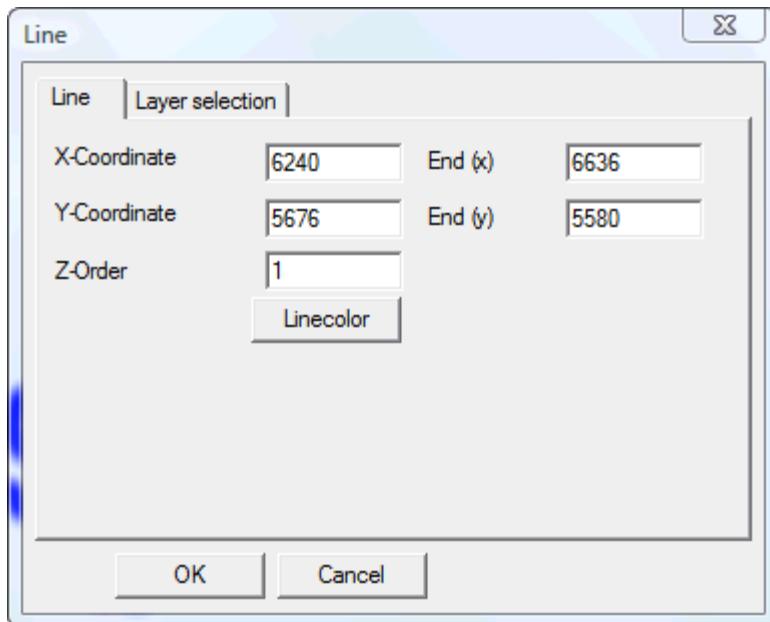


Figure 25.7: The line dialog

In this dialog you can change starting and ending point, the depth (Z-Order) and the color of a line selected before.



25.3.2 Rectangle

For creating a rectangle you select the palette icon „rectangle“. Then you move the mouse to the starting point of your rectangle and hold the left mouse button pressed. Now draw the rectangle and then release the mouse button.

Selected rectangles in the layout can be manipulated directly by dragging the track points.

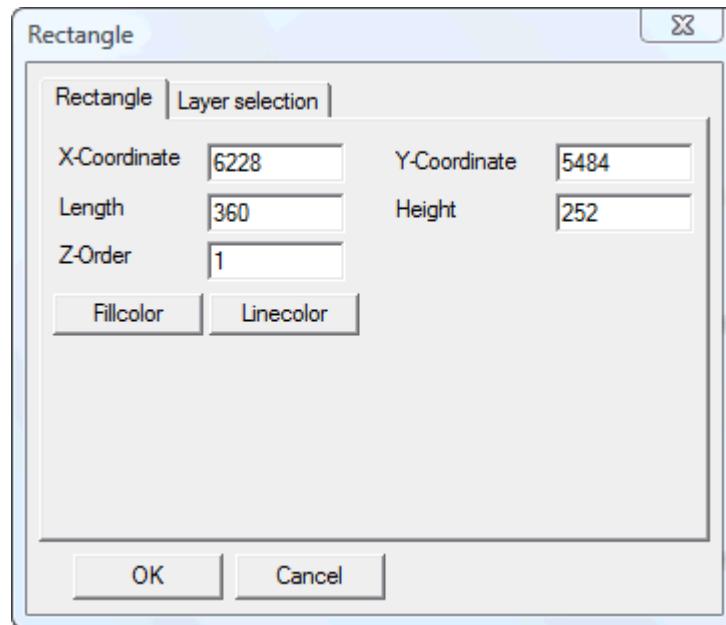


Figure 25.8: The rectangle dialog

In this dialog you can change the position of left top edge as well as width, height, depth (Z-Order), filling and line color of a rectangle selected before.



25.3.3 Circle

For creating a circle you select the palette icon „circle“. Then you move the mouse to the center of your circle and hold the left mouse button pressed. Now draw the circle and then release the mouse button.

Selected circles can be manipulated in the layout with the help of the track rectangle directly. This occurs however evenly in x and y-direction. The smaller of the two values determines the scaling factor.

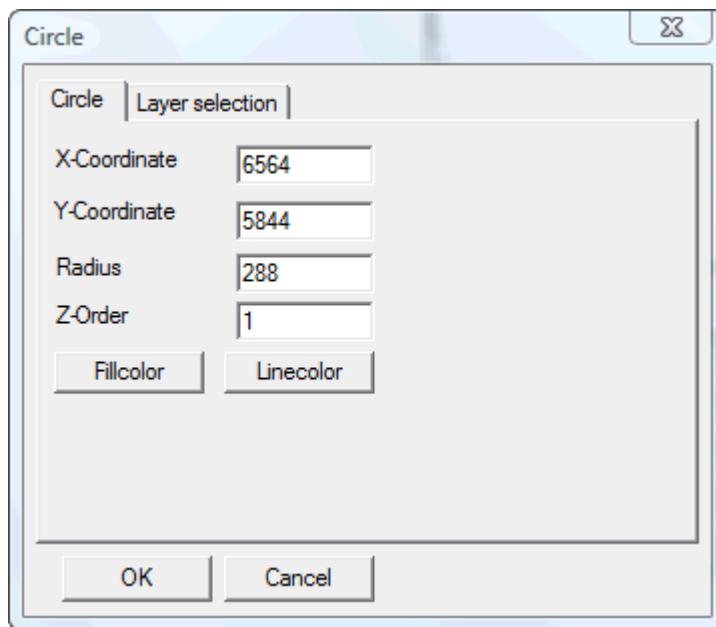


Figure 25.9: The circle dialog

In this dialog you can change the center, radius, depth (Z-Order), filling and line color of a circle selected before.



25.3.4 Polyline

For creating a polyline you select the palette icon „polyline“. Then you click the first point of the polyline. With each further mouse click you can extend the polyline by a line. Click on the right mouse button to end drawing the polyline.

A selected polyline in the layout can be manipulated directly by dragging the track-points.

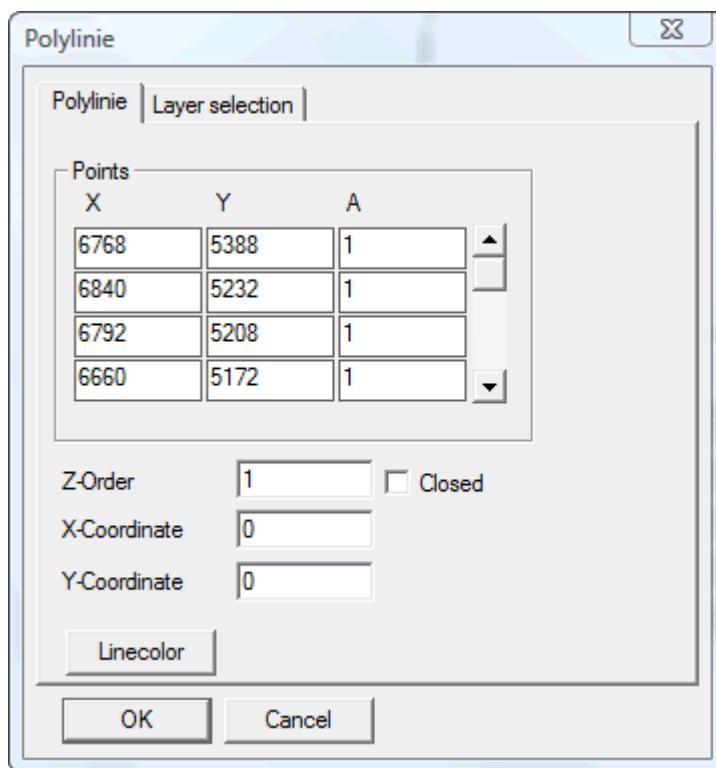


Figure 25.10: The polyline dialog

In this dialog you can carry out complete settings on a polyline selected before.

All ending points of the polyline are displayed above in the dialog.

Furthermore, the depth lets itself (Z-Order) as well as the starting coordinate of the polyline determine. Through the check box "Closed", it can be guaranteed that the polyline becomes a closed polygonal sequence. If necessary a closing line is added to the polyline for this. However, this comes about first after clicking "OK".

By "Line color" the color of the line set can be determined.

25.3.5 Polygon

For creating a polygon you select the palette button "Polygon" on Subsequently, you click on the first point of your polygons. With each further mouse-click can you now the polygon by a line extend. In order to terminate a drawing of the polygons, you click on the right mouse button. The polygonal traverse draft is now closed automatically.

A selected polygon in the layout can be manipulated directly by dragging the track-points.

The handling of the polygons takes place in the dialog of the Polyline



25.3.6 Reference

For creating a reference you select the palette icon „reference“ or group a number of graphic objects. A reference is to be understood as optical representation of a group. So you are able to represent a group several times. If you select the palette icon „reference“, you get the reference dialog. Here you carry out the settings of your choice and then click to the position of the reference to be generated.

Selected references can be manipulated in the layout with the help of the track rectangle directly. This occurs however evenly in x and y-direction. The smaller of the two values determines the scaling factor.

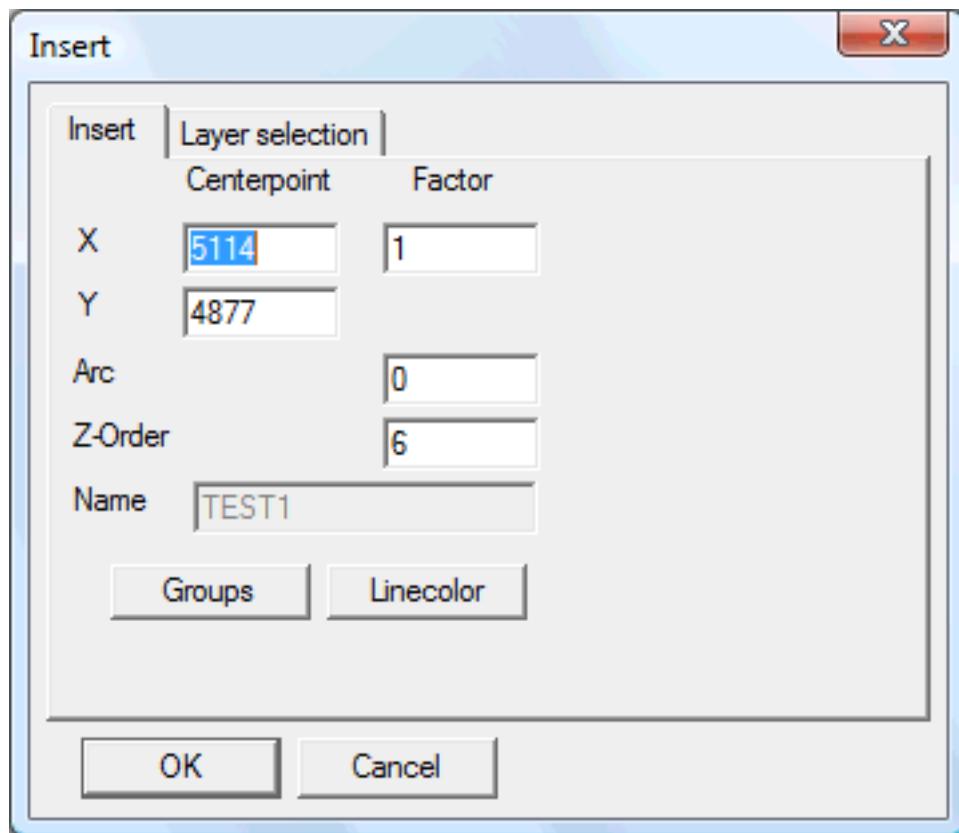


Figure 25.11: The reference dialog

In this dialog you can change the settings of a reference. You can change the coordinate of left top edge and adjust the stretching and compressing factors as well as angle of rotation and depth (Z-Order). By the function *Group* you get the group-selection-dialog where you can select the group to be referenced.



25.3.7 Text

For creating a text you select the palette icon „text“. Afterwards the text input dialog appears. Specify text, color and font here and click on "OK". You can now place the text with the left mouse button. Click at first on the left top edge of the text and hold the mouse button. Now you can drag a rectangle, which will enclose the text. Now release the mouse button

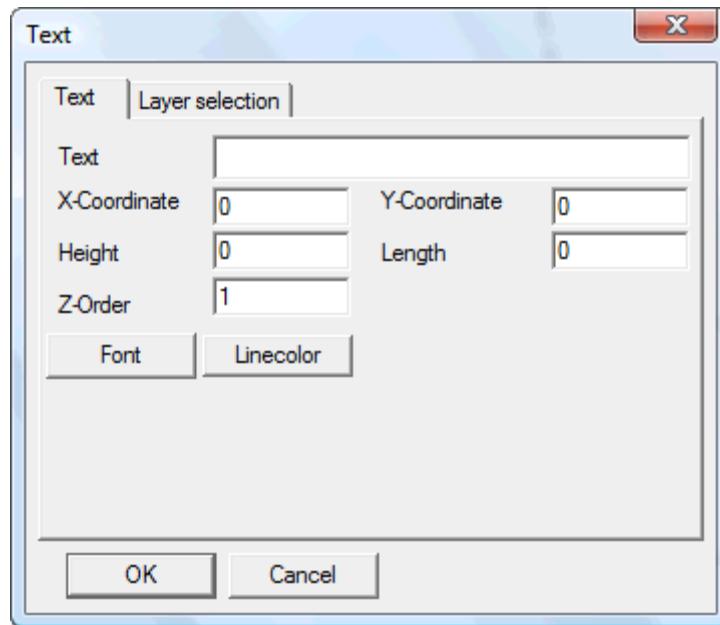


Figure 25.12: The text dialog

In this dialog you can edit the text selected before. You have the possibility to change the left top edge of the text. When height and length of the text are indicated, the enclosing rectangle is defined, from which the expansion of the text is calculated. When the value of length is zero, only the height determines the expansion, which results from that value and the assigned font. At least the depth (Z-Order) and the text itself can be changed. By the menu font you can edit font and color.



25.3.8 Bitmap

In order to place a bitmap into the layout, you select *bitmap* from the pallet. The bitmap-dialog appears subsequently. Indicate the name of the bitmap file, which is to be placed in the layout in the field **File**. Alternatively it can be searched by clicking on the button on the right of the input field. Click afterwards on OK. Now you can define the insert point of the graphic with the left mouse button. The bitmap is represented at this point in correct aspect ratio. You can adapt then position and size by the graphic operations.

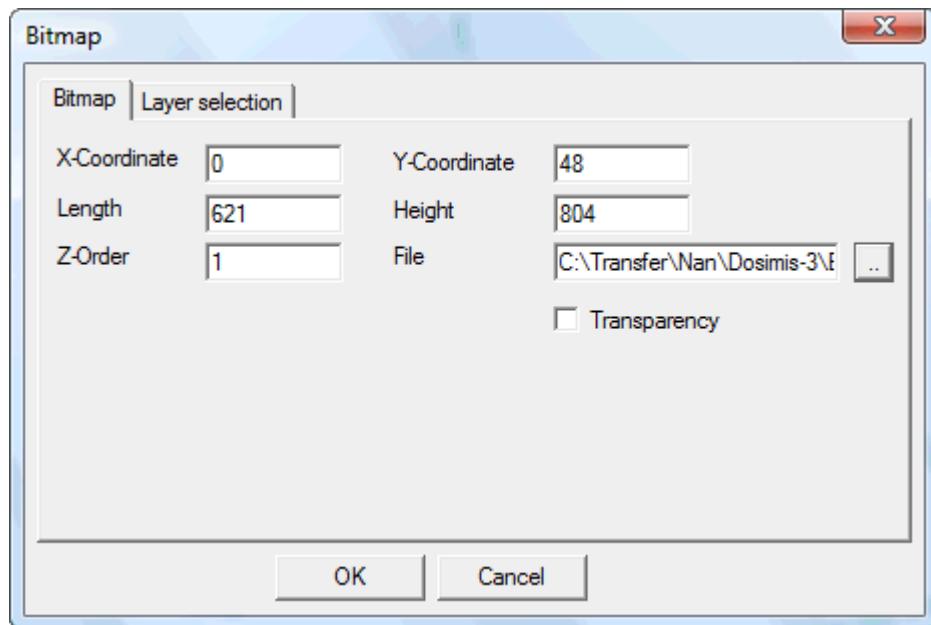


Figure 25.13: The Bitmap dialog

In this dialogue you can edit the parameters of a bitmap selected before. You have the possibility of changing the starting point. If height and width are indicated, an enclosing rectangle is defined, by which the expansion of the diagram is calculated. If the value of width is zero, only the height determines the expansion. This means that the size is calculated by the original size so that the aspect ratio agrees with that of the original file. This is naturally only possible when the referenced file exists. Otherwise the diagram is drawn as a square. Furthermore the depth (**Z-Order**) can be changed. By the switch **Transparency** it can be specified whether the bitmap covers elements, which lie behind it, or whether the elements are to shine through. It is to be noted that the transparent color is defined by the pixel in the upper left corner.



25.3.9 The Group Selection Dialog

In this dialog you can select the group of a reference or rename a group. Select the command button New Name for this.

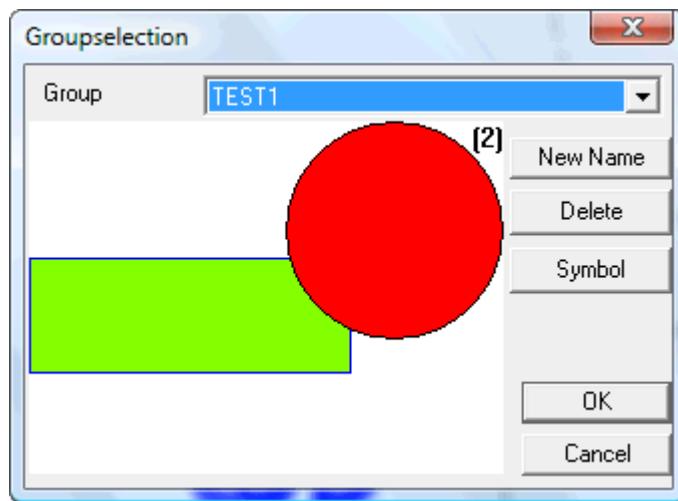


Figure 25.14: The group selection dialog

If this dialog is called from the graphic menu, the button **Delete** appears additional, as shown in the picture here. For more information read the sections View Group or View symbols.

If not the symbols are seen, additionally the possibility exists of transferring the selected group directly into the list of symbols.

On the top right in the representation the number of references is inserted, which refer to this group.

25.3.10 The Dialog „Name of Group“

In this dialog you can change the name of a group. For this purpose enter simply the new name in the input field "new name" and click to OK.

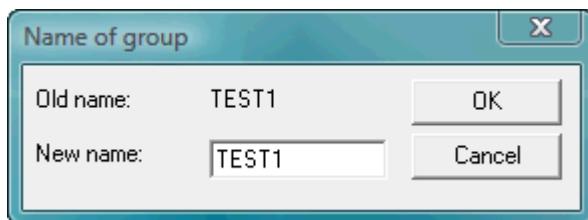


Figure 25.15: The dialog name group



25.4 The Symbols

With the help of graphic elements the possibility exists of using these as symbol for the Dosimis-3 elements. This is done via the definition of groups, which are then assigned to the individual elements. For this the symbols must separately be announced. According to the definition of a group a reference on this group is seen in the work area. In order to define this as symbol, the menu option **As symbol** is to be activated from the context menu of the reference.

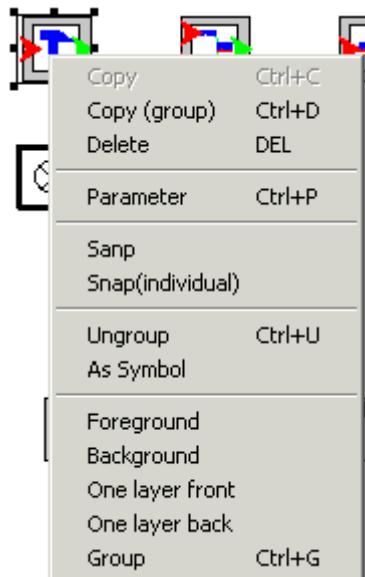


Figure 25.16: define symbol

Alternatively this can be done in the group selection dialog

According to the definition of a symbol this can be assigned to each element of a material flow system. This takes place again over a context menu, this time that of the element, from which the menu option **Graphic** must be selected.

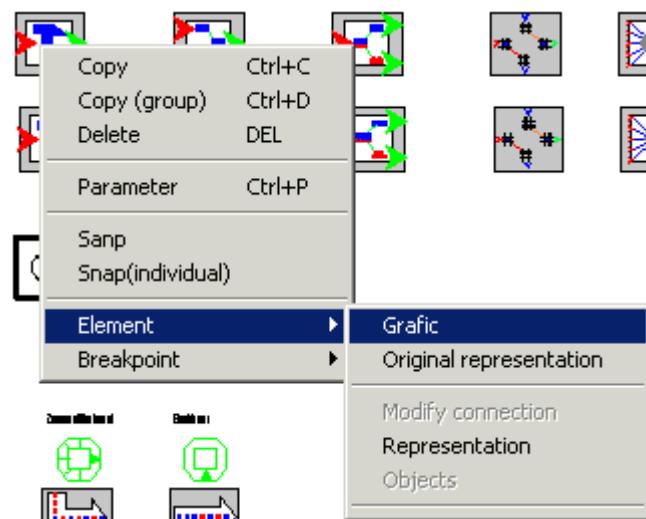


Figure 25.17: Assign symbol



Thus one arrives into the group selection dialog, from which the appropriate symbol can be selected.

Once assigned symbols can also be changed. If the assignment is to be taken back again, then the menu **original representation** is to be selected from the context menu.

The symbols are administered in the file **Ds3edit.dxg** in the work directory. This means that all models of the work directory can access this file. The allocation from symbols to elements is located in the file **modelname.pic**.

Available symbol files can be imported over the menu **Graphic/Import/import DXF**. Since these are sorted however into the list of the groups, these are to be made subsequently with the available well known mechanisms as symbols.



26 3D Visualization

26.1 Introduction

DOSIMIS-3 provides an environment for easily building three dimensional simulation models for analyzing and revising as well as animating the simulation processes.

26.1.1 Installation

In order to use DOSIMIS-3-3D, it is required that before the installation of DOSIMIS-3, Microsoft DirectX 9 (June 2010 or higher) is already installed in the computer. The DOSIMIS-3 installation CD-ROM offers the required version of DirectX. If the DirectX installation being ignored during the installation, it is required to be installed afterwards and restart the DOSIMIS-3 Setup. Otherwise the 3D visualization won't be available

26.1.2 System Requirement

- Installation of Microsoft DirectX 9.0 (or higher version)
- Direct-3D compatible graphic adapter.
- Standard mouse with 2 buttons and mouse wheel

Attention! Most mouse producers provide software which can change the function of the mouse wheel. (e.g. with a scroll function or a pop-up menu). These programs will forcefully hamper your work when using the 3-D interface and should be deactivated.

26.2 Application

After the correct installation, click View/3D-View to create a 3D-view of the chosen model.

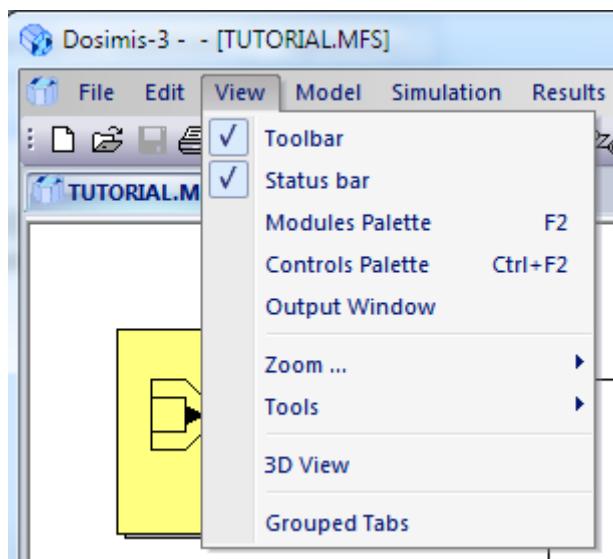


Figure 26.19: open the 3D-View

Click the 3D View and at the same time press the SHIFT button, the previous 3D-Model will be replaced.



26.2.1 Multiple View

Click the menu View/3-D View again to open another 3D window. User can navigate and perform all the operations (choose, move etc.) on respective model in each window and observe all the windows simultaneously.

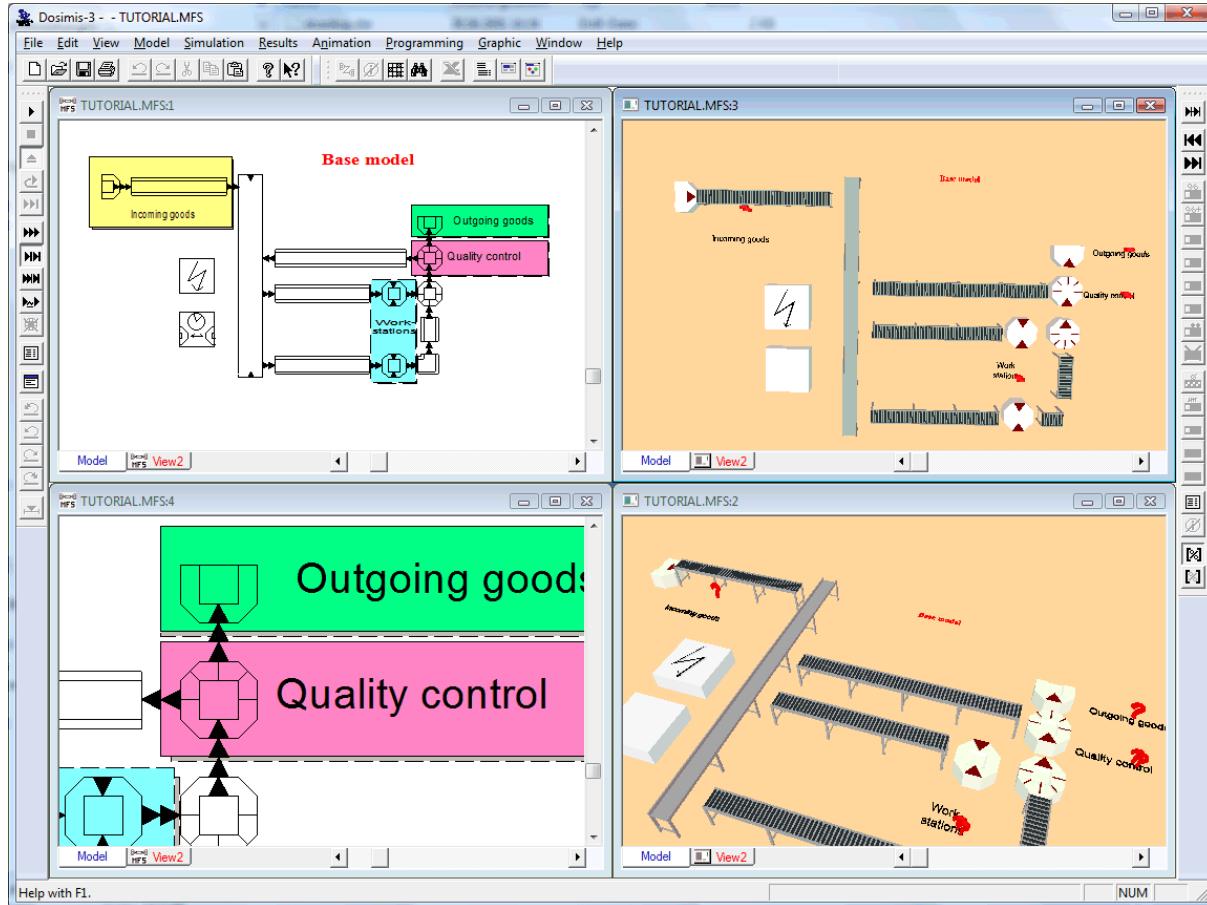


Figure 26.20:Multiple View



26.3 Navigation

The navigation with the mouse is easy and intuitive. User would get used to it in several minutes.

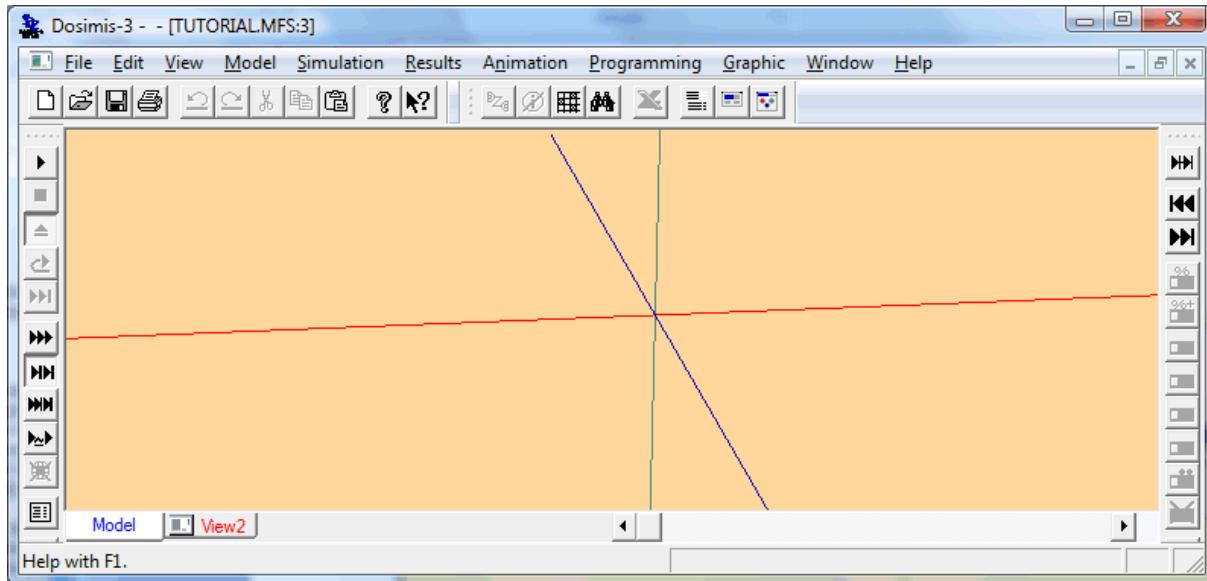


Figure 26.21: the 3D-View coordinate system

In order to better orientate in the 3D View, each direction of the axis is marked with a different color. The X direction (from left to right) is colored red, the Y direction (from front to backward) is blue, and the Z direction is green.

26.3.1 Rotation

Press the middle button and move the mouse. The camera follows the mouse, when it moves around the target object.

26.3.2 Forward/Backwards

By using the mouse wheel, the user can move towards the object or backwards from the object. User can find a fine grid by clicking the scroll wheel and press the SHIFT button.

26.3.3 Sidewise Movement

Press simultaneously the middle button and the SHIFT button, when moving the mouse. The camera move sidewise following the mouse movement.



26.4 Operations in the 3D-View

Click on the blank area in the 3D View with right button of the mouse, it will show a pop-up menu. This menu offers the operations that can be used by the whole model.

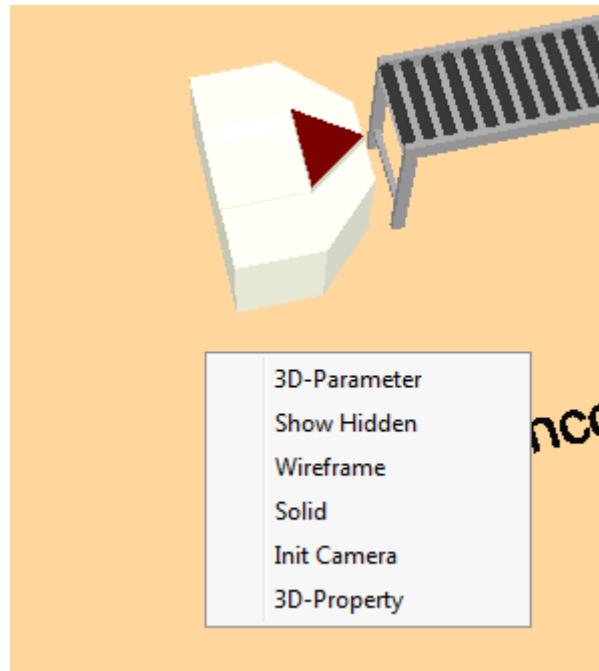


Figure 26.22: pop-up menu of the 3D-View

3D-Parameter opens the parameter dialogue to set basic configuration like the color for the ceiling and ground. Also, the configuration for snap shot is here available (see section Snap of the 3D-objects)

Blend the objects shows all objects. Use this option again to hide the objects.

By using the **Grid**, all the objects will be displayed as a grid. This decreases considerably the time for calculating the screen layout.

Select **Area** to show all the objects in their original representation.

26.5 Operation on the 3D-Objects

User can use the left mouse button to choose the 3D-objects e.g. simulation module. User can identify the object by a frame and colored “spots” around it, the so-called Handles. The three kinds of colors show the different direction of the coordinate system.

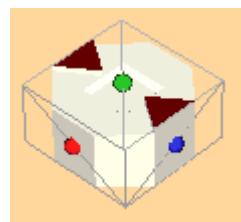


Figure 26.23: Handles of the chosen objects



User can choose several objects by clicking them and pressing the Shift button simultaneously. The selection can be cancelled by clicking the blank area.

26.5.1 Move

You can move an object by selecting the object with the mouse (left button) and dragging the object around the work area with the pressed left mouse button. The vertical movement is possible when the user press the Shift button at the same time. The position of the objects can be set at the property window.

26.5.2 Scaling

With the help of “Handles” (6 colored spots around the selected object), the sizes of the objects can be changed. Click one handle with the left mouse button and move it into desired direction without loose the left button.

The objects change the size differently: static objects will “stretched” while the generic objects - like the roller conveyor – change their size according to the number of tripods and rollers.

The sizes and scales of the objects can also be set the property window.

26.5.3 Rotate

Object rotation works the same way as the scaling per handles, only with the difference that it needs to hold the Ctrl-button. Rotation is only possible in the horizontal level (X/Z-level). The rotation degree can be fixed in the property window.

26.5.4 Parameterization

Double click on the objects to open its parameter dialogue.

26.6 3D Toolbar

The 3D toolbar is a valuable aid when working with 3D objects. The toolbar will allow direct access to some of the basic utility functions in the 3d world.

To activate the 3D toolbar, chose “3D toolbar” from the “View/Toolbars” menu.



Figure 26.24: 3D-Toolbar

26.6.1 Functions of the 3D Toolbar

The toolbar offers the following functions (left to right order)

- Open the 3d setup dialog
- Activation of the grid mode of selected objects
- Activation of the solid mode of selected objects (deactivating grid mode)
- Render selected objects invisible
- Show invisible objects
- Convert to conveyor
- Convert to curve



- Convert to corner converter
- Replace object with X file
- Snap to grid
- Snap to objects
- Snap to planes

Animation slider: Time-shift a running animation

26.7 The Window “object property”

DOSIMIS-3 provides the possibility to control 3D objects through the window “object property”. Show the property window by selection the entry properties from the context menu of an object.

Properties	Value
Coordinates	
X	-329.687500
Y	1.250000
Z	19.531250
Size	
Length	18.750000
Width	1.562500
Height	2.500000
Rotation	
Rotate	179.905136
Display	
Visible	True
Surface	True
Snaps	False
Ani.line	False
Colors	
Stand	<input type="color" value="#a0a0a0"/>
Trough	<input type="color" value="#a0a0a0"/>
Conveyor	<input type="color" value="#333333"/> 333333

Figure 26.25: object property

The following standard property can be set:

- Positioning (attention!: Y-Axis vertical)
- scaling factors for all the extension directions
- Display setting
 - Visible: showing or hiding each object
 - Surface: choose between whole area display and gird net.
 - Snaps: the snap points can be indicated by black crosses.
 - Ani Line: Animation lines can be displayed as objects as a green lines. Objects move along this line during animation.
 - Rotation: the rotation in the plane (also around the Y-axis).



Further properties of special objects:

- Conveyor: color of the tripods, tubs and conveyors.
- Width and length of the tubs. These values are dependent on the object size.
- Curves: radius and angel
- Corner converter: direction left/right

26.8 Graphics Exchange

Click the right button to open the context menu.

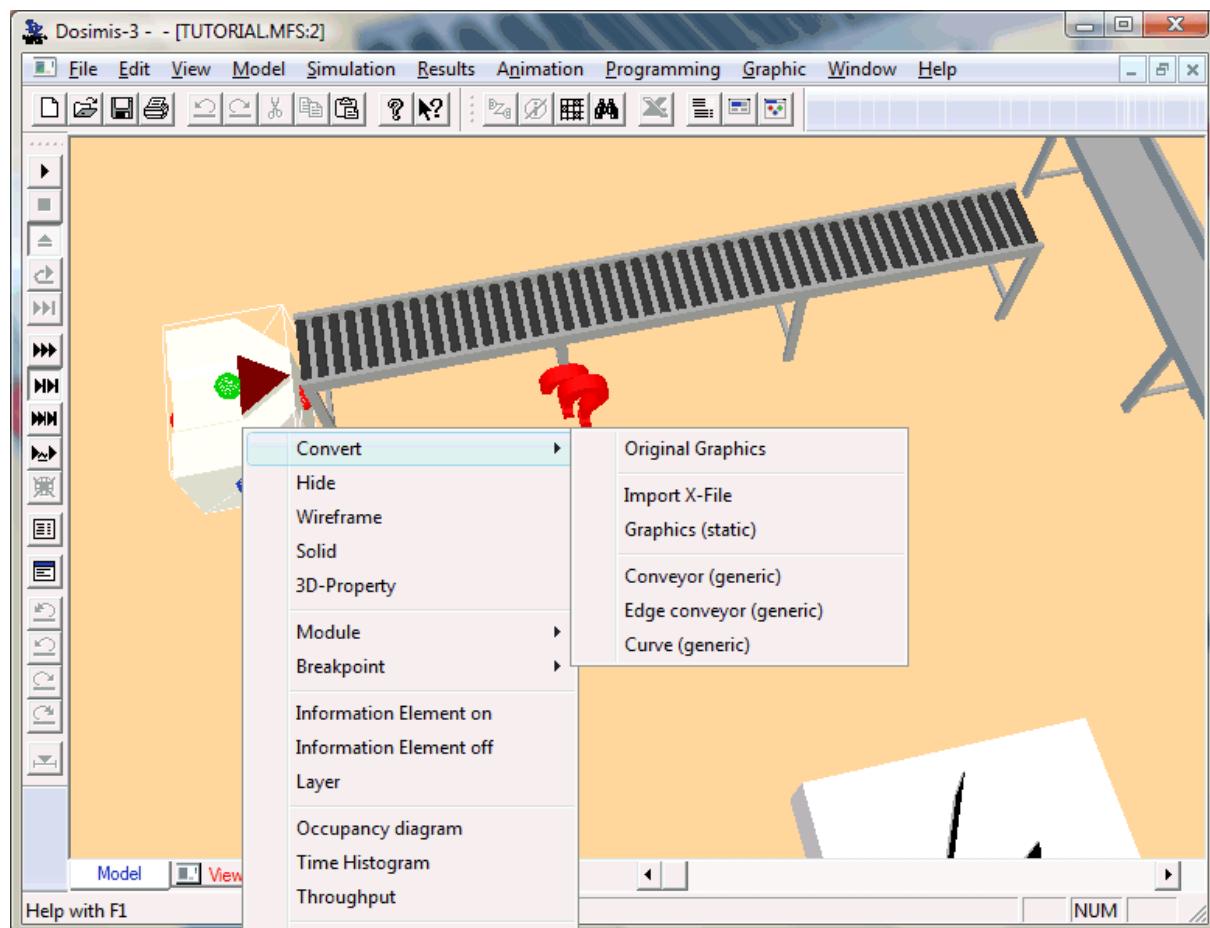


Figure 26.26: pop-up menu of the 3D-modules

User can use the pop-up menu of the objects to show the object as **grid** or **solid**. Moreover, every single object can be **blend out**. These options are provided also in the property window.

26.8.1 Convert the Representation

Under the submenu **Convert**, the object representation can be totally changed. Using the **Original Graphic**, user can change the object representation to the original 3D-model graphics

Use the pop-up menu **X-File import** to read the data in Microsoft DirectX format. The chosen module will be exchanged with this data. For the purpose of demonstration, the installation



CD includes a small collection of X-Files which is installed in the directory **X-File** of the installation directory.

Use **Graphic (static)** to convert the representation to a static graphic and can be freely switched.

Use **Conveyor (generic)** to replace the graphic with a generic object (currently as the representation of a roller conveyor).

Use **Corner converter (generic)** to replace the graphic with a corner converter. The Corner converter will be generated based on the parameter in the property window. So it can be selected for example whether the transfer should take place to the left or to the right.

Use the **Curve (generic)** to replace the graphic with a curve. The curve will be generated based on the parameter in the property window. Normally a right curve is produced (passage in the clockwise direction). If a link curve is needed, then afterwards a negative angle is to be entered in the object properties.

26.9 Duality of the 2D and 3D Display

By the first invocation of the menu **View/3D-View**, the 3D-model will be generated. All DOSIMIS-3 modules will be converted to the 3D view.

The other modules in the 2D-view will be generated in 3D-view, however, they are remaining "hidden" and will be displayed in the 3D window under the pop-up menu.

For the new objects in the 3D-view for the habitual purpose can be generated from the module pallet. These will be placed in 2D-view.

The subsequent movement or rotation of 2D or 3D object will have no influence on the object in the respective another display manner.



26.10 Snap of the 3D-Objects

In order to place the 3D-object in a better way, the 3D interface provides a grid and grid points, the so called “Snaps”. The object will be placed along the grid lines, when they are moved – like they are magnetic to them. When the object is moved near the snap, the object will automatically jump to the appropriate position.

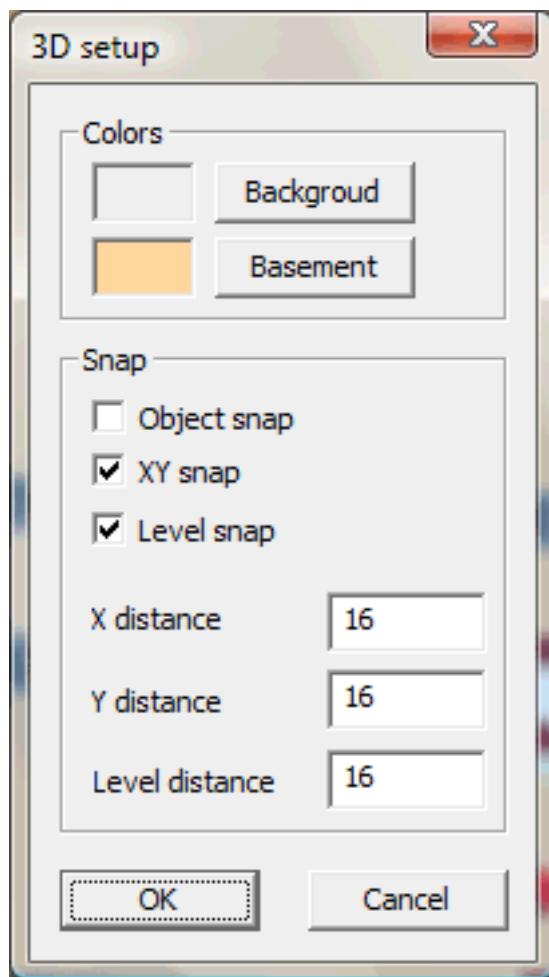


Figure 26.27: 3D Parameter

Snap points are for example at the end of the roller or band conveyors. Therefore, it is possible to place several conveyors directly one after another.

In order to set the grid proportion, user can open the “3D Parameter” dialog box and click the 3D widow with the right bottom of the mouse. At this time, no object should be selected. Please just choose “3D Parameter”.

Alternatively use the snap symbols within the 3d toolbar.

In the dialog box, the Snap proportion can be changed as follows:

- Check box “Object-Snap”: It can be chosen here whether the snap points of the objects (e.g. roller conveyors) should be active or not. Snap points can also be show or hidden through the property window.
- Checkbox “XY-Snap”: It can be chosen here whether the grid for the sidewise movement (grid on the horizontal level) should be active or not.



- Check box „Level-Snap“: It can be chosen here whether a snap proportion of the vertical level of objects should be active or not.
- Input field X-/Y-/ level distance: user can choose the relevant width of the snap grid. The input is in meter. A standard 3D display of a module is 1.5 m.

Choose “OK” to close the dialog box. Afterwards the snap-proportion which is just be parameterized will be activated.

26.11 Animation

A time consistent animation for the 3D model can be displayed. In order to do this, user need to firstly as usual parameterize the DOSIMIS-3 model. It is quite helpful to animate and validate the model in 2D-view.

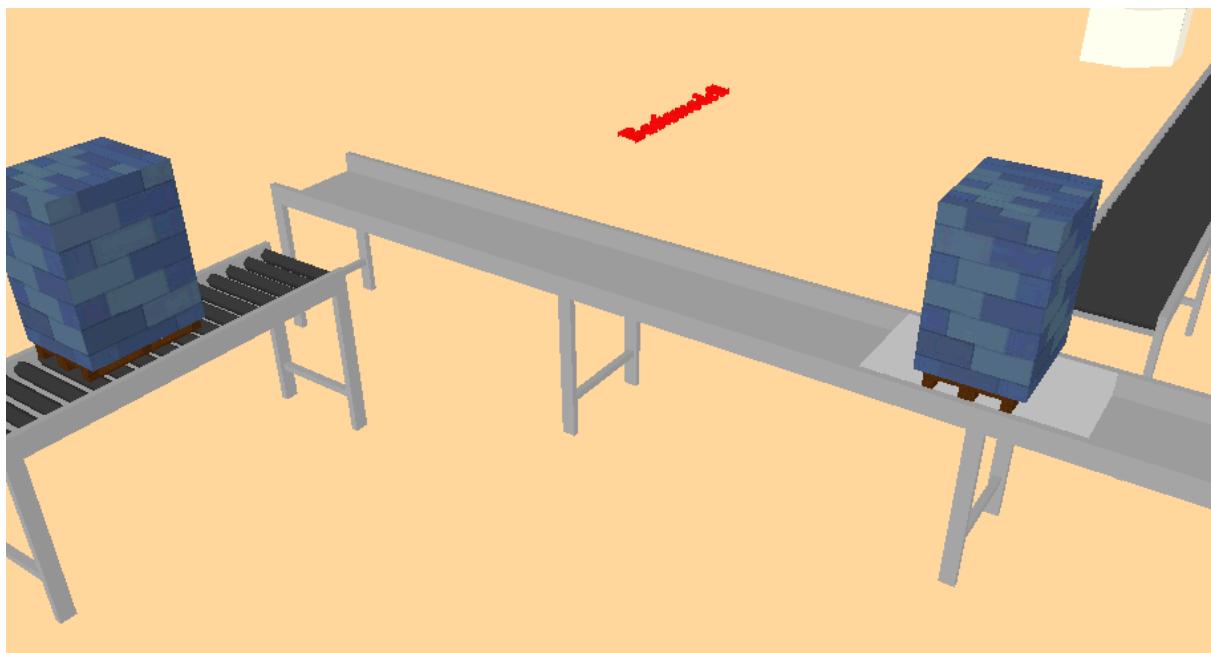


Figure 26.28: 3D Animation

Open the 3D-View and click the button “start Animation” on the animation toolbar, then the 3D-Animation will be started.

An animation calculation will be started depend on the size of the model, number of the animated object and the animation time. This can be interrupted by click the key ESC. However, the animation will not complete displayed and run in total time.

The animation will start after the calculation. During the animation run, user can freely navigate through the model and move or parameterize the module.



26.11.1 Navigation during the Animation

During the animation, the navigation can be controlled by the following keys:

CTRL+ Screen-up	Increase speed
CTRL+ Screen-down	Decrease speed
Space	Animation pause/resume
CTRL+ End	Change between forward/backward animation

The animation speed can be decreased to negative number – then it will move backwardly.

26.12 Animation Parameter

26.12.1 Object type Assignment

DOSIMIS-3 provides the possibility to assign the object types with its own graphics. Open the “Animation/3D Parameter”. On the left side of the dialogue box is a list of the 3D graphics (specification of the relevant X-file) and the assignment to the object types. For the new models, the assignment for all the DOSIMIS-3 object types has only one option “box.x”.

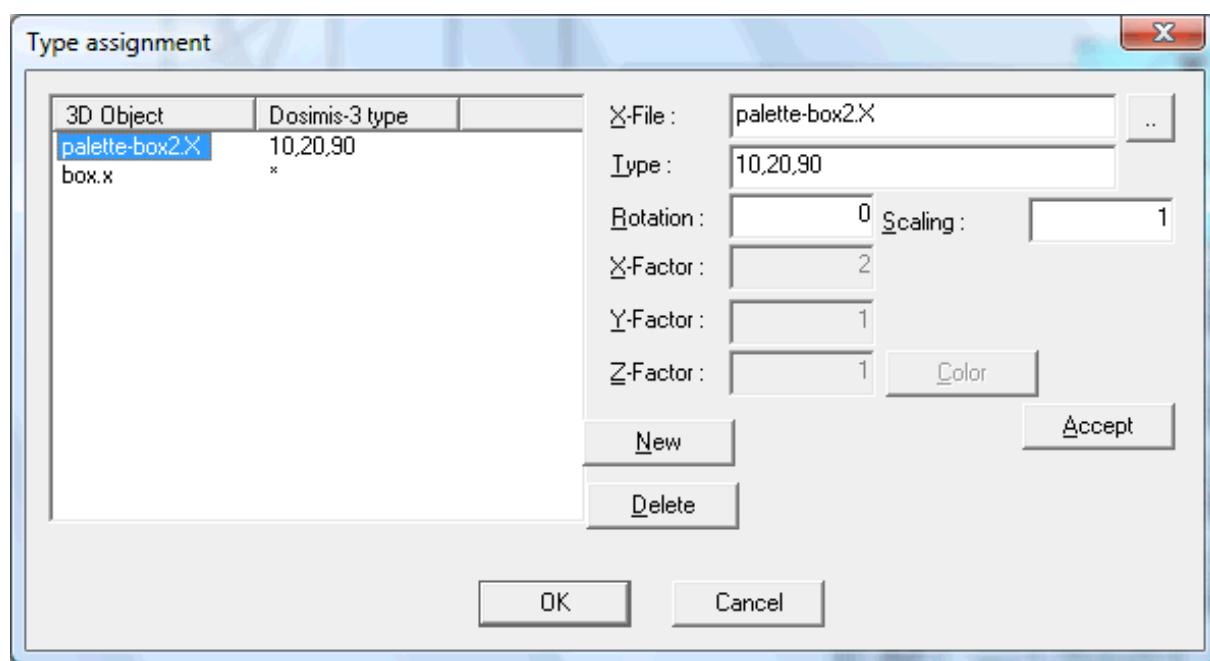


Figure 26.29: 3D Animation parameter

For further attaching other graphics, choose “New”; The list will extend with a new entry. On the right side of the dialogue box, a X-File can be chosen and assigned to a DOSIMIS-3 object type. Permissible assignments are type number, divided by comma, or the input from the a range (from - to).

The other influences on the graphic display of the objects are the input of the rotation (angle input in grade) or the input of the scaling factor.

User can change the assignment at any time. Choose the related input and change the configuration and then choose “OK”

Use the “Remove” button to remove the chosen input.



26.12.2 Animation Speed

Since the 3D Animation runs chronologically, the animation speed can be set under the conventional parameter dialogue box “Animation/Parameter”. There are related time factor can be chosen in the animation manner under “time wise animation”. Additionally user can change the speed during the animation with the key combination CTRL+ Screen-up and CTRL+ Screen-down.

26.13 Camera

DOSIMIS-3 allows you to store camera positions and to restore them at a later timer. So it is possible to keep good view positions, e.g. for using them at presentations. Camera positions in DOSIMIS-3 are shortly „cameras“.

Additionally, camera positions can be set to a certain animation time. So it is possible to define camera paths.

26.13.1 Active Camera

In the top left corner of the 3D view the name of the currently active camera is shown. Usually this will be the so called “Perspective” cam.

26.13.2 Properties of the Camera

To change camera properties, open the 3D properties window by using the view menu. In this window all properties of a selected 3D item are shown. Now click in a free area within the 3D window. The formerly selected modules get unselected and the properties of the currently active camera can be edited in the 3D properties window. Next to the common properties for 3D objects the following settings are shown:

- Name: Name of the camera
- Animation: If set to true, the animation camera will move to this position
- Time: Animation time when this camera position shall be reached

26.13.3 Adding Cameras

By opening the context menu within the 3D view, chose the pop-up menu „Cameras“ and click „New“. Now a new camera gets placed on your current view position. When editing the 3D model, the new camera gets activated immediately and can be moved now. When creating a camera during animation, the new camera get set up with fitting animation settings. Afterwards, the animation cam get reactivated (see “Camera animation”).

26.13.4 Choosing a Camera

All cameras can be selected within the 3D view context menu under „Camera“. The active camera is highlighted by a check mark. Chose another cam by clicking the respective menu entry.

26.13.5 Deleting a Camera

First chose the camera to delete in the Camera context menu of the 3D View. Now open the context menu again and chose “Delete current”.



26.13.6 Initialize a Camera

After choosing “Init Camera”, the currently active camera gets placed above the model. This can be helpful if you lost orientation in the 3D world.

26.14 Camera Animation

In DOSIMIS-3, a camera path can be defined for presentation purposes or for recording videos. While animation, the animation camera moves to certain previously defined camera positions (“cameras”) at given animation times.

26.14.1 Creating a Camera Animation

The easiest way to define a camera path is to use the animation mode. Prepare a model in such a way that it can be animated. Check if your model gets animated well by simply starting the animation and moving the camera around as usual. When you are satisfied with the results you can start recording camera positions.

To do so, start the animation again. To record a camera position, place the active camera (named “Animation”) to an interesting point and perspective. At a fitting time within the animation, create a new camera. To do so, you can simply use the shortcut ALT-C, or use the context menu “Camera/New”.

During the animation, cameras get created in a different way. The newly created camera gets activated for animation and the time property is set to the current animation time. Afterwards, the animation camera is reactivated to create further camera positions.

You should define at least 3 cameras before the animation ends. While doing so, you can use the pause button, or use fast-forward or rewind functions as you like.

After having created all camera positions, stop the animation completely by pressing the stop button in the animation toolbar twice. Then re-start the animation.

Now the animation camera will move from one camera position to the next continuously in a soft motion. The animation cam cannot be moved any more by hand.

As described in the camera section, all animation cameras can be edited or deleted using the 3D properties window and the camera context menu.



26.15 Overview of the Key- and Mouse Action in 3D View

Mouse	Keyboard	Action
Middle button		Camera fly around
Middle button	Shift key	Sidewise camera movement
Scroll wheel		Forward/backward (zoom) movement
Scroll wheel	Shift key	Forward/backward (zoom) movement (small steps)
Move object(left button)		Move object horizontally
Move object(left button)	Shift key	Move object vertically
Move object Handle		Scale object
Move object Handle	CTRL key	Rotate object
	CTRL+ Screen-up	Animation speed up
	CTRL+ Screen-down	Animation slow down
	Space	Animation pause/play
	CTRL+ End	Change between forward/backward animation
	ALT-C	Create new camera

26.16 Restrictions

The animation of “rare” module is not converted yet. The modules are conveying circle and paired shuttle.

Modules with several bases will be displayed erectly and will be in the direction of the first orientation change.





27 Excel-Interface

27.1 Introduction

DOSIMIS-3 has an interface to exchange data with Excel files. Thus it becomes possible to parameterize DOSIMIS-3 modules externally and in tabular form. Consequently one is also able to calculate module parameters by Excel formulas. The format of the tables resembles an interpreter language, whereby it is not only possible to exchange data between Excel and DOSIMIS-3 but also to start automated simulation runs and modify parameters.

The format of the tables resembles an interpreter language, whereby it is not only possible to exchange data between Excel and Dosimis-3 but also to automate constantly returning operational sequences.

This documentation does not concern the transmission of the data of a result diagram to Excel. This functionality is described in the user manual of the built-in functions in the chapter of results.

27.2 Working Method

For the data exchange with Excel, DOSIMIS-3 is linked with a Excel table. If an Excel application has been started, the table page 'is loaded' into the open application, so that changes can still be made there during work with DOSIMIS-3.

27.3 Structure of Table

So that DOSIMIS-3 can exchange data with an Excel application, a table must be available in a specific format. Create a new Excel document and edit the first table sheet. A keyword must be located in the first cell of every line. A determined number of parameters(for each keyword) follow the keyword, a parameter per cell. The processing of a table ends with a line with emptier first cell or with the keyword "end".

27.4 Connecting with Excel

An Excel sheet must be stored once on the hard disk so that DOSIMIS-3 can access them. Nevertheless, it is possible for DOSIMIS-3 to access current table data even if the *current* table was not yet stored. DOSIMIS-3 accesses the table directly by a current Excel application and does *not* read the data from the table file. However, a file name is urgently necessary for data exchange.

In order to connect DOSIMIS-3 with the table, at first one loads or generates a DOSIMIS-3 simulation and an appropriate table. Now one selects the function **Excel-Data-Transfer** in DOSIMIS-3 in the menu *Model*. A submenu appears, from which one selects **Open**. Now one obtains a file selection dialog with which one selects the table to be connected. If no file with the entered name exists, the file will be created new in the project directory.

If an Excel application is opened, the table is loaded there and displayed. Otherwise an Excel application is started automatically.

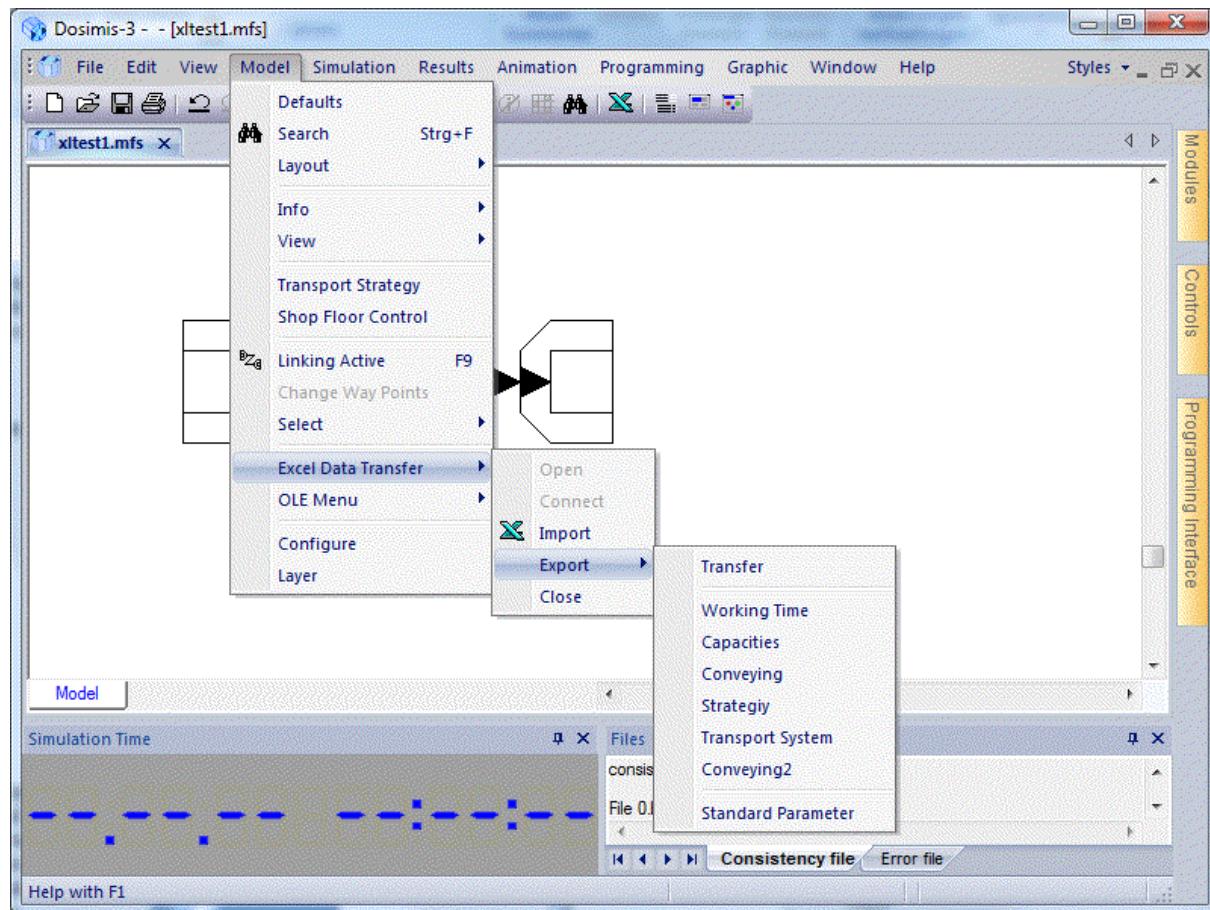


Figure 27.1: Menu Data-Transfer after opening a table

Note: If no file in the project directory exists, this can be created with the help of the context menu of the file open dialog. Click with the right mouse button in the work area and select *New/Microsoft Excel Spreadsheet*.

If Excel is not yet started, this can be done with the help of the context menu of an excel file (*Open*).

Beneath the output of all parameters it is possible, to export selected parameters.

- **Working time:** The working time of the lists will be exported. As well as the default process times of (un-)loading stations and the setup times are added.
- **Capacity:** The number of places in a module is exported.
- **Transports:** The length and the speed of the modules are exported.
- **Strategy:** Right of way and distribution strategy.
- **Transport system:** The destination code and the priority of modules from a transport system (unloading, loading station, (multiple-) work station).
- **Transports2:** Additional information as position in a shuttle or conveying circuit.

27.5 Transfer of Data

Click in the menu *Model/Excel-Data-Transfer* to the menu item **run extern**. The Excel-Script is now executed.



Shortcut

Symbol:	
Toolbar:	<i>Modeling</i>
Keyboard:	no representation

27.6 Automatic Creating of a Table

An Excel table is created automatically by the function **Export** in the menu **Excel-Data-Transfer**. The table is structured that during execution of the script all modules are filled with the parameters, which were adjusted at exporting.

If selected parameters should be exported, an possibility exists, to export only the parameters of working time, capacity or transport. For example with **work time** only the parameters of a workstation, assembly, disassembly, ... are exported, which have to do with the working times of the objects. These are also assembly lists. If a work area is selected, the tasks are exported.

27.7 Closing of the Data Transfer

The function **Close** ends the data exchange with Excel. The table is closed in current application and Excel, does it run as background tasks, is now ended.

27.8 Keywords and Parameters

The data transfer module detects the following keywords in the first cell of every line and that relevant parameter in the following cells. Uppercase/lowercase is ignored by the transfer module.

Used parameter types:

int	defines an integer
int_var	Name of an integer-Variable or integer
float	a real number
float_var	Name of a Float-Variable or real number
str	a character string (Text)

Is no type indicated, it concerns a fixed text. This is highlighted in addition by a **bold type**.

27.8.1 Command Set

Set Set module parameter.

27.8.1.1 Material Flow

set randomnumbers

Change all random numbers of the model



27.8.1.2 Module Parameter

The second key fixes the parameter, which should be changed. There are several different keywords valid.

set standardparameter str name

Set the standard parameter of the module *name*.

The export takes place column by column with the following parameters:

Speed Object length Length Places forward control
Columns, which have no corresponding parameter must be initialized with 0.

The following have the same syntax:

set keyword str name float_var value

Set the parameter *keyword* of module *name* on *value*.

Keyword:

speed, objectlength, length, loading_time, unloading_time
loadingpath, unloadingpath, loadingspeed, unloadingspeed,
xSpeedslow, xSpeedfast, xSlowpath,
ySpeedslow, ySpeedfast, ySlowpath,
lifting_time, table_distance, stop_posotion, palspeed, pallength

set keyword str name int_var value

Set the parameter *keyword* of module *name* on *value*.

Keyword:

places, opening_time, closing_time

set keyword str name int value

Set the parameter *keyword* of module *name* on *value*.

Keyword:

info-element, forwardcontrol, normal_position, acceleration, waiting,
destination_address, new_destination_addr, loading_prio, automatically, stop_allowed

set comment str name str: value

Sets the parameter *comment* of module *name* on *value*.

set automatic str name int: agv int: apl

Set automatic pathfinding for ATS or work plans, if the corresponding value is 1.

set cycletime

This command has three variants according the module type.

Workstation:

set cycletime str name int typ1 float_var Llim float_var Ulim
int strat (int min) (int max) (int qual)



Set the cycle time of module *name*. *Typ1* is the object type. If this is not available, a new entry will be generated.

Assembly:

```
set cycletime      str name      int typ1      int waiting
                  int strat     (int min)    (int max)    (int qual)
```

Set the cycle time of module *name* and the object type *Typ1*. If this is not available, a new entry will be generated. *Waiting* means waiting for assembly for this object type.

Disassembly:

```
set cycletime      str name      int type1     int type2      str DistFkt      float L1  float U1
                  int strat     (int min)    (int max)    (int qual)
```

Set the cycle time of module *name*. *Type1* is the object type. If this is not available, a new entry will be generated.. *Type2* is the new object type. The name of the distribution function defines the type of distribution.

Llim und *Ulim* are depended on the distribution function:

Fixed : *Llim* is cycle time, *Ulim* is ignored

Uniformly: *Llim* is lower limit, *Ulim* is upper limit

Normal: *Llim* is mean value, *Ulim* is deviation

Exponentially: *Llim* is mean value

Erlang : *Llim* is mean value, *Ulim* is k-parameter

Strat: Set worker strategy.

0=without
1=no interrupt
2=with interrupt

The optional parameter *min*, *max* und *qual* are the corresponding parameters of the strategy and have to be set, when strategy 1 or 2 are used!

This operation can be used only with workstations.

According to less place the parameter are listed in two lines. In the spread sheet all parameters have to be placed in the same line.

```
set newtype      str name      int typ1      int typ2      float w
```

Set an entry of the list of new types of module *name*. *Typ1* is an existing type of the working list (entering object), *typ2* the new type. *W* is the probability.

This operation can be used only with workstations.

```
set setuptime     str name      int typ1      int typ2      float Rtime      int strat
                (int min)    (int max)    (int qual)
```

Set an entry in the set-up list of the workstation. *Typ1* is the last object, *typ2* the actual object. If there is no entry for this combination, a new entry will be created.. *Rtime* is the set-up time.

Strat: Set worker strategy.

0=without
1=no interrupt
2=with interrupt

The optional parameter *min*, *max* und *qual* are the corresponding parameters of the strategy and have to be set, when strategy 1 or 2 are used!



This operation can be used only with workstations.

set distribution str name str Distribution

Set the distribution function of the cycle time of module *name*. The parameter *Distribution* can be one of the following:

FIXED
UNIFORMLY
NORMAL
EXPONENTIALLY
ERLANG

set emptied str name int x1 int x2 int x3

Defines the strategy of a bulk section.

set battery str name int art float loadspeed

Defines the loading speed at the workstation in an AT-System.

set position str name str connection int posno float pos
(float height)

Defines the positions of entrance/exists of a shuttle and paired shuttle (no height). The value of *connection* defines, whether an entrance or an exit should be modified.

Remark: The behavior of dynamic lists, declared on the example of the object types of a workstation.

If an object type is referenced, which cannot be found in the current list, a new entry for this type will be created. If an existing object type should be replaced, the whole list has to be deleted. This happens with the command „**set cycletime** *name* 0“. Afterwards a new entry for each object has to be created (incl. the list of new object types).

An exception from this rule must be considered with failure times. Since here the key of 0 is a valid parameter, the lists are deleted by the key **-1**.

27.8.1.3 Work Area Parameter

set tasks	str name	int prio	int interrupt	str kind
int change	int fail_change	int prio_change	int setup_change	
int value	str module	str failure		

Set the parameter of the *tasks* of the work area *name*. If no module or failure is assigned, a “#” appears in that field.

set pathlist str name int bs1 int bs2 float Value

Set the time for the way from module *bs1* to module *bs2* to value *Value*. In the next columns of an export the names of the modules will appear. These are ignored during the interpretation.

27.8.2 Command Get

get Transfer data from DOSIMIS-3 into the excel sheet.

The data behind the module name have the same format as with the set command. All entries till the search entry (the entry, which makes the search unique) are necessary. The result will be found in the following column(s).

Syntax (Example Parameter):



Parameter:

name: Name of the module

get	length	str name	
get	objectlength	str name	
get	cycletime	str name	int type
get	setuptime	str name	int type1 int type2

Syntax (Example Simulation results):

get throughput str name float time int stat

Parameter:

name: Name of the module

time: Time of statistic

stat: Kind of statistic:

0: INTERIM_STATISTICS

1: INTERVAL_STATISTICS

2: END_STATISTICS

3: PRERUN_STATISTICS

4: LAST_STATISTICS (time is ignored)

Beneath *throughput* the following keywords are valid:

aver_occupation, *perc_occupation* *utilization*, *blocking*, *fault*, *pause*, *waitforworker*, *nostatistic*, *throughput*, *act_occupation*, *max_occupation*, *min_occupation*, *waitingforparts*, *waiting*, *ObjThroughput*.

Only the command *ObjThroughput* forces a further parameter. By this the object type is defined, which has to be investigate. The total throughput, independent on the object type, can be examined by the command *throughput*.

Further keywords can be recalled by the command *Keyword Moduleresults*.

27.8.3 Command Start

start Start a DOSIMIS-3 - function

Syntax:

start simulation

Start simulation run. ATTENTION! The DOSIMIS-3-projekt file will be saved automatically.

start batchfile Parameter1 Parameter2 Parameter3 Parameter4

Start a batchfile with parameter 1 to 4.

Rem: The start of the simulation program from a batchfile takes place with the command

start /low /wait C:\Programme\SDZ\Dosimis-3\English\ds3sim.exe file.mfs

Meaning:

/low The process will be started with the lowest priority.

/wait After starting the process will wait, until the simulation run is finished, before the next instruction is executed.

.mfs The extension **mfs** is necessary.

Call other batch files with the command **call**. With that instruction the process will return to continue next statement.



Further information according batch files will be found in the windows documentation.

27.8.4 Command Save

save Save the DOSIMIS-3-Project

Syntax:

save

27.8.5 Command Save_as

save_as Save the DOSIMIS-3-Project with different name.

Syntax:

Save_as str filename

Parameter:

Filename: New file name of the project (no extension).

27.8.6 Command Goto

Goto Continue the execution on a different line of the table.

Syntax:

Goto int line

Parameter:

line: Line in the Excel-sheet, where the execution will be continued.

27.8.7 Command Keyword

Keyword transfers the keywords of a group of names into the excel sheet.

Syntax:

Keyword str key

Parameter:

key: group name, whose keywords will be transferred into the Excel sheet. Admissible names are: *Moduleresults* and *Workarearesults*.

Rem.: Since the keywords of the module results refer both to the standard statistics and to the module-type-dependent additional statistics, not all keywords are combinable with all modules types. The syntax of these instruction are referenced in the paragraph of *command get*.



27.8.8 Further Commands

; Comment

Syntax:

; arbitrary text

end Stop interpretation

Syntax:

end

Alternative an empty cell can be used as a keyword.

27.9 Example

Generate a DOSIMIS-3 project, which contains a source, a sink and a work station, which are connected by nodes with each other. Adjust valid parameters for these modules (please, reference the DOSIMIS-3 documentation for this)

- Now you give the workstation the name WST_1 and you set the velocity to 0.5 m/s.
- No work time distributions, no new types and no set-up times should be defined!
- Save the project as „xltest1.mfs“.

Now you start Microsoft-Excel and you create a new Excel table. You register the following on the first table sheet, beginning at cell A1 (# serves here as separators for cells). Figure 27.2 shows the finished table.

```
set # cycletime# WST_1 # 1 # 0,3 # 0,5 # 0
get # speed# WST_1
set # speed# WST_1 # =D2+0,1
set # distribution# WST_1 # 1
save_as # xltest2
end
```

Now close (!) the table, but not the application. Specify as file name xltest1.xls (remember the directory).



	A	B	C	D	E	F	G	H
1	set	cycletime	WST_1	1	0,3	0,5	0	
2	get	speed	WST_1					
3	set	speed	WST_1	0,1				
4	save_as	xltest2						
5	end							
6								
7								
8								
9								

Figure 27.2: Excel table *xltest1.xls* before execution

Now go back to DOSIMIS-3. You should see the project *xltest1mfs* on your display. Select the menu Model/Excel-Data-Transfer/Open. Now you specify the name of the Excel table just generated (*xltest1.xls*). Guarantee that you are in the correct directory. If you now display your still open Excel application by the task bar, you will notice that *xltest1.xls* was opened.

Select the function Model/Excel-Data-Transfer/run in DOSIMIS-3.

- The table script is now processed. An entry is at first added to the work station into work time distribution (line 1).
- Afterwards the velocity of the work station is transferred (0.5 m/sec) in the Excel table into cell D2 (line 2).
- Now the transferred velocity is increased by 0,1 and returned to the work station (line 3).
- Finally the DOSIMIS-3 model is saved as „*xltest2.mfs*“ (line 4) and the processing completed (line 5).

Consequently, the velocity of the work station is increased by 0,1 m/s with each restart of the script. Now one can modify the table and let e.g. execute simulation.

Bring Excel to this to the display. As you to see the table was changed: The keyword *get* has cell D2 filled with the current velocity. Cell D3 was adapted accordingly by the Excel formula.

Now insert the following line before the keyword *end*:

start # simulation

The table should look as follows:



	A	B	C	D	E	F	G	H	I
1	set	cycletime	WST_1	1	0,3	0,5	0		
2	get	speed	WST_1	0,6					
3	set	speed	WST_1	0,7					
4	save_as	xltest2							
5	start	simulation							
6	end								
7									
8									
9									

Figure 27.3: The changed table, after unique run

You do not need to store this table now. Since the Excel-Data-Transfer is still opened, the current data are transferred automatically.

Select the function *run* in the menu item *Excel-Data-Transfer*. As you see, the velocity is still increased by 0,1 m/s and a simulation run begins.

It would not have been in principle necessary to store the table before simulation. However, one has two advantages as a result:

- DOSIMIS-3 never inquires whether is to be stored, if data were modified (automation).
- By the keyword „*save as*“ one can store the changed modules under other name.

With each restart of the script the velocity of AST_1 is increased by 0,1 m/s, the project saved and a simulation run is executed. Experiment with the transfer function. In cooperation with the Excel formula functions, you will know to soon estimate the new possibilities certainly.

If you would like to close the data transfer, select *Model/Excel-Data-Transfer/Close*. The Excel table within Excel is now closed. If changes were practiced, Excel asks you whether you would like to save this.

27.10 Note

- If you open the Excel interface, the table should not already be opened in a current Excel application. Reason for this is an error at Excel. Excel asks the user whether he would like to reload the already opened (and changed) table. If one denies here, it comes for the error message "OPEN method of the Workbooks object is incorrect" (this messages may be differ in the English version of Excel).
- If you have a Excel cell in editing, i.e. a cursor flashes within the cell, *run* cannot access onto the table. Then the following error message appears: „This process cannot be finished, because ..." is active (this messages may be differ in the English version of Excel).
- You should not close the table opened for transfer within Excel since no more data exchange is from now on possible with the table . You should use *Model/Excel-Data-*



Transfer/Close. Otherwise the error message results: „The called object was separated from the clients“ (this messages may be differ in the English version of Excel).

- If the connection between Excel cannot be set up, start Excel and try again to open a table from DOSIMIS-3.

However, none of the error named here should lead to data loss. Applications continues normally after acknowledgement.



28 COM - Server

28.1 Introduction

In order to automate processes during a project, the possibility exists of controlling Dosimis-3 e.g. from Visual basic (VBA). For this the COM interface is available, which supplies the control functions. Basically each program, which supports COM server, can be used.

28.2 Data Types

Dosimis-3 makes 9 data types available. The two most important are *IDs3Application* and *IDs3Document*. The first one corresponds to the program, while the second type corresponds to a Dosimis-3 model. The other (*IDs3Control*, *IDs3Element*, *IDs3Evaluation*, *IDs3Failure*, *IDs3Junction* and *IDs3Workarea*) correspond to a control, a module, an evaluation module, a failure, a junction between two modules and a work area of the model.

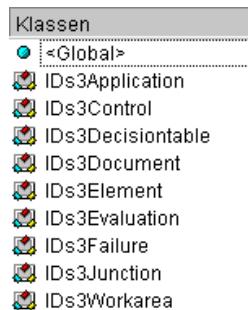


Figure 28.1: the different data types

The start up is made by the type *IDs3Application*. Then with the methods of *IDs3Application* you can open e.g. a model and start the simulation.

The class *IDs3Decisiontable* does not support further methods.



28.3 Methods

28.3.1 Overview

With the exception of IDs3Document all types have only a small number of methods. These are appropriate to open a model, to manipulate it a little bit and to let the model be simulated. Additionally simulation results can be recalled.

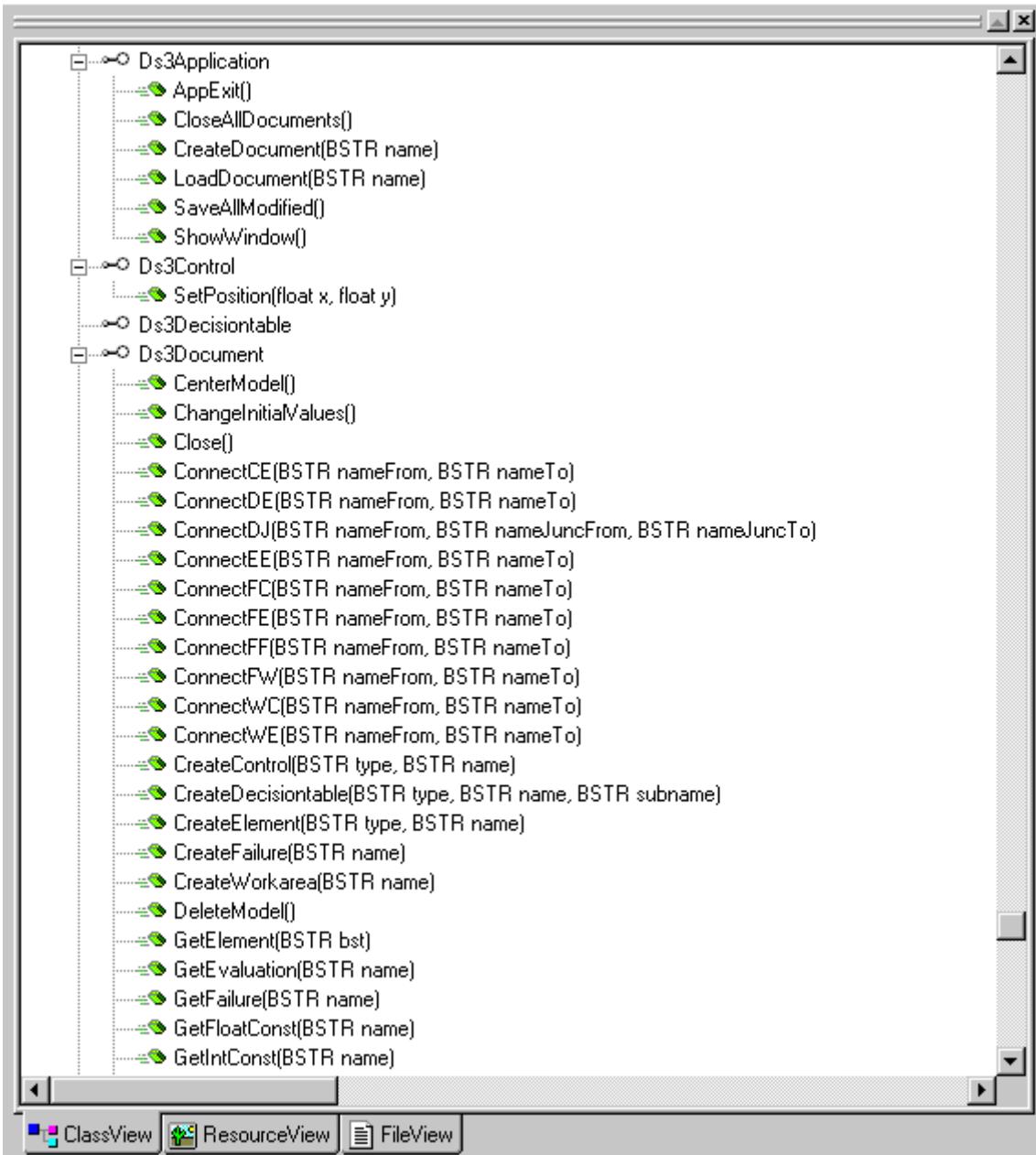


Figure 28.2: Overview methods

A further group of functions support the creation of new elements in a model as well as creating whole simulation models. This applies specially to many function of IDs3Document.



During the description the data types IDispatch, BSTR and VARIANT have a special meaning. These have other names in VBA, their meaning are however in all cases nearly the same.

Data type	Description	VBA - Data type
IDispatch	Reference to a COM-Object	Object
BSTR	Text variable or Text constant	String
Float	Single accurate floating value.	Single
VARIANT	Variable with variable data type (e.g. real Variant valued or integer)	

28.3.2 Methods of IDs3Application

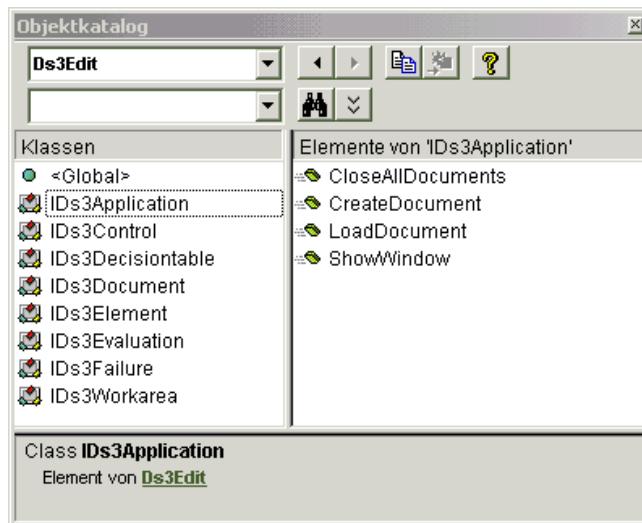


Figure 28.3: Methods of IDs3Application

- void **AppExit ()**
Close all documents and exit application.
- IDispatch* **CreateDocument(BSTR name)**
Create a new Dosimis-3 model.
- void **CloseAllDocuments()**
Close all models.
- long **GetVersionNumber ()**
Liefert die Versionsnummer von Dosimis-3. Diese besteht aus 5 Ziffern. Die erste Ziffer bestimmt die Hauptnummer, die nächsten beiden Ziffern die Unternummer und die letzten beiden Ziffern die Revisionnummer. Beispiel: Version 4.2b wird als 40202 codiert.
- BSTR **GetVersionString()**
Liefert die Versionsnummer von Dosimis-3 als Klartext.
- IDispatch* **LoadDocument(BSTR name)**
Open a Dosimis-3 Model.
- void **SaveAllModified ()**
Save all open documents, when changes have been made.
- void **ShowWindow()**
Make the application window visible.



The central method of the application is *LoadDocument*. This expects the complete path of a model including the extension (.mfs). If Dosimis-3 not already executes in foreground, the application windows are made visible through the command *ShowWindow*. At the end of the handling Dosimis-3 can be terminated with *CloseAllDocuments*. Perhaps it is to check with the method *SaveAllModified* whether modifications have to be saved, so no changes get lost. Additionally Dosimis-3 can be finished by *AppExit*.

28.3.3 Methods of IDs3Control

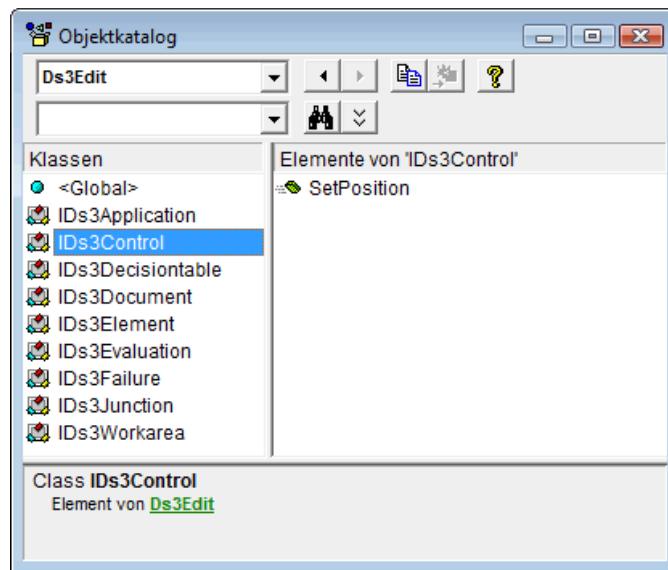


Figure 28.4: Methods of *IDs3Control*

- **void SetPosition(float x, float y)**
Set or change the position of a control (global control, capacity monitor, ...) in the modell to the coordinates (x/y).

28.3.4 Methods of IDs3Document

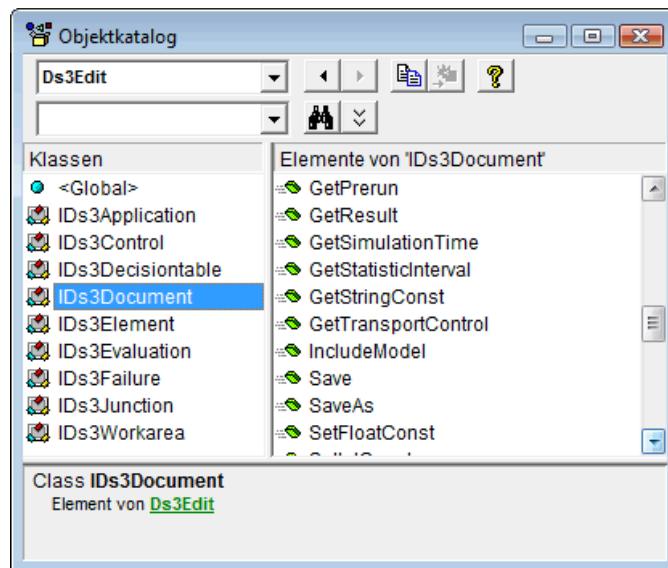


Figure 28.5: Methods of *IDs3Document*



- **void CenterModel().**
Change the coordinates of the elements, so that the model is centered in the middle of the work area.
- **void ChangeInitialValues().**
Change all rand seeds of the actual model.
- **void Close()**
Close the model.
- **void ConnectEE(BSTR nameFrom, BSTR nameTo)**
- **void ConnectFE(BSTR nameFrom, BSTR nameTo)**
- **void ConnectFC(BSTR nameFrom, BSTR nameTo)**
- **void ConnectFF(BSTR nameFrom, BSTR nameTo)**
- **void ConnectFW(BSTR nameFrom, BSTR nameTo)**
- **void ConnectWE(BSTR nameFrom, BSTR nameTo)**
- **void ConnectWC(BSTR nameFrom, BSTR nameTo)**
- **void ConnectCE(BSTR nameFrom, BSTR nameTo)**
- **void ConnectDE(BSTR nameFrom, BSTR nameTo)**
Creates a connection between to elements of Dosimis-3. *NameFrom* indicates the start element, *nameTo* the destination element. The single letters mean
 - E : Element
 - F : Failure
 - C : Control
 - W : Workarea
 - D : Decision table.
- **void ConnectDJ(BSTR nameFrom, BSTR nameJuncFrom, BSTR nameJuncTo)**
Creates a connection between a decision table and a junction. *NameFrom* indicates the start element, *nameJuncFrom* the name of the element in front of the junction and *nameJuncTo* the name of the element behind the junction.
- **IDispatch* CreateElement(BSTR type, BSTR name)**
Create an element of type *type* with the name *name*. *Type* is the (german) plain text of the type of the element (QUELLE, SENKE, ...).
- **IDispatch* CreateFailure(BSTR name)**
Create a failure wirth the name *name*.
- **IDispatch* CreateControl(BSTR type, BSTR name)**
Create a control of type *type* with the name *name*. *Type* is the (german) plain text of the type of the control (BEREICKONTROLLE, TIMER, QUICKTABLE, ...).
- **IDispatch* CreateWorkarea(BSTR name)**
Create a workarea with the name *name*.
- **IDispatch* CreateDecisiontable(BSTR type, BSTR name, BSTR subname)**
Create a decision table with the name *name*. *Type* is the type of the decision table (only CONTROL at this moment). *Subname* is the name of the global control the decision table is assigned to. This control must exist. It is to be created in advanced by the method *CreateControl*.
- **void DeleteModel().**
Deletes all elements of the current model and reinitializes the material flow system.
- **IDispatch* GetElement(BSTR name)**
Provide a reference to the data type IDs3Element. This is the first found module with the name *name*.



- **IDispatch* GetEvaluation (BSTR name)**
Provide a reference to the data type IDs3Evaluation. This is the first found evaluation with the name *name*.
- **IDispatch* GetFailure(BSTR name)**
Provide a reference to the data type IDs3Failure. This is the first found failure with the name *name*.
- **double GetFloatConst(BSTR name)**
Results in the value of the float - constant with the name *name*.
- **long GetIntConst(BSTR name)**
Results in the value of the integer - constant with the name *name*.
- **IDispatch* GetJunction(BSTR nameFrom, BSTR nameTo)**
Provide a reference to the data type IDs3Junction., This is the first junction found whichew leads from module *nameFrom* to module *nameTo*.
- **long GetPrerun()**
Gets the prerun time of the model.
- **VARIANT GetResult(BSTR ResType, BSTR name, double Time, long StatType, long ObjType)**
Provide the simulation result *ResType* of the module *name* at the point in time *time* and the type of statistics the *StatType*. Because some statistic types depend on the type of object, a further parameter *ObjType* is given, which is ignored in other cases.
Concerning the keywords, which *ResType* may assume, the same applies, as described in the Excel - file interface, description of the [get command](#) in combination with simulation results.
- **long GetSimulationtime()**
Gets the simulation time of the model.
- **long GetStatisticinterval ()**
Gets the durartion between two statistic outputs of the model.
- **BSTR GetStringConst(BSTR name)**
Results in the value of the text - constant with the name *name*.
- **BSTR GetTransportControl (long vehicle)**
Results in the name of the tranposrt control whiche schedules the vehicle *vehicle*.
- **void IncludeModel (BSTR name, float x, float y, BSTR prefix)**
Inserts a modell at position x/y of the actual model. The text *prefix* is placed in front of all names.
- **void Save()**
Save the model. This instruction should be executed between a parameter modification and the start of the simulation, since otherwise a safety query takes place.
- **void SaveAs(String name)**
Save the model with the name *name*.
- **void SetFloatConst(BSTR name, double value)**
Modify the float - constant with the name *name* to the value *value*.
- **void SetIntConst(BSTR name, long value).**
Modify the integer - constant with the name *name* to the value *value*.
- **void SetParameter1(VARIANT type, VARIANT name, VARIANT para, VARIANT value1);**
- **void SetParameter2(VARIANT type, VARIANT name, VARIANT para, VARIANT value1, VARIANT value2);**
- **: : : : :**
- **void SetParameter11(VARIANT type, VARIANT name, VARIANT para, VARIANT value1, VARIANT value2, VARIANT value3, VARIANT value4,**



- VARIANT value5, VARIANT value6, VARIANT value7, VARIANT value8,
 VARIANT value9, VARIANT value10, VARIANT value11);
- void **SetParameter12**(VARIANT type, VARIANT name, VARIANT para,
 VARIANT value1, VARIANT value2, VARIANT value3, VARIANT value4,
 VARIANT value5, VARIANT value6, VARIANT value7, VARIANT value8,
 VARIANT value9, VARIANT value10, VARIANT value11, VARIANT value12);
 With the help of these functions the parameters of the Dosimis-3 elements can be set or changed. The calling conventions are the same as in the excel interface. The first two parameters select the element, where *type* is the type of the element (Module (or Element), Workarea, Failure, ...) and *name* is the name of the element. *Para* identifies the parameter, which is to be changed. Depending on the number of arguments the corresponding function is to select.
 - void **SetPrerun**(long time)
 Set prerun time of the model to *time* [minutes]}
 - void **SetSimulationTime**(long time)
 Set simulation time of the model to *time* [minutes]}
 - void **SetStatisticInterval**(long time)
 Set the time of the statistic interval to *time* [minutes]}
 - void **SetStringConst**(BSTR name, BSTR value)
 Modify the text - constant with the name *name* to the value *value*.
 - void **ShowWindow**()
 Displays the model window.
 - long **StartSimulation**()
 Start the simulation for the model. Return value unequal zero, if simulation has been (erroneous) interrupted.
 - void **ZoomModel**().
 Change the view so that the model is drawn optimal.

With the help of the *create-* and *connect* methods complete simulation models can be build. Further on these can be parameterized by the *SetParameter* - methods.

Central meaning with the parameter manipulation is attached to the two methods *SetFloatVar* and *SetIntVar*. Thus most parameters of the model can be achieved, because in almost all situations the use of variables is enabled. Additionally a failure can be passivated or activated by the methods *SetPassiv* of the type *IDs3Failure*. This must be looked up first in the model by *GetFailure*.

In the case of analyzing the model *GetResult* provides the results of a simulation run. This method refers to the same syntax as the command *Get* from the Excel - Interface.



28.3.5 Methods of IDs3Element

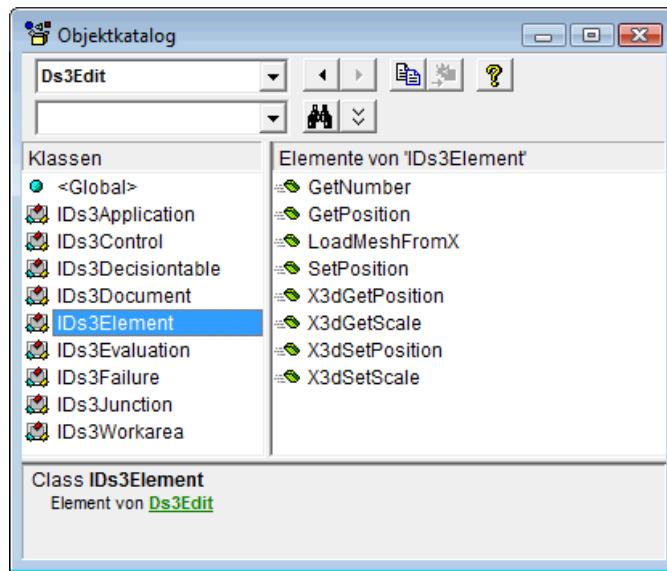


Figure 28.6: Methods of IDs3Element

- void **SetPosition**(long pos, long ri, float x, float y)
Set or change the position of the element in the modell to the coordinates (x/y). *ri* determines the direction of the module. Modules with several way points (shuttle, conveyor, ...) *pos* determines the number of the way point. The first way point counts to 1.
Values of 1 to 4 are allowed for *Ri*. Each value has the meaning of one of the four rectangular directions (1 = 0 degree, 2 = 180 degree, 3 = 270 degree, 4 = 90 degree). The coordinates can have values according the interval from (0,0) to (10000, 10000).
- void **GetPosition**(long pos, long ri, float x, float y)
Results in the direction and the coordinate of the module.
- long **GetNumber**
Results in the number of the current element.



28.3.6 Methods of IDs3Evaluation

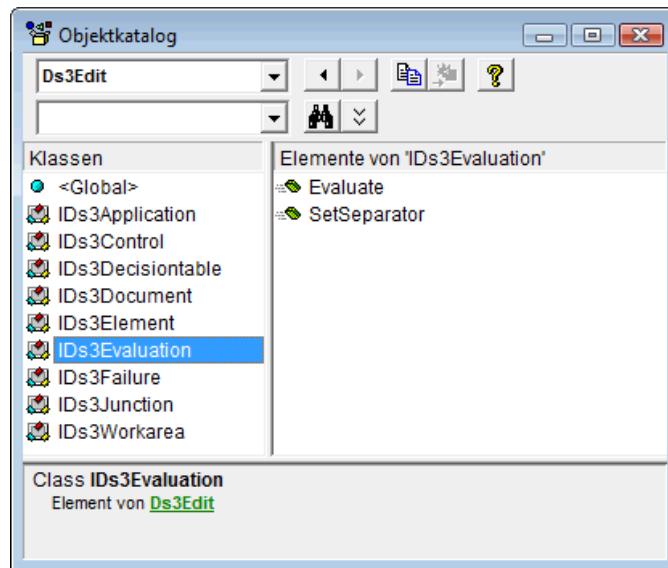


Figure 28.7: Methods of IDs3Evaluation

- **void Evaluate()**
Starts the evaluation. The results are placed into the clipboard. These can be used for further operations by the paste methode of the application.
- **void SetSeparator(BSTR decimal, BSTR list)**
Set the special characters for the field separartor and the decimal separator. In opposite to the country settings Visual Basic operates with the american settings.

28.3.7 Methods of IDs3Failure

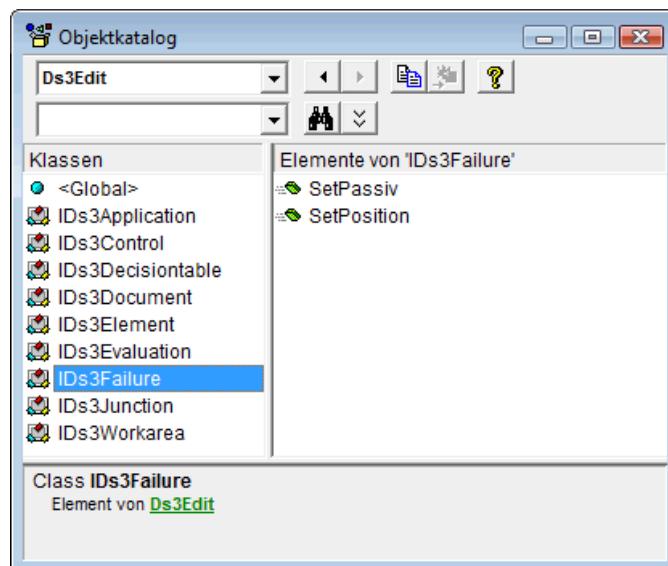


Figure 28.8: Methods of IDs3Failure

- **boolean SetPassiv(boolean passiv)**
Actiavtes or deactivates a failure.



- **void SetPosition(float x, float y)**
Set or change the position of the element in the modell to the coordinates (x/y). These method supplies the possibility of deactivating individual failures to make for example experiments with different layer models in a model.

28.3.8 Methods of IDs3Junction

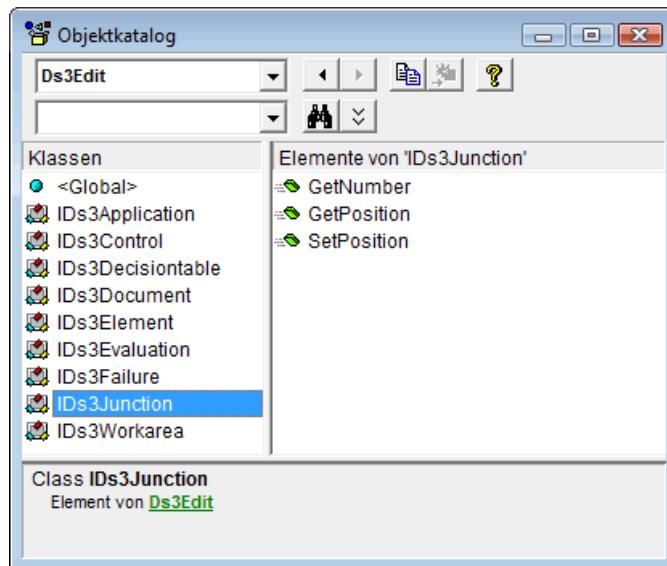


Figure 28.9: Methods of IDs3Junction

- **long GetNumber ()**
returns the number of the junction.
- **void SetPosition(long pos, float x, float y)**
Set or change the position of the element in the modell to the coordinates (x/y). *Pos* determines the nummber of the way point. The first way point is at position 1.
- **void GetPosition(long pos, float x, float y)**
Results in the coordinate of the junction.



28.3.9 Methods of IDs3Workarea

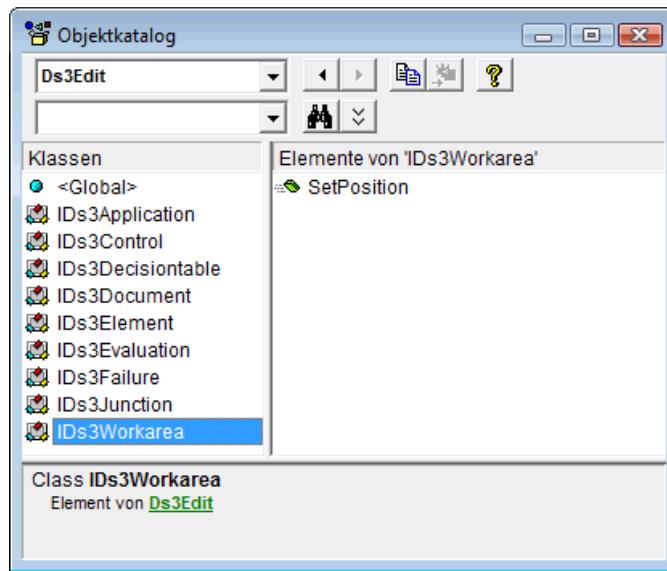


Figure 28.10: Methods of IDs3Workarea

- void **SetPosition**(float x, float y)
Set or change the position of the element in the modell to the coordinates (x/y).



28.4 Connecting to Visual Basic for Excel

In this example the usage with VBA (Visual basic for applications) in interaction with Excel is demonstrated. This presupposes fundamental knowledge in handling Basic and deals only with the COM - specific peculiarities.

In order to use the method, the so-called *Type Library* must be registered. This occurs in the menu by *Extra/Reference*. The Dosimis-3 Type Library is called *Ds3Edit.tlb* and can be found in the installation directory from Dosimis-3.

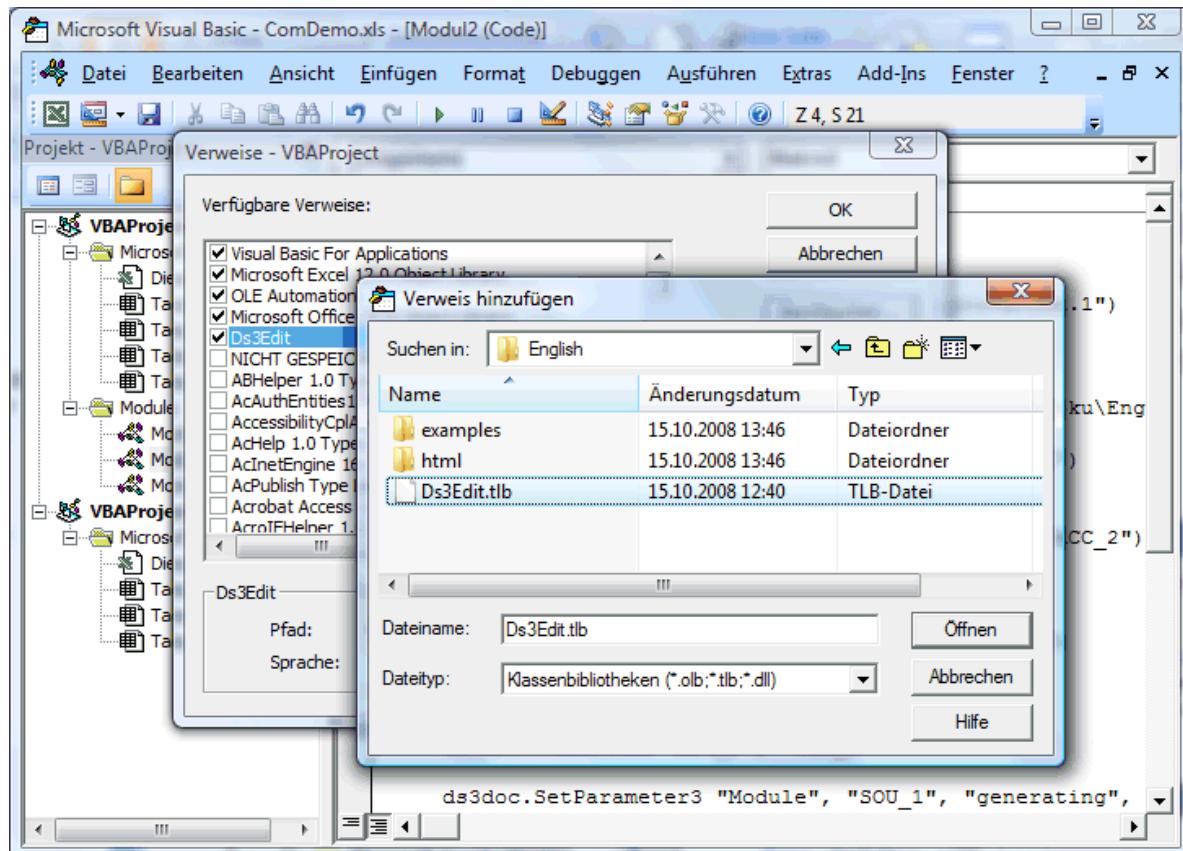


Figure 28.11: Registering the Dosimis-3 COM- Server to VBA



28.4.1 Usage

Operating with VBA presents itself as quite comfortable. VBA knows context depending popup menus, which helps for the definition of the instructions. The declaration type occurs in VBA by means of the *DIM - statement*. This follows the keyword As, whereby after input of the second letter a selection of all registered data types is presented.

Figure 28.12: VBA context menu of data types

An instance of application is created by the method *New* of Visual Basic. From this it can be navigated by the context popup menu of the appropriate instance.

New IDs3Application

If methods are to be accessed, this can take place by the popup menus of the variable, which arises after input of the point.



The screenshot shows the Microsoft Visual Basic Editor interface. The title bar reads "Microsoft Visual Basic - ComDemo.xls - [Modul1 (Code)]". The menu bar includes Datei, Bearbeiten, Ansicht, Einfügen, Format, Debuggen, Ausführen, Extras, Add-Ins, Fenster, and ?.

The left pane displays the "Projekt - VBAProject" window, listing two projects: "VBAProject (ComDemo.xls)" and "VBAProject (xltest1.xls)".

The right pane shows the code editor for "Makro1" in "Modul1". The code is as follows:

```
Sub Makro1()
    Dim ds3app As IDs3Application
    Set ds3app = CreateObject("Dosimis-3.Application.1")

    Dim ds3doc As IDs3Document
    Set ds3doc = ds3app.LoadDocument ("\ds3dev\doku\Deutsch

    Dim ds3sto As IDs3Failure
    Set ds3sto = ds3doc.

    ds3sto.SetPassiv CenterModel
    ActiveWorkbook.App ChangelnitialValues
    Close

    Dim res As Variant
    For i = 1 To 2 Step 1
        ds3doc.SetIntV ConnectCE
        ds3doc.Save ConnectDE
        ds3doc.StartS ConnectDJ
        ds3doc.StartS ConnectEE
        res = ds3doc.GetResult("Durchsatz", "SEN_13", 0, 4
        Sheets("Tabelle1").Select
        rg = "A" & i
        Range(rg).Value = res
    Next
    ds3app.CloseAllDocuments
End Sub
```

Figure 28.13: VBA context menu of methods



28.4.2 Example: Experiment

In the following the use of the COM - interface is to be demonstrated.

```
Sub Makro1()
    Dim ds3app As IDs3Application
    Set ds3app = New IDs3Application
    ds3app.ShowWindow

    Dim ds3doc As IDs3Document
    Set ds3doc = ds3app.LoadDocument(ActiveWorkbook.Path & "\ComDemo.mfs")

    ActiveWorkbook.Application.DisplayAlerts = False
    ActiveWorkbook.Application.ScreenUpdating = False

    For i = 1 To 10 Step 1
        ds3doc.SetIntConst "Segmente", i
        ds3doc.Save
        ds3doc.StartSimulation

        Sheets("Results").Cells(i,1) = ds3doc.GetResult("Throughput", "SEN_13", 0, 4, 0)
    Next
    ds3app.CloseAllDocuments
End Sub
```

If an individual Visual basic instruction lasts too long, a timeout with an error message takes place. In order to suppress this, the method *DisplayAlerts* is available in VBA for Excel. This is set to *False* before entering into the loop, so that this will pass through without interruption. In addition the refresh of the display is suppressed by disabling the method *ScreenUpdating*.

Within the loop the manipulation of the integer variable *Segmente* takes place and afterwards the simulation run. After termination of the calculation the result will transferred into the Excel sheet *Results*. Here by the statements (rg = "A" & i and Range(rg).Value = res) it is guaranteed that in each loop the throughput is entered in a new field of the table.



28.4.3 Example: Model Creation

The following example should demonstrate, how to create a new model from the bottom by using the COM - interface. The new mini model consists only of a source, an accumulating conveyor and a sink.

```

Sub Makrol()
    Dim ds3app As IDs3Application
    Set ds3app = New IDs3Application
    ds3app.ShowWindow

    Dim ds3doc As IDs3Document
    Set ds3doc = ds3app.CreateDocument(ActiveWorkbook.Path & "\ComDemo2.mfs")

    Dim ele As IDs3Element
    Set ele = ds3doc.CreateElement("QUELLE", "SOU_1")
    ele.SetPosition 0, 1, 5000, 5000

    Set ele = ds3doc.CreateElement("STAUSTRECKE", "ACC_2")
    ele.SetPosition 1, 1, 5024, 5000
    ele.SetPosition 2, 1, 5048, 5000

    Set ele = ds3doc.CreateElement("SENKE", "SIN_3")
    ele.SetPosition 0, 1, 5072, 5000

    ds3doc.ConnectEE "SOU_1", "ACC_2"
    ds3doc.ConnectEE "ACC_2", "SIN_3"

    ds3doc.SetParameter3 "Module", "SOU_1", "generating", 1, 0, 1
    ds3doc.SetParameter2 "Module", "SOU_1", "objecttype", 1, 1
    ds3doc.SetParameter4 "Module", "SOU_1", "distribution", 0, "FIXED", 120, 0

    ds3doc.SetParameter1 "Module", "ACC_2", "speed", 0.2
    ds3doc.SetParameter1 "Module", "ACC_2", "objectlength", 1.2
    ds3doc.SetParameter1 "Module", "ACC_2", "places", 5

    ds3doc.SetParameter4 "Module", "SIN_3", "distribution", 0, "FIXED", 20, 0

    ds3doc.ZoomModel
    ds3doc.Save

    ds3app.AppExit
End Sub

```

First of all the three modules are created and placed in the layout by the method *SetPosition*. Afterward the modules are connected with each other by the method *ConnectEE*. In the third area the parameter are set. After getting the total view of the model in the work space the model will be saved and the application is exiting. Consider that the first keyword of the create function must be the German term, all other are language independent in the same manner as the excel interpreter itself.

Determining the coordinates should respect the snap of Dosimis-3. Here an offset of 24 is used. A complete check of the coordinates (rectangle for conveyor, 24 offset with way points, ...) does not take place. Further information can be derived from a given model by exploring the data of representation from the context menu (*Module/Representation*) of a module. An additions work set can be activated from the menu *View/Zoom/Coordinates*, which enables the tracing of the coordinates for each mouse position in the status bar.



28.5 Connecting to Visual C++

In this example the usage with Visual C++ is to be demonstrated. This replies fundamental knowledge in handling Visual C++ and deals only with the COM - specific peculiarities. In a first step the so-called Typelibrary has to be imported. This occurs from the menu *Project/Add Class/MFC/MFC Class From TypeLib*.

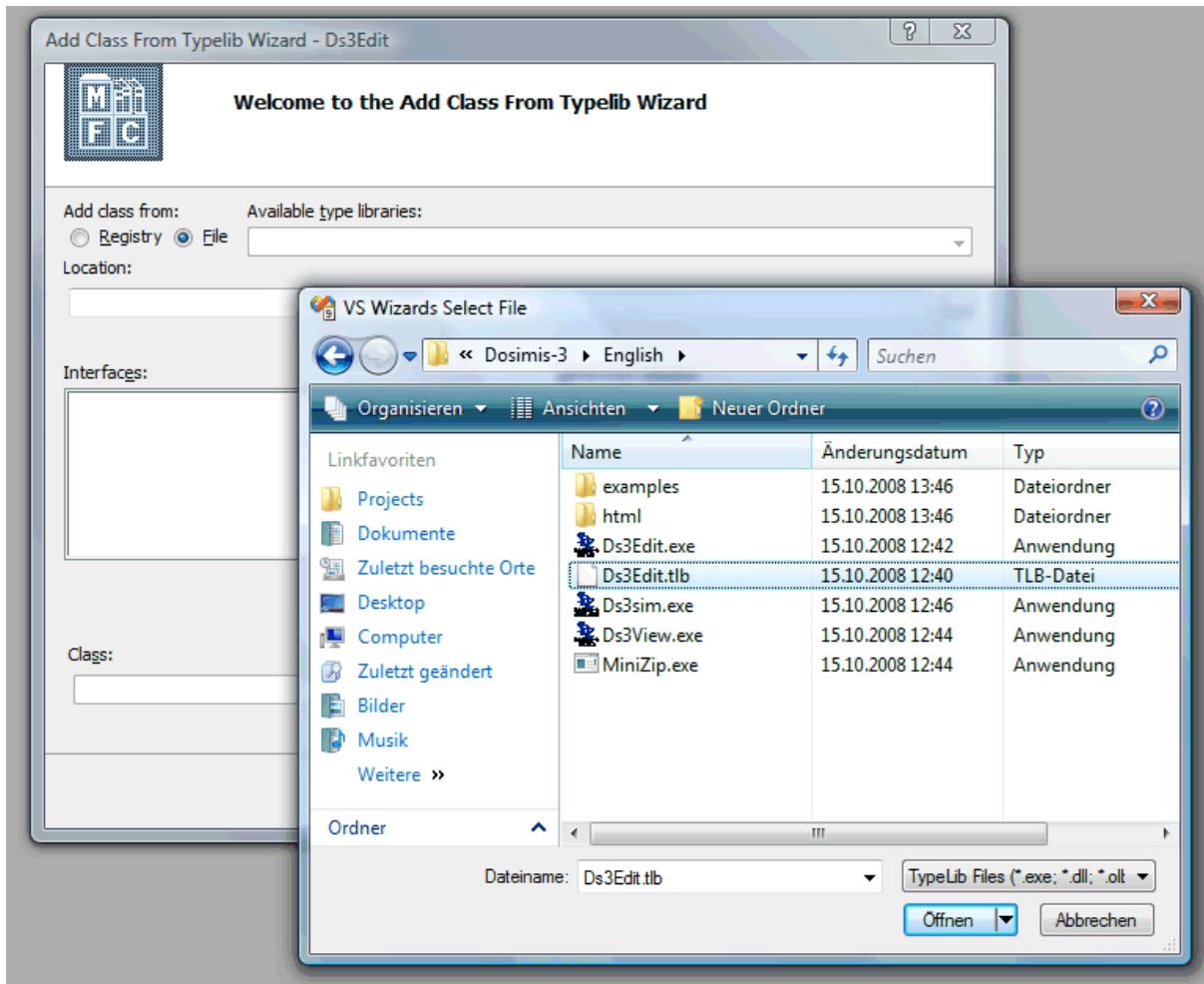


Figure 28.14: Registering of the Dosimis-3 COM - Servers to Visual C++

In order not to lose sight of substantial functionality, in this description the check of the return values does not take place as far as possible. This is a disadvantage opposite Visual basic. There most examinations take place via the run time system.



28.5.1 Usage

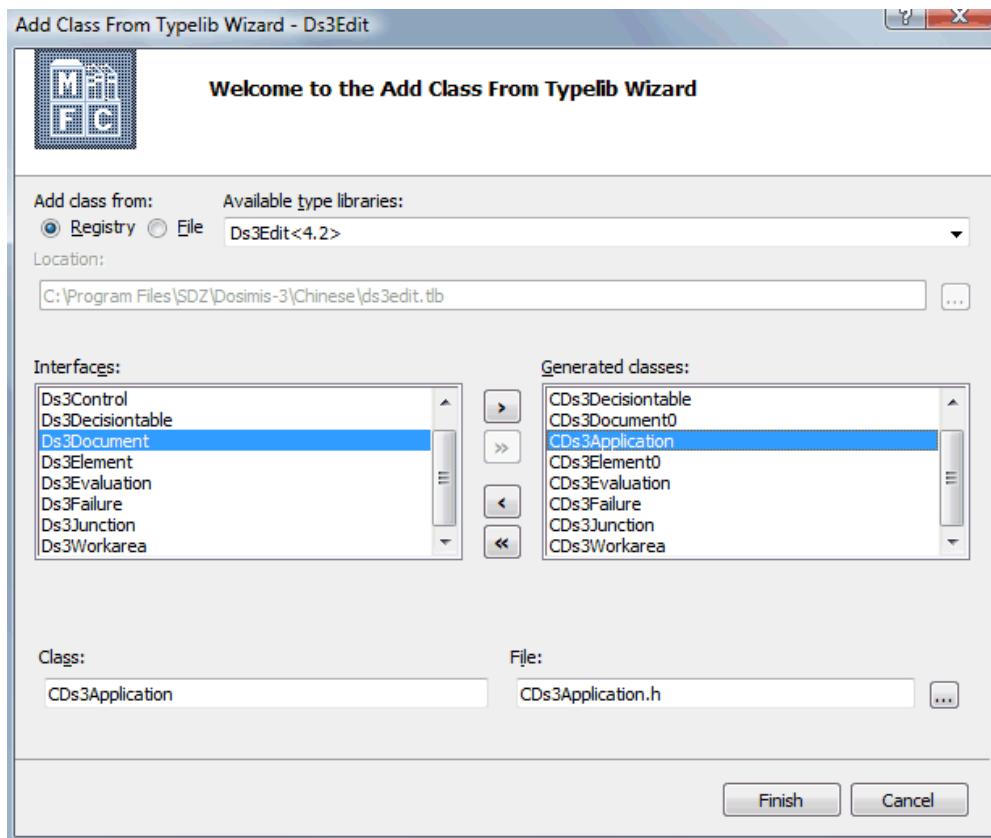


Figure 28.15: Data Types in Visual C++

Before a COM method in Visual C++ can be called, the system must be initialized. This occurs by means of the call `::OleInitialize(NULL)`. This must take place at the beginning of the program. Alternatively this can take place in MFC programs also by means of `AfxOleInit()`.

An instance of application is created by the method `CreateDispatch` of the class **Ds3Application**. If this were successfully created, concerning this the methods of the application can be called.

The method `CreateDispatch` expects a parameter. This must the text

"Dosimis-3.Application.1".

When methods are to be accessed, also this can take place by the popup menus of the variable.

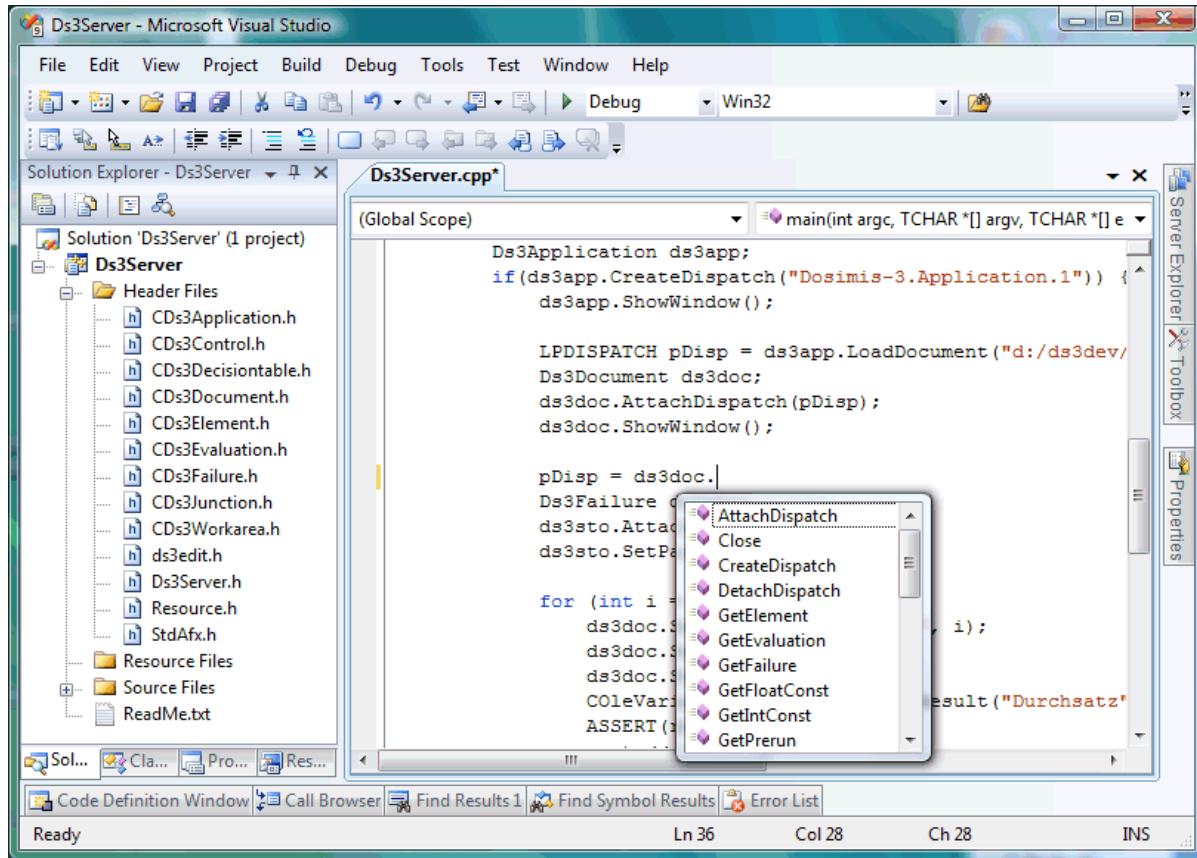


Figure 28.16: Context - Menu of Methods in Visual C++

The handling is however a little bit more pedantically than in VBA. In such a way a model is created by the statements:

- LPDISPATCH pDisp = ds3app.LoadDocument("d:/ds3dev/doku/Bilder/ComDemo.mfs");
- Ds3Document ds3doc;
- ds3doc.AttachDispatch(pDisp);

Between the call of the method and the allocation a so-called dispatcher shifts itself, to which the return value of the method must be assigned and to which the instance connects by *AttachDispatch*.



28.5.2 Example: Experiment

In the following the usage of the COM - interface with Visual C++ to is to be demonstrated. For this a new project of the type *Win32 Console Application* with *Support of MFC* activated was created with the help of the *ProjectWizzard*.

```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    if (AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(), 0)) {
        ::OleInitialize(NULL);

        Ds3Application ds3app;
        if(ds3app.CreateDispatch("Dosimis-3.Application.1")) {

            LPDISPATCH pDisp = ds3app.LoadDocument("d:/ds3dev/doku/Bilder/ComDemo.mfs");
            Ds3Document ds3doc;
            ds3doc.AttachDispatch(pDisp);

            pDisp = ds3doc.GetFailure("STO_1");
            Ds3Failure ds3sto;
            ds3sto.AttachDispatch(pDisp);
            ds3sto.SetPassiv(TRUE);

            for (int i = 1; i < 3; i++) {
                ds3doc.SetIntVar("segments", i);
                ds3doc.Save();
                ds3doc.StartSimulation();
                ColeVariant res = ds3doc.GetResult("Throughput", "SEN_13", 0, 4, 0);
                ASSERT(res.vt == VT_I4);
                cout << res.intValue << endl;
            }
        }
    }
    return 0;
}
```

Within the loop the manipulation of the integer variable "segments" takes place and afterwards the simulation run. After the termination of the calculation the result is written to standard output (*cout << res.intValue << endl;*).





29 Program Interface for Controls

29.1 Introduction

When using Dosimis-3 you will find situations which exceed the functionality of common modeling methods. Those tasks can be modeled by using a program interface. The program interface supplies functionality for changing the behavior of both user interface and simulator, e.g. by programming controls, additional statistics or new layout features.

Using the program interface requires basic knowledge about working with decision tables, especially if changing the behavior of the simulator is desired. Also fundamentals in programming C/C++ and using the development environment Visual Studio/Visual C++ is necessary. How to use C++ classes and virtual functions should be known either. Also few elementary classes from MFC (Microsoft Foundation Classes) will be used.

Usually, programming the application program interface (API) of Dosimis-3 does not require high quality solutions, as they would be necessary in a professional environment. However, this certainly depends on the purpose of the module being created.

- **single solution for one project**

The programming shall work for a single project. Detailed error-catching mechanisms are unnecessary if the program developer himself creates the fitting Dosimis-3 model. In case of an error, the developer knows how to react.

- **development of a control for the whole company**

This requires a stable implementation, because other persons will use the program code. In case of an error the developer is near, so there is no need for a strict resistance against mistakes by users. Typical mistakes should be caught, anyhow.

- **development of general functionality modules for third-party users**

This situation claims an implementation meeting high quality standards. The module must be stable, it must generate meaningful error messages. A module documentation is necessary.

Two areas need to be distinguished: The user interface and simulator. The user interface gives the ability to generate, edit and analyze models. The simulator is responsible for computing the object flow within a model, as defined by its elements and their parameters.

New functionality is made available in so called dynamic link libraries (DLL).

29.2 Little C/C++/MFC Glossary

29.2.1 Keywords

Some few keywords are needed when developing external program libraries for Dosimis-3. Those will be explained in simple words. For an exact explanation refer to the Visual C++ language reference.

const A variable declared *const* can only be read. It must not be on the left side of an assignment.



static	A name which is only locally known (within the current C++ module file).
virtual	A virtual function can be overloaded. Within subclasses, the deepest implementation will be executed when calling a virtual function.
Function() = 0	A function without an implementation (a so called pure virtual function). A class containing any pure virtual function cannot be instanced. In derived classes, pure virtual functions must be defined to be instanced.

29.2.2 Common Words and Terms of C/C++

constructor	This is a function which automatically will be called, when a class gets instanced. Its purpose is to initialize the structure. The function name of the constructor is always the same as the class name itself (e.g. CString::CString()) and may have parameters.
destructor	This function is called just before an instance of a class will be destroyed. Its purpose is to free memory used by additional data. The name of the destructor function is the same as the class name, but gets preceded by a ‘~’ (e.g. CString::~CString()) and must not have parameters. Destructors should be virtual.
instance	The instance of a class is similar to a variable of the class-type. A class is a definition for a construct. An instance is the construct itself. One class can have multiple instances. Instances can be created statically (CString str("text")) or dynamically (CString * str = new CString("text")). Dynamically created instances have to be deleted with the keyword <i>delete</i> after usage, to avoid memory leaks (memory areas which are allocated but not needed any more)

29.2.3 Common Words and Terms of Visual-C++

resource	Next to its implementation, a windows program often requires user interfaces like menus, dialogs, icons, and so on. Those are kept in a resource file. Using Visual C++, you can edit all resources of your project by using the integrated resource editor.
-----------------	--

29.3 Menu

The program interface can be easily accessed and configured by the “Programming” menu.

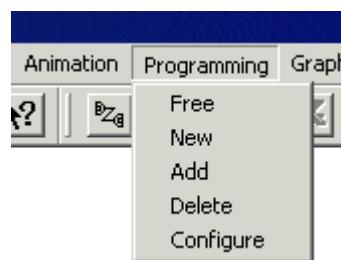


Figure 29.1: Menu

Free/Load	Unload a user library, e.g. for reloading it after being rebuilt. If a user library is not loaded, its symbols cannot be shown in Dosmis-3. Instead a square will be drawn, containing the name of the user library. When compiling a new user library version, it must not be loaded, because
------------------	--

**New**

the linker will abort when writing to a dll-file already in use (*LINK : fatal error LNK1168: cannot open transit.dll for writing*). Unloading a module is only possible if the Dosimis-3 model is unchanged. Otherwise it is necessary to save the model first. Loading a user library will be done automatically when double-clicking on a control symbol.

Add

This will create a new user library. You will find a detailed description in the next chapter.

Delete

Before using a user library, it has to be added to the model. A list of all referenced user libraries will be stored in the file *Ds3Library.ini*, together with certain configuration parameters.

Configure

An unused user library can be deleted from the library list in *Ds3Library.ini*. The user library itself and the C++ project files have to be deleted separately. For a complete deletion, the user library project has to be removed from the Visual-C workspace.

Here you can configure the directories for the development environment (Visual Studio), the templates and the project source code.

When working with the program interface, there should always be a connection between the implementation of user libraries and Dosimis-3. But the user library cannot be rebuilt then - it is write-protected while being opened in Dosimis-3. So it is necessary to unload (free) the user library, before recompiling it. After freeing a user library, it has to be reloaded manually. This can be done by using the *Load* menu or by double-clicking the corresponding control symbol in a Dosimis-3 model. If a user library is not loaded, all according control symbols are drawn as a quadrate containing the library name.

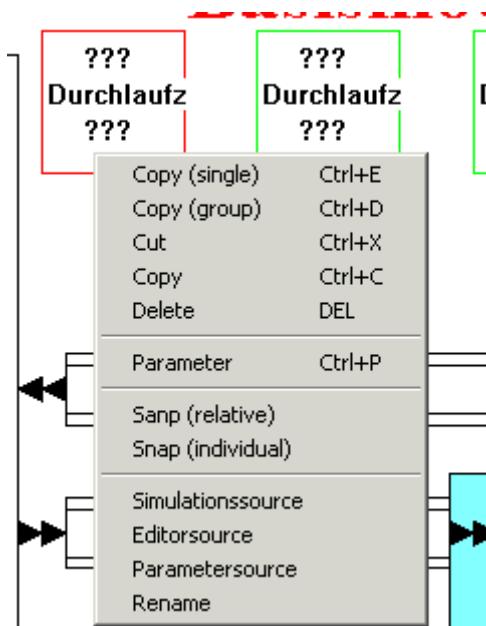


Figure 29.2: Context menu

When opening the context menu of a control, next to the common functionality to all Dosimis-3 elements you can open the development environment for the corresponding user library. Also you may rename the control. Renaming the control means assigning another user library to it, giving it a completely different functionality.



29.4 Configuration

The configuration of the program interface is done by defining three directory paths. By default, the directories are configured with standard values (*c:\program files\....*), but they can be changed to fitting pathnames.

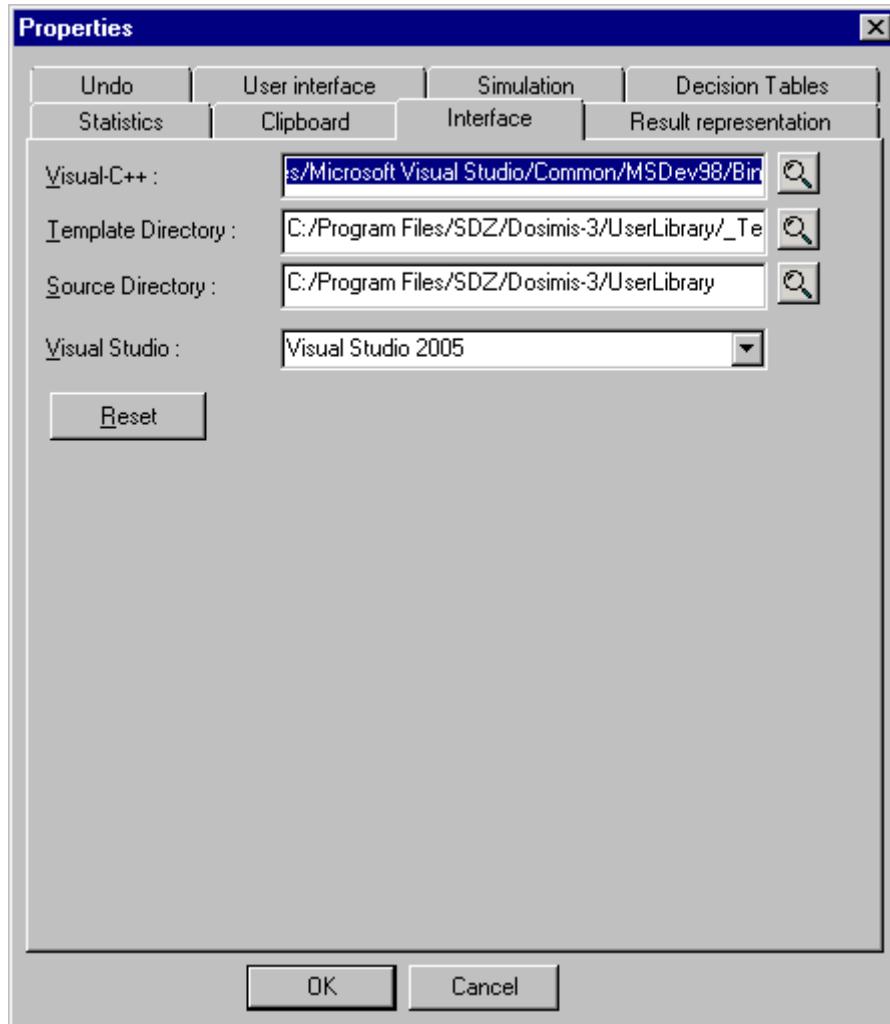


Figure 29.3: Configuration of the program interface

Visual C++	The directory containing the Visual-C++ executable (<i>msdev.exe</i>).
Template Directory	The template directory contains the source files and Visual-C++ project files for generating a new user library. Templates were installed with Dosimis-3 and are located in the Dosimis-3 application directory (<i>\program files\SDZ\Dosimis-3\UserLibrary\Template</i>).
Source Directory	The source directory contains the workspace file <i>UserLibrary.dsw</i> , which contains information about all user library projects.
Visual Studio	The interface can be configured for Visual Studio 2005. Therefore a special Version of Dosimis-3 is needed. Even the files from the directory <i>Unsupported</i> will not fit. Please consult your dealer.
Reset	When using this button, all paths will be set to their default values and the list of used user libraries will be deleted.





30 Integrated Editor

Dosimis-3 has an integrated editor, where small changes can be made from the user interface.

The screenshot shows the DOSIMIS-3 Integrated Editor window titled "Working". The menu bar includes "File", "Edit", "Compile", and "Hilfe". The tab bar at the top shows "Working.cpp | Working.h | WorkingProp.cpp | WorkingProp.h | Compile Error(s)". The main area displays C++ code:

```

#include "stdafx.h"
#include "Working.h"
#include "WorkingRes.h"
#include "WorkingProp.h"

HRESULT CPbsWorking::Draw(CDC* pDC, int px, int py)
{
    DrawFromBitmap(pDC,px,py,IDB_PROGBAUSTEIN);
    return S_OK;
}

BOOL CPbsWorking::DatenOk(FILE * of)
{
    if (m_einfahrdauer <= 0) {
        if (of) {
            fprintf(of, "Einfahrdauer muss > 0 sein!\n");
        }
        return FALSE;
    }
    if (m_bearbeitungsdauer <= 0) {
        if (of) {
            fprintf(of, "Bearbeitungsdauer muss > 0 sein!\n");
        }
        return FALSE;
    }
    return TRUE;
}

```

At the bottom, there are buttons for "next error" and "save & compile". Below the buttons are "OK", "Cancel", and "Help" buttons.

Figure 30.4: integrated editor

This is started from the context menu. All modules, which are according to standard in the template of the element, are shown. In this environment **save & compile** the compilation can be started after changes by that button. The result will be found in the sub window *Compile error (s)*.

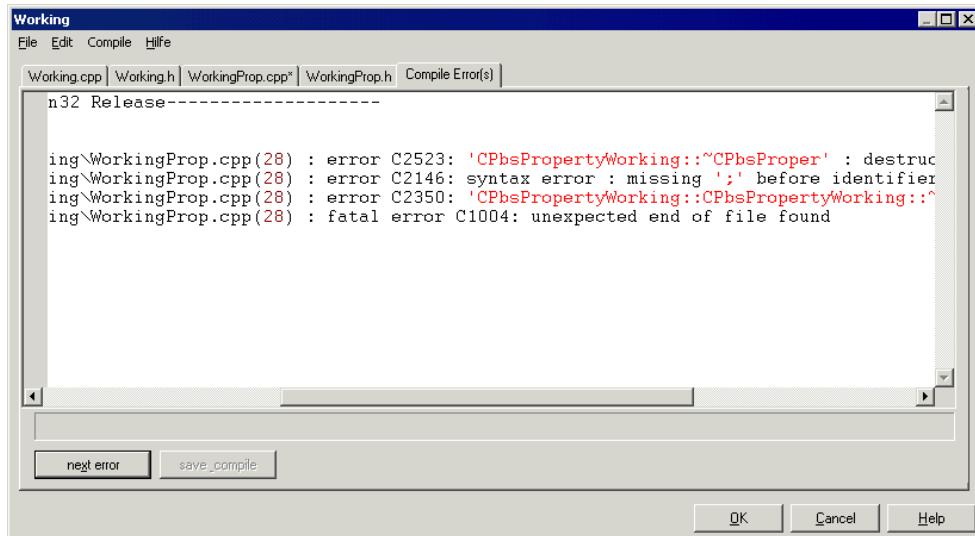


Figure 30.5: error window

After pressing the button **next error** each error will be shown step by step, so it can be corrected.

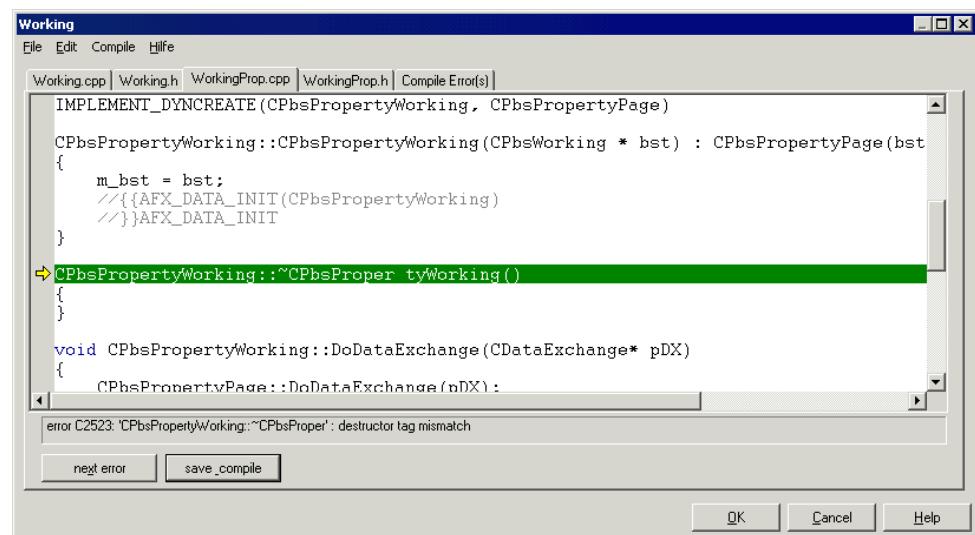


Figure 30.6: error analysis

This editor is not intended to replace the development environment, but to help to repair smaller corrections without large expenditure. As development environment with debugger and the further convenience Visual Studio is recommended.



31 Program Interface Common

31.1 Information of the Interface

The extension of Dosimis-3 with the help of programming takes place with the help of C++ programming. For different usage templates are provided, which give the structure of the implementation. Each template possesses a central function (*GetProgInfo*), from which for example an entry in the toolbar can be created. For this reason two enumerated data types and a data structure exists, where information are taken from. The first enumeration determines the kind of implementation. This is used to create the structure of the toolbar. The fields *pBitmap*, *sToolinfo* and *sMessage* determin the symbol and the tooltip in the toolbar and extended information which are shown in the status bar. *SKey* should contain the letters for a shortcut to the element, which is drawn in the toolbar below the symbol. By the value of *nVersion* the version of the implementation will be checked. Version are up compatible. Only a warning takes place. A down compatibility is newer save, because some function in the interface library may be not available.

```

enum EPrgType {
    EPT_UNKNOWN,
    EPT_STEUERUNG,
    EPT_WEICHE,
    EPT_BEARBEITUNG,
    EPT_LAST
};

enum ECmpType {
    ECT_UNKNOWN,
    ECT_DEBUG,
    ECT_RELEASE
};

struct ProgInfo {
    CBitmap * pBitmap; //!< Bitmap in der Palette
    char * sToolinfo; //!< Text, der als Tooltip angezeigt wird
    char * sMessage; //!< Text, der in der Statuszeile angezeigt wird
    char * sKey; //!< 3 Buchstabiger Text, der in der Toolbar platziert wird
    EPrgType pType; //!< Prog - Type Steuerung, Weiche, Bearbeitung
    ECmpType cType; //!< CompileType 1 : Debug, 2: Release
    long nVersion; //!< Versionskennung der dll
};

```



```

extern "C" AFX_EXT_API HRESULT GetProgInfo(ProgInfo * pInfo)
{
    static CBitmap bmp;
    if (!bmp.GetSafeHandle()) {
        bmp.LoadBitmap(IDB_PROGBAUSTEIN);
    }
    pInfo->pBitmap = &bmp;
    pInfo->sMessage = "Erzeugt einen Baustein vom Typ Arbeit";
    pInfo->sToolinfo = "Arbeit";
    pInfo->sKey = "ARB";
    pInfo->pType = EPT_BEARBEITUNG; // Baustein
    pInfo->nVersion = IFC_VERSION;
#ifndef _DEBUG
    pInfo->cType = ECT_DEBUG;
#else
    pInfo->cType = ECT_RELEASE;
#endif
    return NO_ERROR;
}

```

Older modules should extent this function, because it is needed to create the toolbar of programming modules.

A further function, which all interfaces must support, is the create - function, which creates an instance of the element. These are:

```

extern "C" AFX_EXT_API HRESULT CreateProgBaustein(IProgBaustein ** result)
{
    // Diese Folgezeile korrigieren, wenn der Klassenname geändert wurde
    *result = new ...;
    return NO_ERROR;
}
and/or.
extern "C" AFX_EXT_API HRESULT CreateProgStrg(IProgStrg ** result)
{
    // Diese Folgezeile korrigieren, wenn der Klassenname geändert wurde
    *result = new ...;
    return NO_ERROR;
}

```

The names of these functions should not be changed.

31.2 Debugging User Libraries

31.2.1 Debug by Using Control Output

Debugging the Simulation Interface

Debug mechanisms a useful specially when simulating. It can be done by writing control output to a file on selected events. This protocol file has to be opened on initializing the simulation (SimInitialisierung). To synchronize output when animating your model, use the functions *atx_printV* (output to animation text) and *prot_printV* (output to the protocol window).

Debugging the User Interface

It is more complicated to debug your user interface code, because there is no unique initialization sequence. You should initialize data, open and close files within the function DllMain, which is located in the “*projectname*”.*cpp* file.



```

FILE * protokoll = NULL;
extern "C" int APIENTRY
DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID lpReserved)
{
    // Remove this if you use lpReserved
    UNREFERENCED_PARAMETER(lpReserved);

    if (dwReason == DLL_PROCESS_ATTACH)
    {
        TRACE0 ("DURCHLAUFZEIT.DLL Initializing!\n");

        // Extension DLL one-time initialization
        if (!AfxInitExtensionModule(DurchlaufzeitDLL, hInstance))
            return 0;
        :
        :
        new CDynLinkLibrary(DurchlaufzeitDLL);
        if (protokoll == NULL) {
            protokoll = fopen("protokoll.txt","w");
        }
    }
    else if (dwReason == DLL_PROCESS_DETACH)
    {
        TRACE0 ("DURCHLAUFZEIT.DLL Terminating!\n");
        // Terminate the library before destructors are called
        AfxTermExtensionModule(DurchlaufzeitDLL);
        if (protokoll != NULL) {
            fclose(protokoll);
            protokoll = NULL;
        }
    }
    return 1;    // ok
}

```

You variables can be accessed by other cpp files by declaring them as *extern* within the header file “*projectname*.h”:

```

void SetSimlib(ISimlib * slb); //!< \sa IProgStrg::SetSimlib
extern FILE * protokoll;
#endif

```



31.2.2 Debugging using the Visual-C++ - Debugger (Visual Studio 2008 / 2010)

Advanced users may use the Visual C++ debugger. Visual Studio distinguishes between a *release* and a *debug* build of a project. Proprietary applications like Dosimis-3 are shipped in a release build. But data types differ in the different build types, so it is not possible to build a debug version of a user library and use it with Dosimis-3. So it is necessary to change the build settings of your project, for being able to debug it in a release build. In the project settings dialog, change the C/C++ optimization level to *Disable (debug)*. In the *Link* tab enable *Generate debug info*.

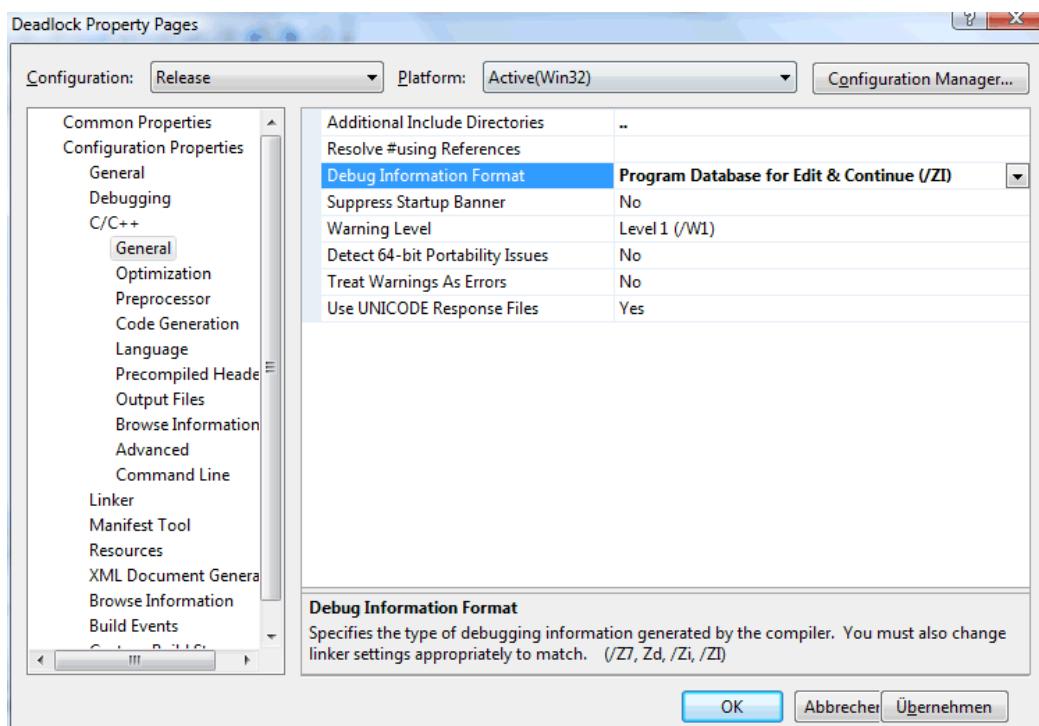


Figure 31.1: Compiler – Settings (Debug)

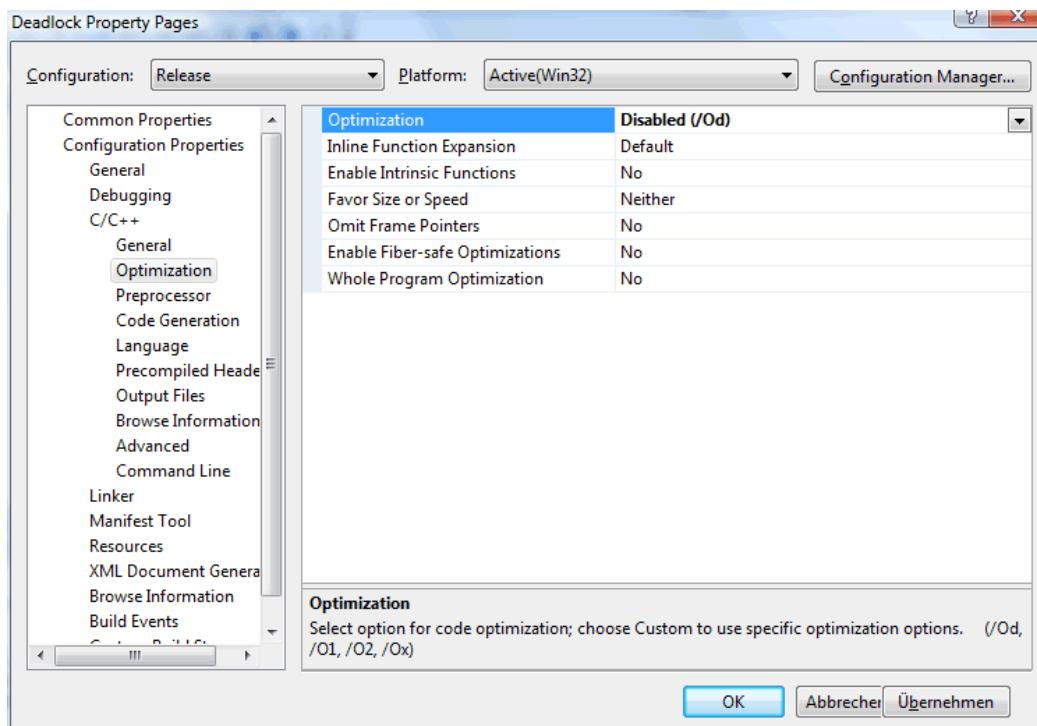


Figure 31.2: Compiler - Settings (Debug)

The linker should generate debug information.

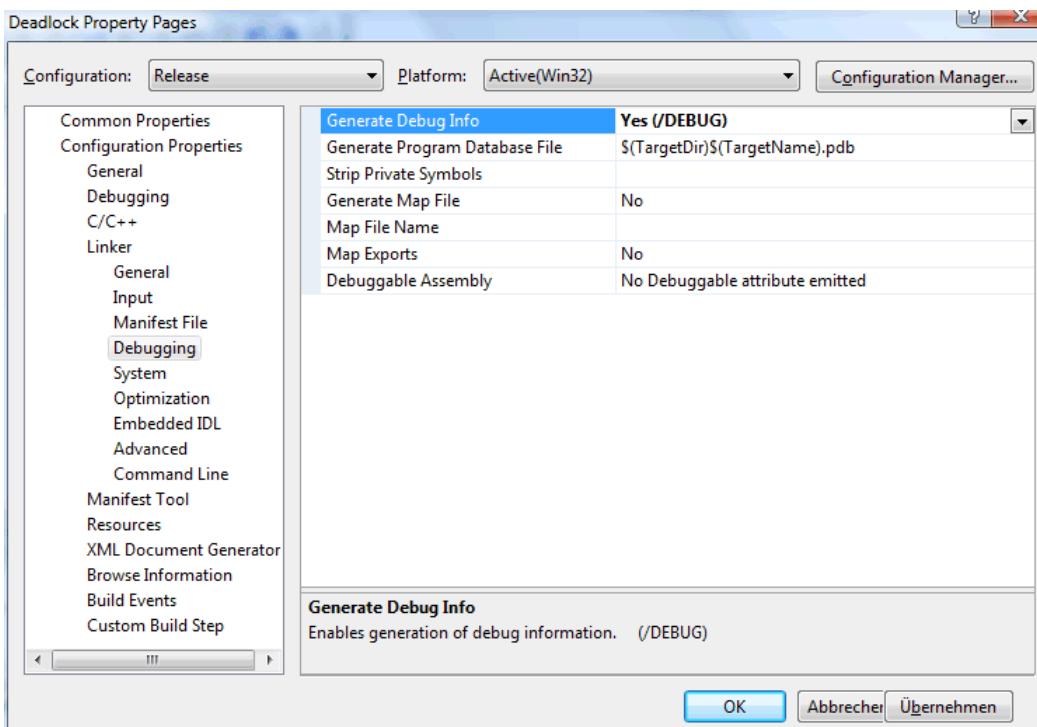


Figure 31.3: Linker - Settings (Debug)



In the debug settings tab, enter the path to the application being debugged. Set it to the Dosimis-3 simulator executable (ds3sim.exe) for debugging the simulator interface or set it to the editor executable (ds3edit.exe) for debugging user-interface functionality.

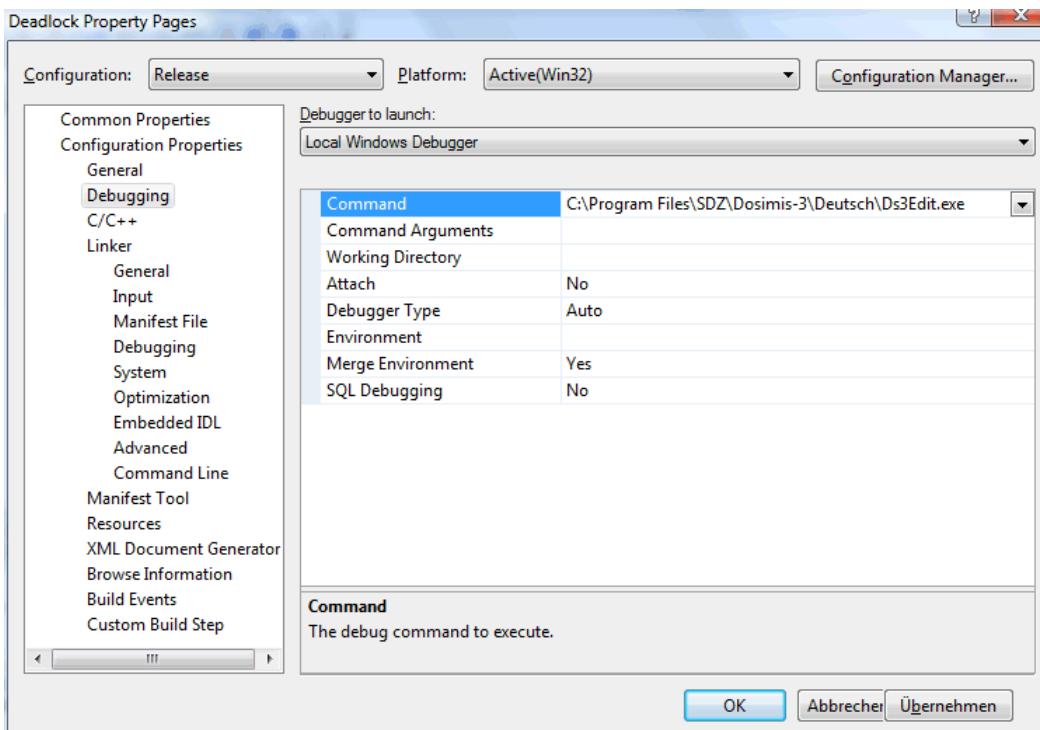


Figure 31.4: Debug - Settings

After starting the debugger (e.g. by pressing F5), the corresponding executable will be launched. The source code of the user library can now be analyzed. Of course it is necessary to have at least one control referencing your library within your model layout.



31.2.3 Debugging using the Visual-C++ - Debugger (Visual Studio 2005)

Advanced users may use the Visual C++ debugger. Visual Studio distinguishes between a *release* and a *debug* build of a project. Proprietary applications like Dosimis-3 are shipped in a release build. But data types differ in the different build types, so it is not possible to build a debug version of a user library and use it with Dosimis-3. So it is necessary to change the build settings of your project, for being able to debug it in a release build. In the project settings dialog, change the C/C++ optimization level to *Disable (debug)*. In the *Link* tab enable *Generate debug info*.

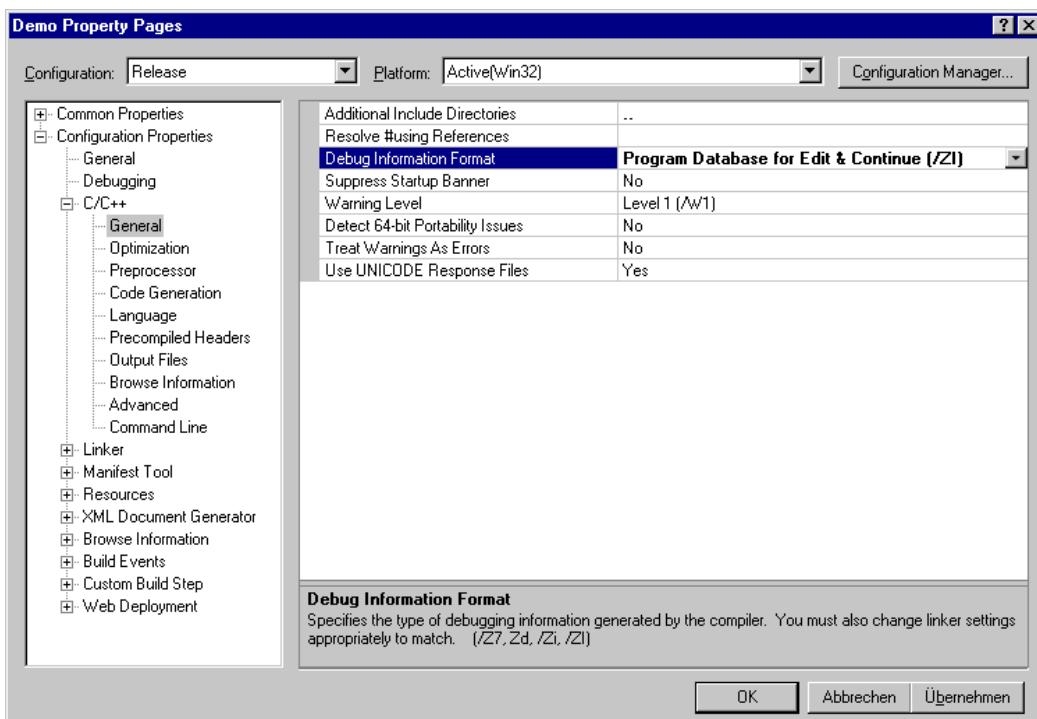


Figure 31.5: Compiler - settings (Debug)

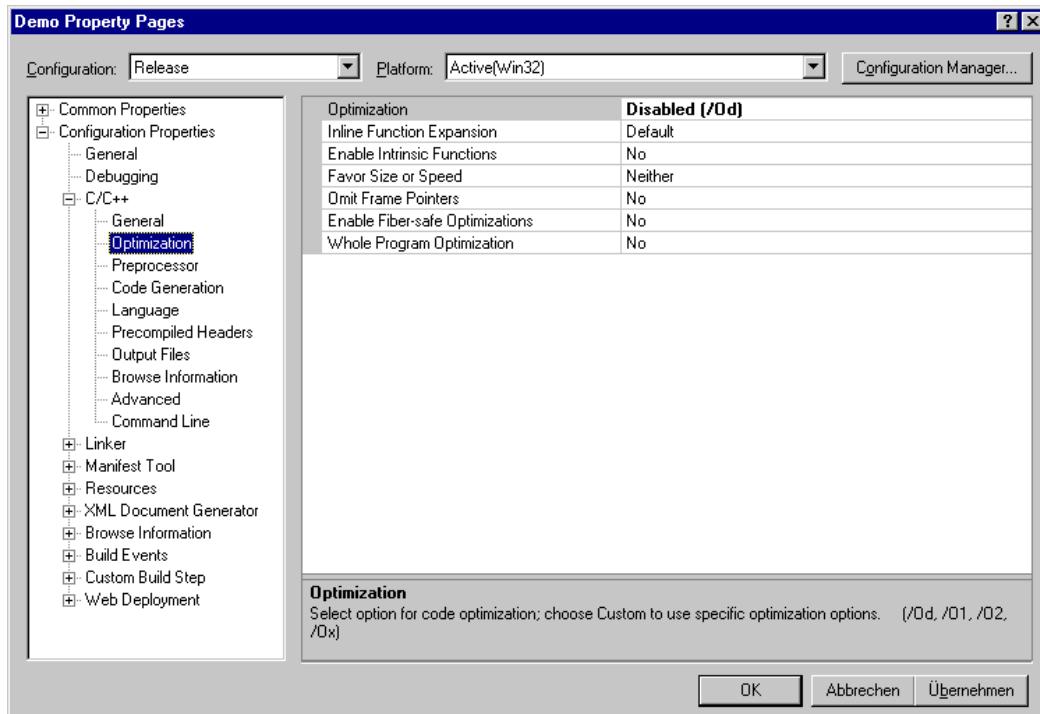


Figure 31.6: Compiler - settings (Debug optimization)

The linker should generate debug information.

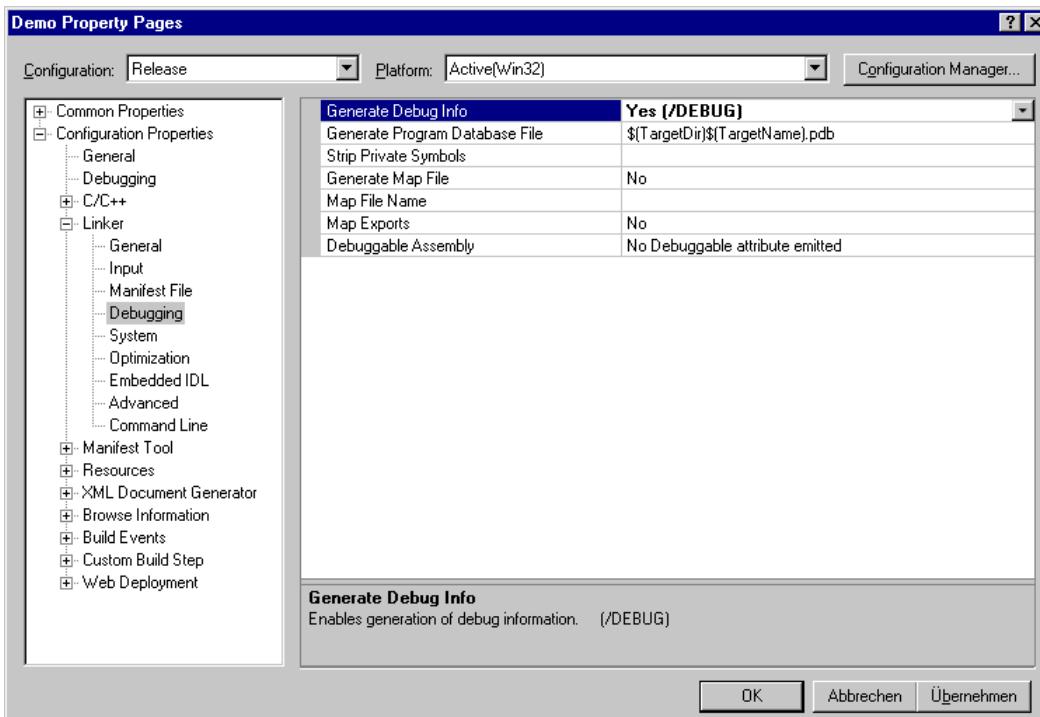


Figure 31.7: Linker - settings (Debug linker)



In the debug settings tab, enter the path to the application being debugged. Set it to the Dosimis-3 simulator executable (ds3sim.exe) for debugging the simulator interface or set it to the editor executable (ds3edit.exe) for debugging user-interface functionality.

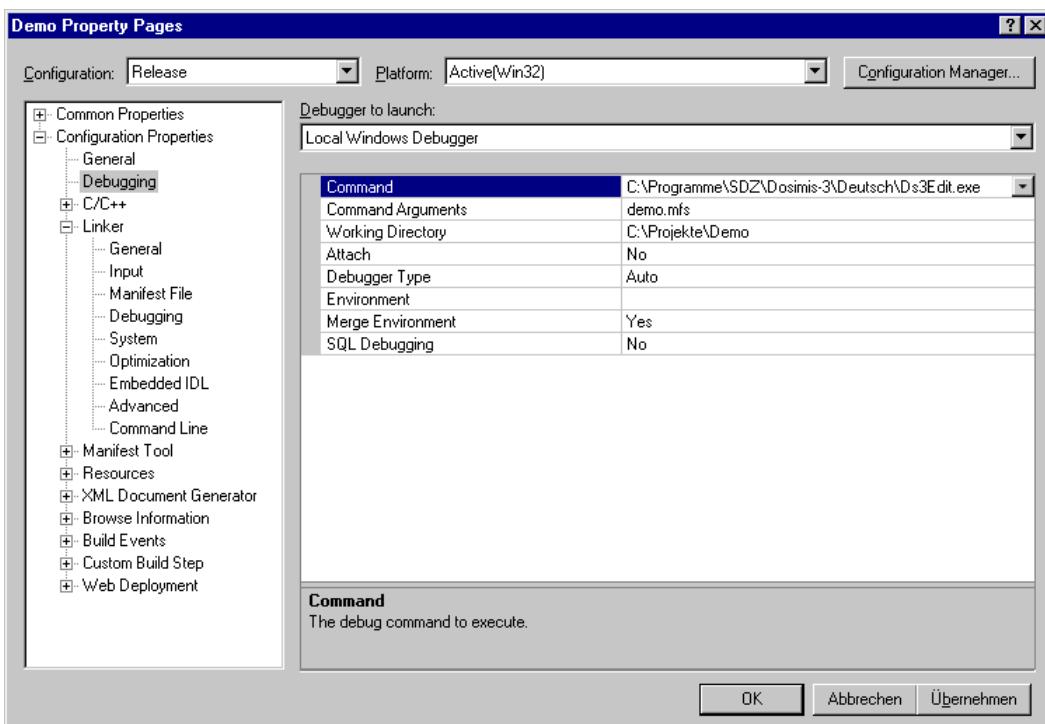


Figure 31.8: Debug - Settings

After starting the debugger (e.g. by pressing F5), the corresponding executable will be launched. The source code of the user library can now be analyzed. Of course it is necessary to have at least one control referencing your library within your model layout.

31.2.4 Debugging using the Visual Studio Debugger (Visual Studio 6.0)

Visual Studio 6 is not supported anymore!

Advanced users may use the Visual C++ (version 6.0) debugger. Visual Studio distinguishes between a *release* and a *debug* build of a project. Proprietary applications like Dosimis-3 are shipped in a release build. But data types differ in the different build types, so it is not possible to build a debug version of a user library and use it with Dosimis-3. So it is necessary to change the build settings of your project, for being able to debug it in a release build. In the project settings dialog, change the C/C++ optimization level to *Disable (debug)*. In the *Link* tab enable *Generate debug info*.

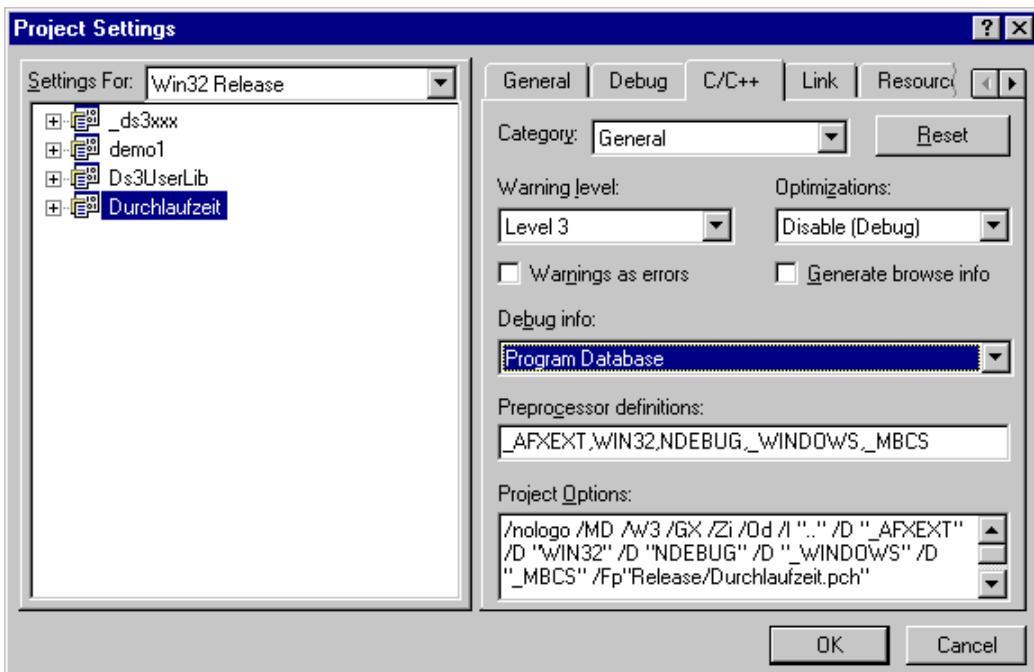


Figure 31.9: Compiler settings (debug)

The linker should generate debug information.

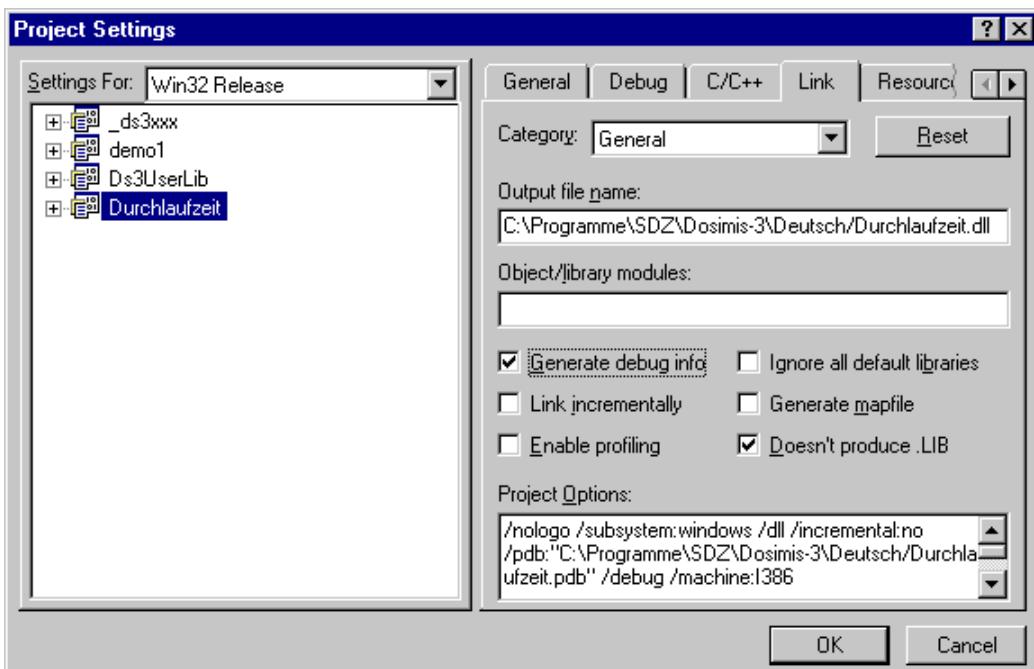


Figure 31.10: Linker settings (debug)

In the debug settings tab, enter the path to the application being debugged. Set it to the Dosimis-3 simulator executable (ds3sim.exe) for debugging the simulator interface or set it to the editor executable (ds3edit.exe) for debugging user-interface functionality.

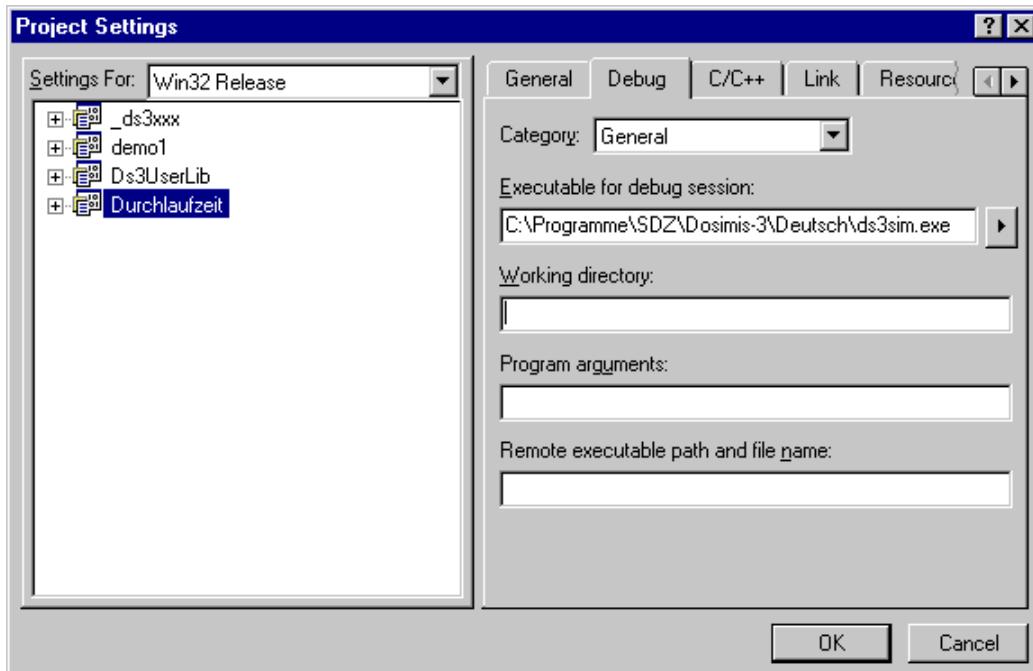


Figure 31.11: Debug settings

After starting the debugger (e.g. by pressing F5), the corresponding executable will be launched. The source code of the user library can now be analyzed. Of course it is necessary to have at least one control referencing your library within your model layout.

31.2.5 Using the Dosimis-3 Data Structures

Debugging the Implementation

The data types of the Dosimis-3 interface library consist of a type name and an internal structure. The structure is not accessible, which makes all Dosimis-3 specific data types anonymous while debugging. For analyzing the data structures of Dosimis-3 data types, the corresponding C++ files are shipped with Dosimis-3. By putting them to the project directory, they get accessible by the debugger. If you want to use the data structure with Visual Studio 6.0, you need a different version of Dosimis-3. Please contact your dealer.

Using the Structures

The data structures should only be accessed by the given program interface. But there is a way to access them directly, anyhow. All data types are headed by the macro *PROTECTED*. When accessing the structure directly, this causes an error message:

```
DurchlaufzeitSim.cpp(34) : error C2248: 'nr' : cannot access protected member declared in
class 'baustein'
    baustein.h(695) : see declaration of 'nr'
```

Reason is the definition of the macro *PROTECTED*, which can be seen in *typen.h*.

```
#define PROTECTED protected:
```

As you see, the word *PROTECTED* is defined to the key word *protected*, which disallows access to class members from outside.



By redefining the macro, this access protection can be disabled. This can be done by defining *PROTECTED* to nothing, explicitly by deleting the word *protected* within *typen.h*.

```
#define PROTECTED
```

31.3 Template Files

31.3.1 Customizing an own Environment

In the *_template* directory all template files for creating new user library projects are found. All files beginning with *_ds3xxx* will be copied to the project directory and get renamed to the project name. If you change the template files, your changes will be taken to the code body of your next generated user library project.

For example, certain project settings (i.e. the path to Dosimis-3 or generating debug information) can be done once and are preset each time you create a new project.

You should **never** change the base name *ds3xxx* of the template files. The project creation process cannot work properly then.

31.4 Directory Structure

31.4.1 Project Directory

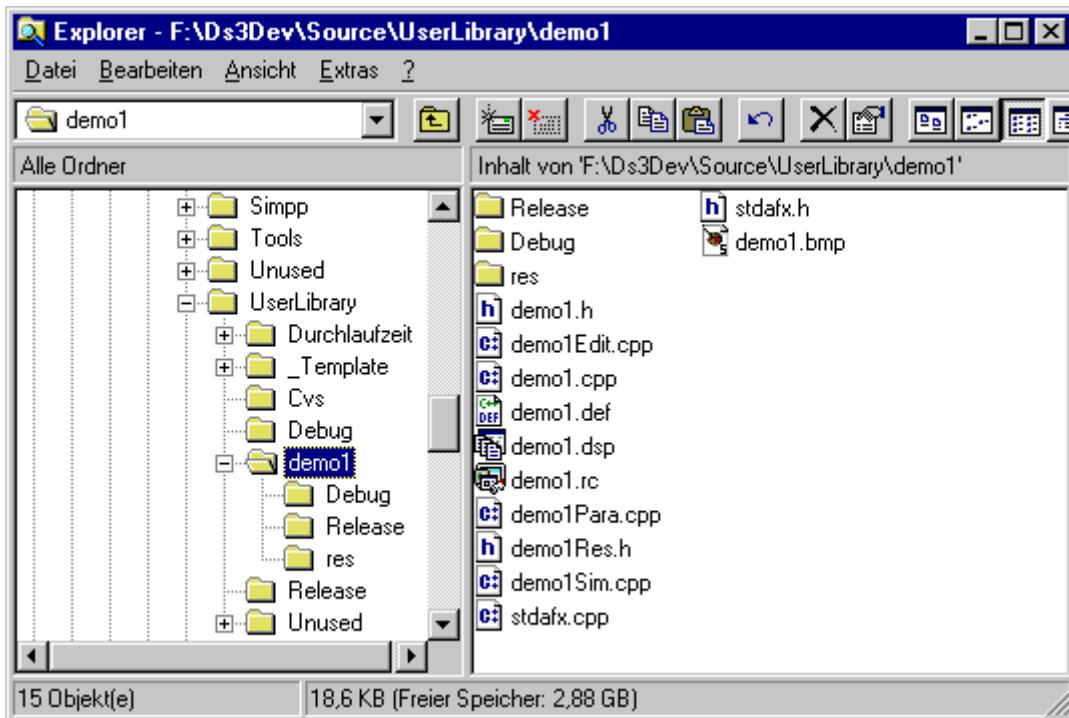


Figure 31.12: Project directory

DEMO1.DSP	The generated project file.
DEMO1.APS, DEMO1.CLW und DEMO1.PLG	Temporary Visual Studio files



DEMO1.DEF	Definition file
DEMO1.CPP	Environment file for the target DLL
DEMO1.RC	Resource file, containing the parameter dialog and icons
DEMO1RES.H	Header file for project resources
DEMO1.H	Header file for the library interface
DEMO1EDIT.CPP	Editor (user interface) functions of the library
DEMO1PARA.CPP	Parameter functions of the library
DEMO1SIM.CPP	Simulator functions of the library
STDAFX.CPP, STDAFX.H	System include files of the program interface

31.4.2 Workspace Directory

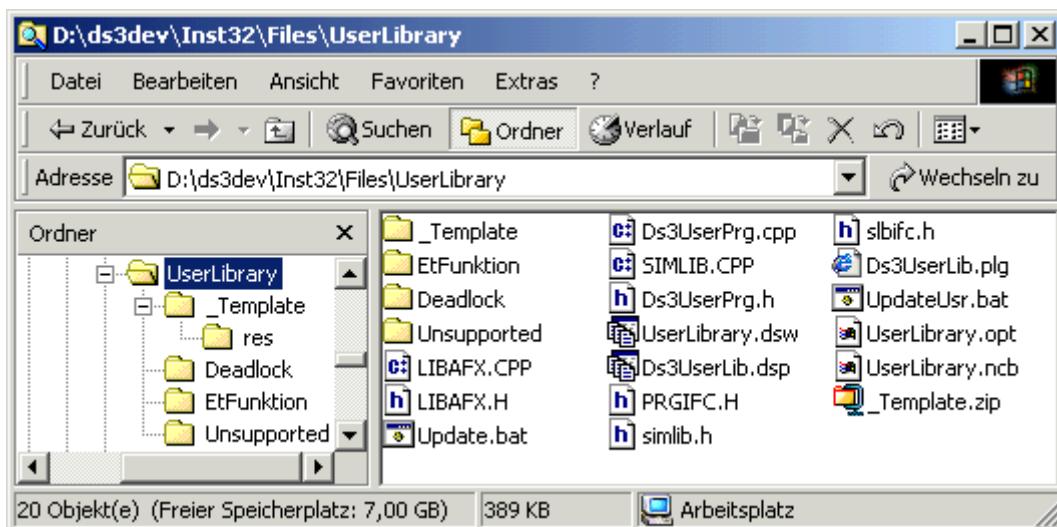


Figure 31.13: Workspace directory

Files:

SIMLIB.CPP, SIMLIB.H und SLBIFC.H	Functionality which can be used in both simulator and user interface
USERLIBRARY.DSW	Workspace file
USERLIBRARY.NCB,	
USERLIBRARY.OPT	Temporary Visual Studio files
Ds3USERLIB.DSP	Project file for the interface. Contains all necessary files for implementing an interface to the editor and simulator and for implementing a base class for further interfaces.
PRGIFC.H, DS3USERPRG.CPP und DS3USERPRG.H	Base class for all program interface libraries
LIBAFX.CPP, LIBAFX.H	Files of the program library

31.4.3 Files of the Program Interface

The settings of the program interface are contained by file named *Ds3Library.ini*. This file is located in the Dosimis-3 executable directory.

31.4.3.1 *Ds3Library.ini*



```

MSD d:/Programme/Microsoft Visual Studio/Common/MSDev98/Bin
SRC F:\Ds3Dev\Source\UserLibrary\
TPL F:\Ds3Dev\Source\UserLibrary\_Template\
DLL Durchlaufzeit
DLL demo1
FIN

```

- MSD Path to Visual Studio
- SRC Path to the project sources
- TPL Path to the template directory
- DLL all accessible interface libraries

31.5 Control Structure

Interfaces, which co-operate with the simulator of Dosimis-3, must consider the different modes of the simulation call during the implementation.

31.5.1 Offline Simulation

In the offline simulation the calculation is started in an own process. This means that all variables are initialized at the beginning. The use of global variables is likewise unproblematic as the structure of complex data structures, because these are released again with the completion of the processes.

31.5.2 Online Simulation

In the online simulation the use of global variables is somewhat more complicated. The computation takes place not during its own process, but within the user interface. This meant that the global variables are initialized only once with the call of the program, not however with each simulation run. For this each interface possesses a function, which is called with beginning of simulation and where these initializations have to be made.

It is also to be noted that new allocated data within the process is not released automatically after the completion of the simulation. This is to be respected with the implementation. The release of that memory is to be made in the functions, which are called at the end of the simulation run. Here it is also to be proofed, that no "dirt" is released in the variables. These should be set back at the end of the simulation, so with the next simulation run no values are accessed, which originate still from the previous computation.

31.5.3 Multi Threading

In principle it is possible in Dosimis-3 to encase extensions in own threads. It is however to be noted that functions from the simulation library cannot be called. This calls are possible only in the working routines of the interface. So results from these thread functions must be buffered, in order to use them with the next call of a working routine. The functions of the interface library are not multithread enabled in the interest of a fast processing.





32 Creating a new User Library

32.1 Description (Visual Studio 2008 / 2010)

32.1.1 Step 1

A new user library can be created using the menu **Programming/New/...**. First it is to be selected, what kind of element is to be created. There are templates available for control, working and crossings. The first one corresponds with a superordinated control like a decision table, the two other templates correspond with the modules workstation or crossing.

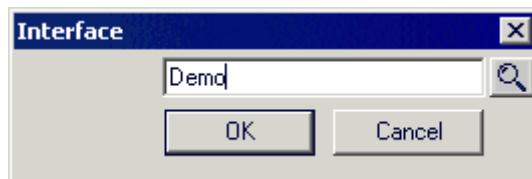


Figure 32.1: Defining a new interface

After that a dialog box appears, asking for a name for the user library. On confirming the dialog, Visual Studio will launch automatically, if it is not already running.

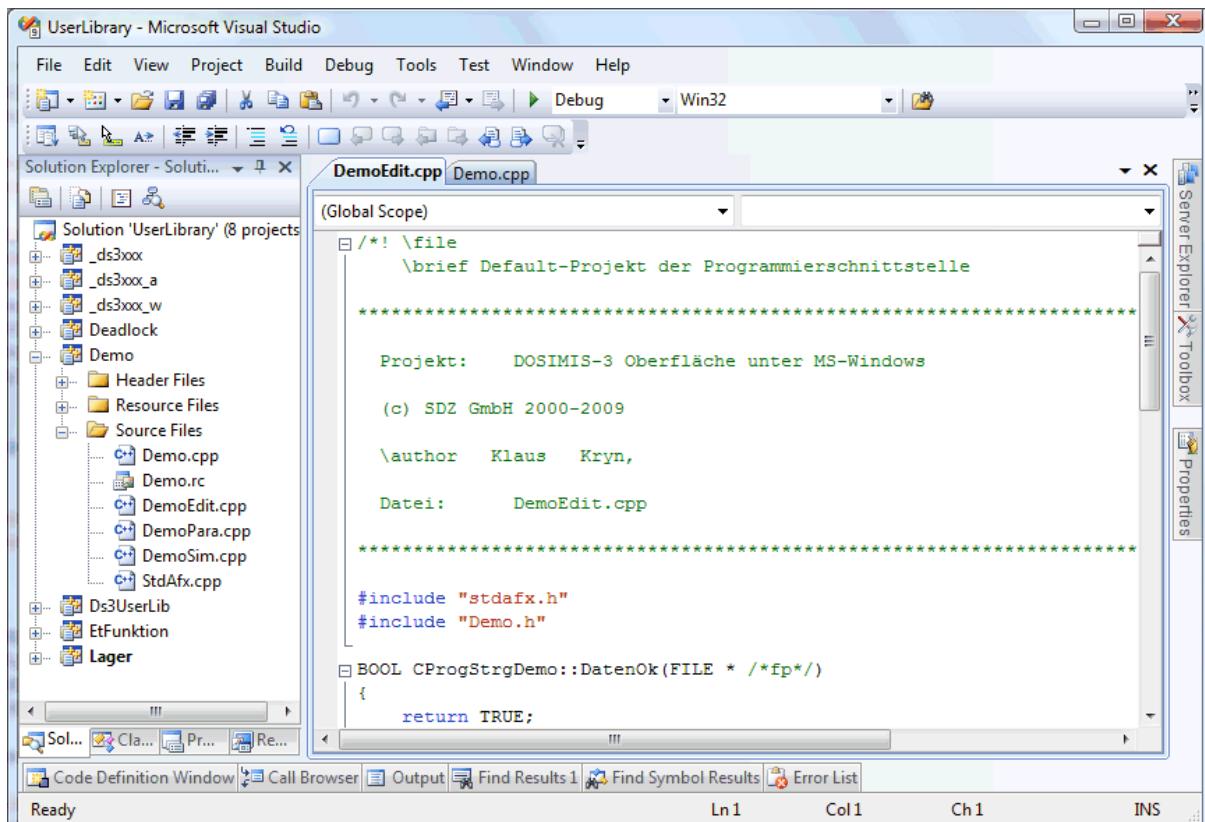


Figure 32.2: Development environment (Visual C++ 2008)



32.1.2 Step 2

When building the new project, the Visual C++ linker gives error messages on missing C++ functions. These are functions from the base class *CProgStrg* and functions from *Simlib*.

```

Output
Show output from: Build
1>Linking...
1> Creating library .....\Ds3Edit\Debug\Demo.lib and object .....\Ds3Edit\Debug\Demo.exp
1>DemoEdit.obj : error LNK2019: unresolved external symbol "protected: void __thiscall CProgStrg::DrawFromBitmap(class CProgStrg&)" (?_DrawFromBitmap@CProgStrg@@PAV0@)
1>DemoPara.obj : error LNK2019: unresolved external symbol "public: __thiscall CProgStrg::CProgStrg(void)" (?__CProgStrg@@QAE@X)
1>DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual long __thiscall CProgStrg::QueryInterface(_GUID const *,void **,long *const)" (?QueryInterface@CProgStrg@@QAEJREFVGUID@@PAXPAX)
1>DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual unsigned long __thiscall CProgStrg::AddRef(void const *)" (?AddRef@CProgStrg@@QAEKX)
1>DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual unsigned long __thiscall CProgStrg::Release(void const *)" (?Release@CProgStrg@@QAEKX)
1>DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual class ISimlib4 * __thiscall CProgStrg::SetEditLib(class ISimlib4 const &)" (?SetEditLib@CProgStrg@@QAEPAVISimlib4@@X)
1>DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual void __thiscall CProgStrg::SetEditLib(class ISimlib4 const &)" (?SetEditLib@CProgStrg@@QAFPAVISimlib4@@X)
1>DemoPara.obj : error LNK2019: unresolved external symbol "long __cdecl GetPrgNr(class IProgStrg *)" (?GetPrgNr@@YAJP@IProgStrg@@)
1>DemoPara.obj : error LNK2019: unresolved external symbol "char const * __cdecl GetPrgBez(class IProgStrg *)" (?GetPrgBez@IProgStrg@@YAPBZ)
1>DemoPara.obj : error LNK2019: unresolved external symbol "void __cdecl SetPrgBez(class IProgStrg *,char const *)" (?SetPrgBez@IProgStrg@@YAFPAV0@Z)
1>..\..\Ds3Edit\Debug\Demo.dll : fatal error LNK1120: 10 unresolved externals
1>Build log was saved at "file:///c:/ds3dev/source/UserLibrary/Demo/Debug/BuildLog.htm"
1>Demo - 11 error(s), 0 warning(s)
=====
Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ======

```

Figure 32.3: Typical link error if no dependencies have been defined

To solve that problem, define a dependency on the library *Ds3UserLib* by using the menu *Project/Dependencies...*

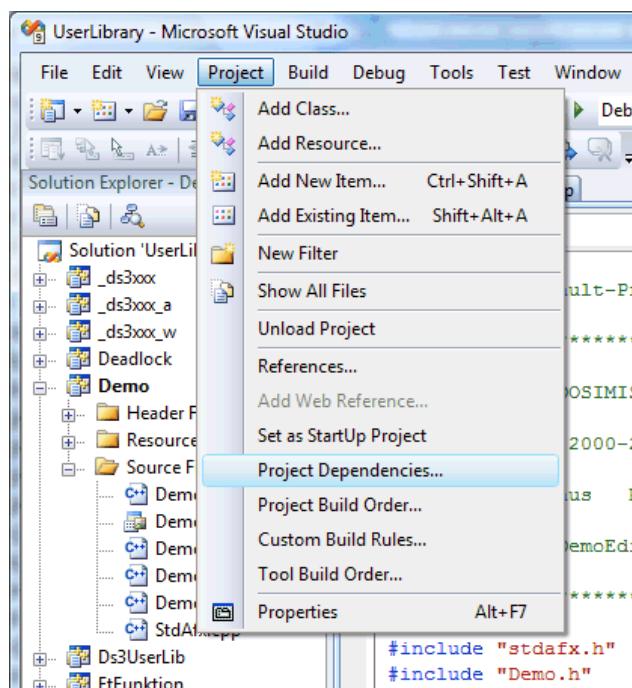


Figure 32.4: Defining the project dependencies



The new project now depends only on Ds3UserLib.

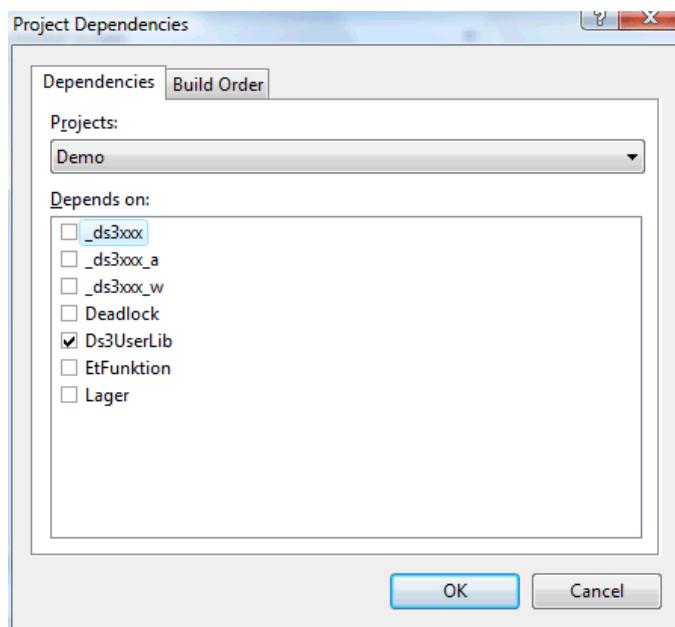


Figure 32.5: Project Dependencies

32.1.3 Step 3

The last step sets the linker output directory to the path where Dosimis-3 was installed to (the directory containing ds3edit.exe). This is done by typing the Dosimis-3 installation path into the field Output Directory in the Project Settings dialog. Open that dialog by selecting the menu Project/Properties

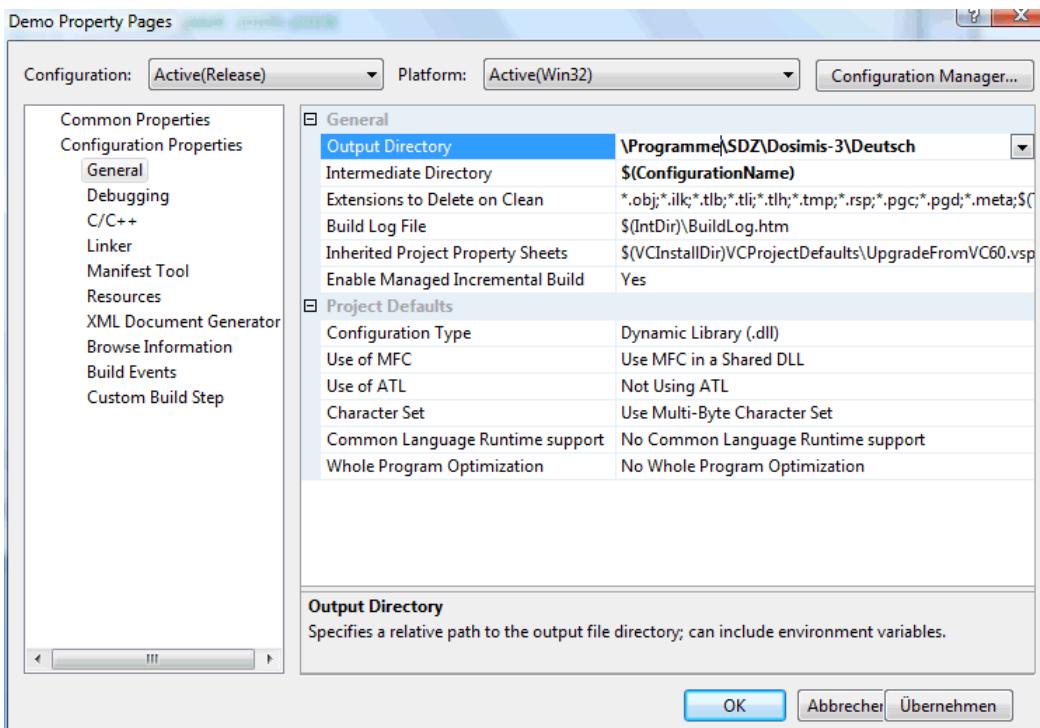


Figure 32.6: Output - Settings



After the first build (simply hit F7 in Visual Studio), the newly created user library can be used in Dosimis-3. Of course it does not contain any special functionality. The template source code delivered with Dosimis-3 just gives a body to implement special behavior.

32.2 Description (Visual Studio 2005)

32.2.1 Step 1

A new user library can be created using the menu **Programming/New/...**. First it is to be selected, what kind of element is to be created. There are templates available for control, working and crossings. The first one corresponds with a superordinated control like a decision table, the two other templates correspond with the modules workstation or crossing.

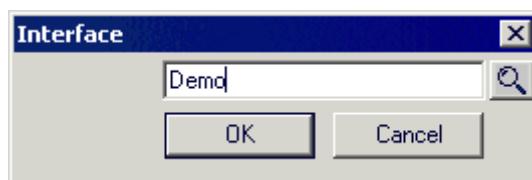


Figure 32.7: Defining a new interface

After that a dialog box appears, asking for a name for the user library. On confirming the dialog, Visual Studio will launch automatically, if it is not already running.

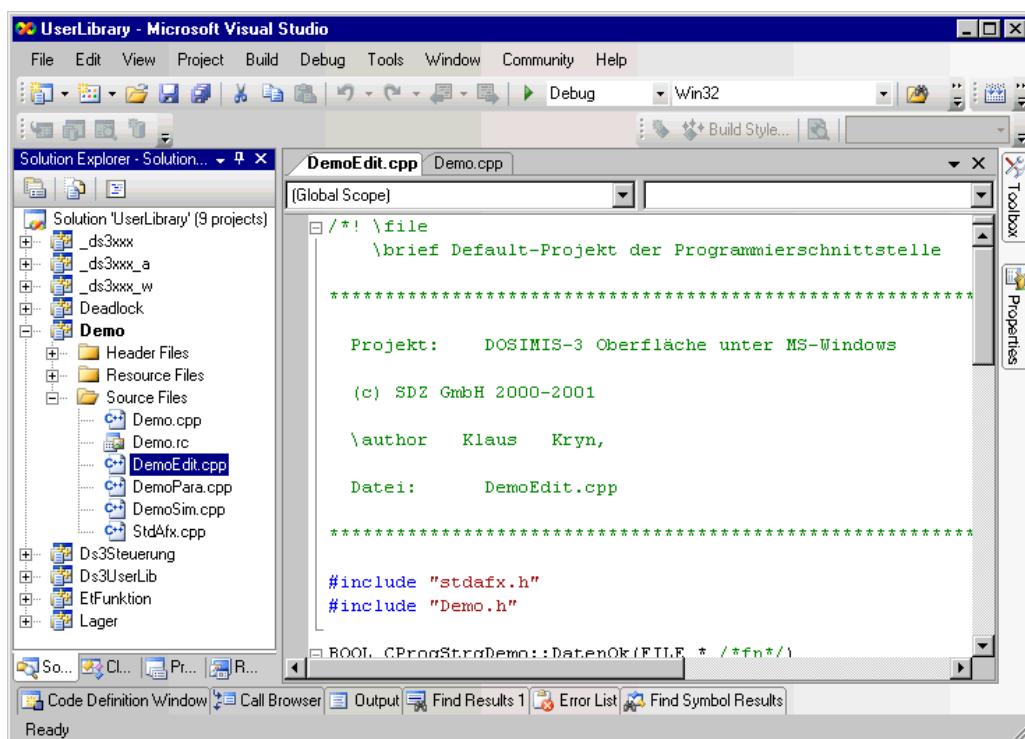


Figure 32.8: Development environment (Visual C++ 2005)



32.2.2 Step 2

When building the new project, the Visual C++ linker gives error messages on missing C++ functions. These are functions from the base class *CProgStrg* and functions from *Simlib*.

```

Output
Show output from: Build
Creating library .\...\...\Ds3Edit\Debug\Demo.lib and object .\...\...\Ds3Edit\Debug\Demo.exp
DemoEdit.obj : error LNK2019: unresolved external symbol "protected: void __thiscall CProgStrg::DrawFromBitmap(class CDC *,int)" (?__OCProgStrg@QAE@I)
DemoPara.obj : error LNK2019: unresolved external symbol "public: __thiscall CProgStrg::CProgStrg(void)" (?__OCProgStrg@QAE@I)
DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual long __thiscall CProgStrg::QueryInterface(struct _GUID const *,void **,long *)" (?QueryInterface@CProgStrg@@UINTERFACE@GUID@@PAX@PAX@_GUID@@PAX@I)
DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual unsigned long __thiscall CProgStrg::AddRef(void)" (?AddRef@CProgStrg@@IAE@I)
DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual unsigned long __thiscall CProgStrg::Release(void)" (?Release@CProgStrg@@IAE@I)
DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual class ISimlib4 * __thiscall CProgStrg::SetSimlib(class ISimlib4*)" (?SetSimlib@CProgStrg@@IAE@A)
DemoPara.obj : error LNK2001: unresolved external symbol "public: virtual void __thiscall CProgStrg::SetEditlib(class CDs3Doc *)" (?SetEditlib@CProgStrg@@IAE@A)
DemoPara.obj : error LNK2019: unresolved external symbol "long __cdecl GetPrgNr(class IProgStrg *)" (?GetPrgNr@IProgStrg@@YAJPAVIProgS@I)
DemoPara.obj : error LNK2019: unresolved external symbol "char const * __cdecl GetPrgBez(class IProgStrg *)" (?GetPrgBez@IProgStrg@@YAPBDA@I)
DemoPara.obj : error LNK2019: unresolved external symbol "void __cdecl SetPrgBez(class IProgStrg *,char const *)" (?SetPrgBez@IProgStrg@@YAPVSetPrgBez@IProgStrg@@PBD@I)
.\...\...\Ds3Edit\Debug\Demo.dll : fatal error LNK1120: 10 unresolved externals
Creating browse information file...
Microsoft Browse Information Maintenance Utility Version 8.00.50727
Copyright (C) Microsoft Corporation. All rights reserved.
Build log was saved at "file:///c:/ds3dev/source/UserLibrary\Demo\Debug\BuildLog.htm"
Demo - 11 error(s), 0 warning(s)
===== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped =====

```

Figure 32.9: Typical link error if no dependencies have been defined

To solve that problem, define a dependency on the library *Ds3UserLib* by using the menu *Project/Dependencies...*

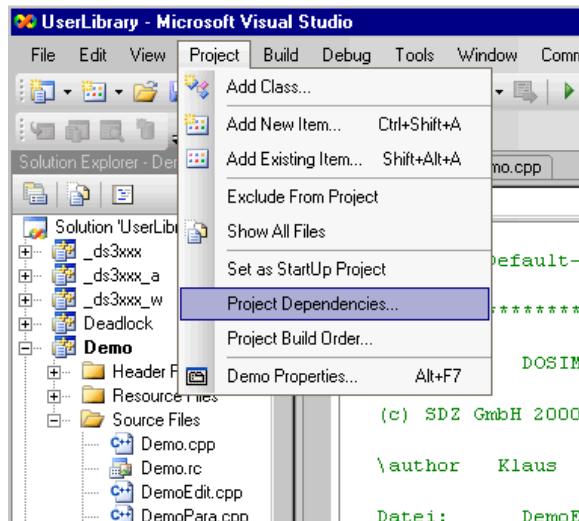


Figure 32.10: Defining the project dependencies



The new project now depends only on Ds3UserLib.

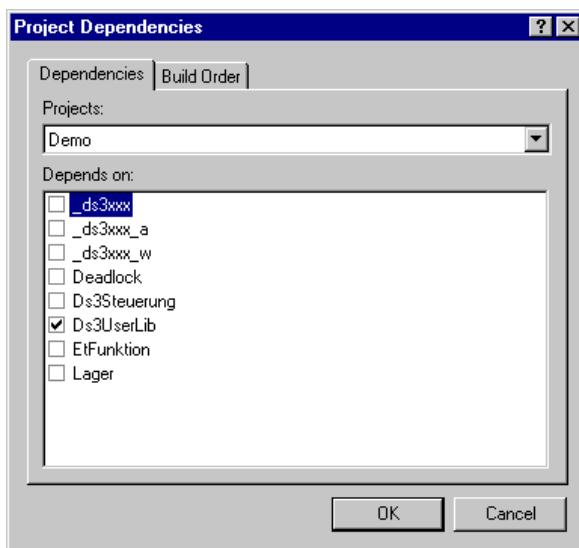


Figure 32.11: Project - Dependencies

32.2.3 Step 3

The last step sets the linker output directory to the path where Dosimis-3 was installed to (the directory containing *ds3edit.exe*). This is done by typing the Dosimis-3 installation path into the field Output Directory in the Project Settings dialog. Open that dialog by selecting the menu Project/Properties

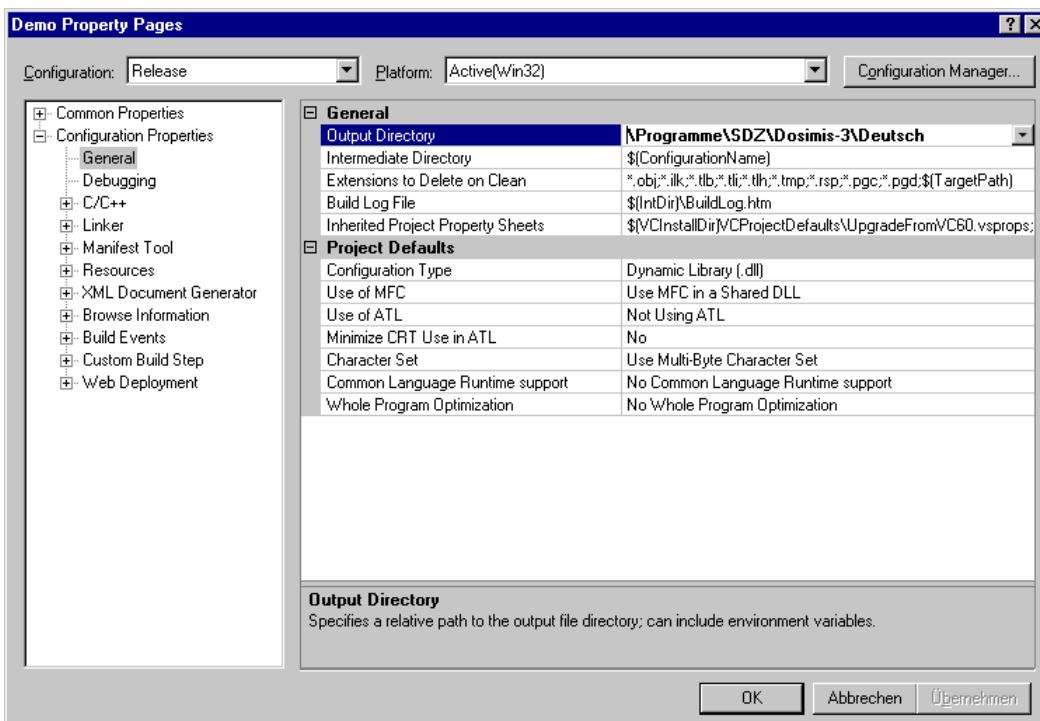


Figure 32.12: Output - Configuration

After the first build (simply hit F7 in Visual Studio), the newly created user library can be used in Dosimis-3. Of course it does not contain any special functionality. The template source code delivered with Dosimis-3 just gives a body to implement special behavior.



32.3 Description (Visual Studio 6.0)

Visual Studio 6 is not supported anymore!

32.3.1 Step 1

A new user library can be created using the menu **Programming/New/...**. First it is to be selected, what kind of element is to be created. There are templates available for control, working and crossings. The first one corresponds with a superordinated control like a decision table, the two other templates correspond with the modules workstation or crossing.

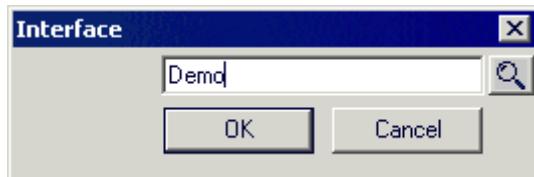


Figure 32.13: Define a new user library

After that a dialog box appears, asking for a name for the user library. On confirming the dialog, Visual Studio will launch automatically, if it is not already running.

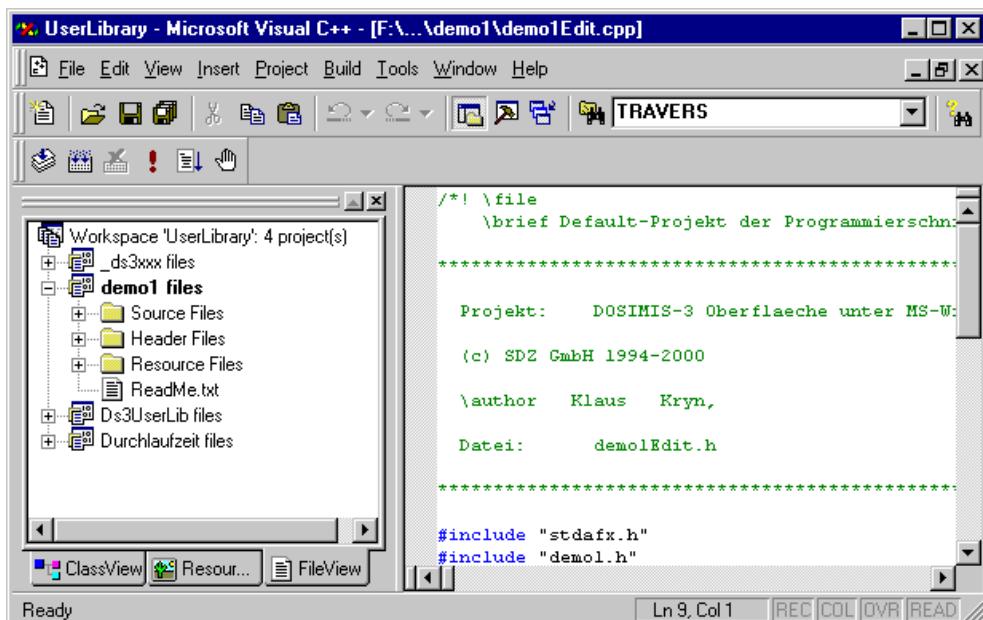


Figure 32.14: Development environment Visual Studio



32.3.2 Step 2

When building the new project, the Visual C++ linker gives error messages on missing C++ functions. These are functions from the base class CPProgStrg and functions from Simlib.

```

Linking...
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
/subsystem:windows /dll /incremental:yes "/pdb:\ds3dev\source\ds3edit\Debug\demo1.pdb" /debug /machine:I386 "/def:.\.
.\Debug\demo1.obj
.\Debug\demo1Edit.obj
.\Debug\demo1Para.obj
.\Debug\demo1Sim.obj
.\Debug\StdAfx.obj
.\Debug\demo1.res
Creating library \ds3dev\source\ds3edit\Debug\demo1.lib and object \ds3dev\source\ds3edit\Debug\demo1.exp
demo1Edit.obj : error LNK2001: unresolved external symbol "protected: void __thiscall CPProgStrg::DrawFromBitmap(class demo1Para obj)" [class demo1Para.obj]
LNK2001: unresolved external symbol "public: virtual void __thiscall CPProgStrg::SetEditlib(class demo1Para obj)" [class demo1Para.obj]
LNK2001: unresolved external symbol "public: virtual void __thiscall CPProgStrg::SetSimlib(class demo1Para obj)" [class demo1Para.obj]
LNK2001: unresolved external symbol "public: virtual unsigned long __thiscall CPProgStrg::AddRef(class demo1Para obj)" [class demo1Para.obj]
LNK2001: unresolved external symbol "public: virtual unsigned long __thiscall CPProgStrg::Release(class demo1Para obj)" [class demo1Para.obj]
LNK2001: unresolved external symbol "public: virtual long __thiscall CPProgStrg::QueryInterface(class demo1Para obj)" [class demo1Para.obj]
LNK2001: unresolved external symbol "public: __thiscall CPProgStrg::CPProgStrg(void)" (?0CPProgStrg@?0CPProgStrg@@C)
Error executing link.exe.

```

Figure 32.15: Missing interface functions

To solve that problem, define a dependency on the library *Ds3UserLib* by using the menu *Project/Dependencies...*

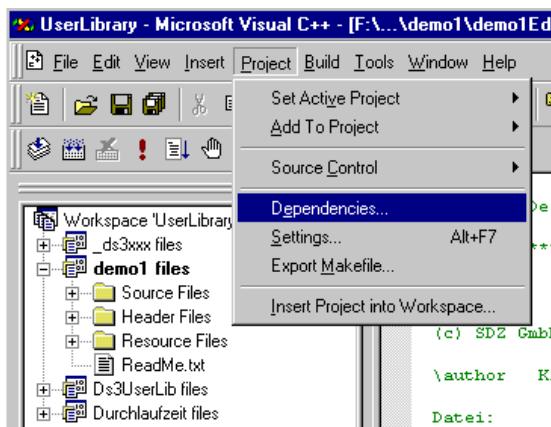


Figure 32.16: Defining dependencies

The new project now depends on that library and so is able to use its classes and functions.

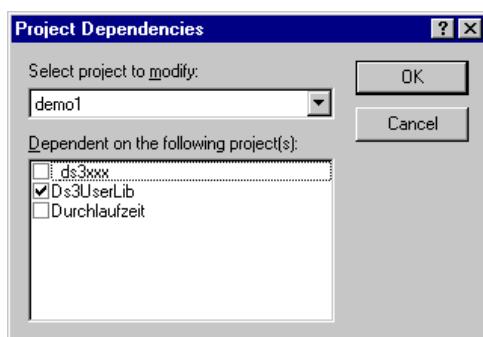


Figure 32.17: Project dependency



32.3.3 Step 3

The last step sets the linker output directory to the path where Dosimis-3 was installed to (the directory containing ds3edit.exe). This is done by typing the Dosimis-3 installation path into the field Output files in the Project Settings dialog. Open that dialog by selecting the menu Project/Settings...

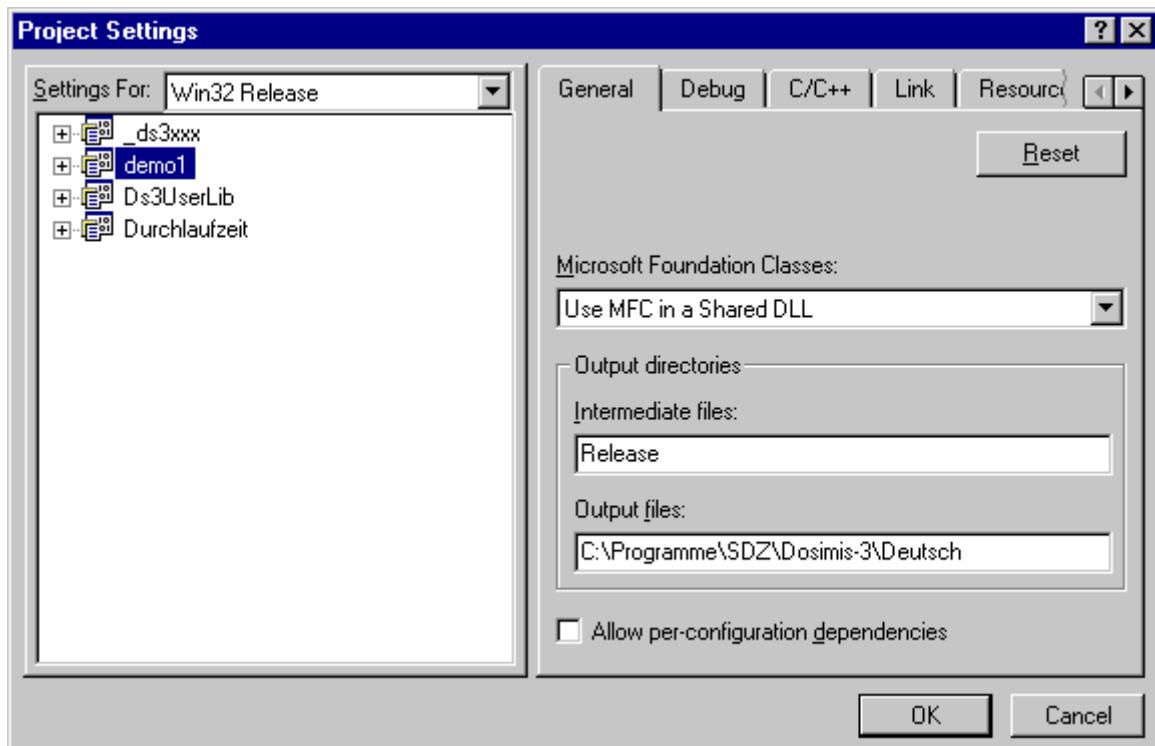


Figure 32.18: Output settings

After the first build (simply hit F7 in Visual Studio), the newly created user library can be used in Dosimis-3. Of course it does not contain any special functionality. The template source code delivered with Dosimis-3 just gives a body to implement special behavior.

32.4 Placing new Controls in the Model Layout

Select the type of element you want to place in the model. Get the actual version of the toolbar by using the menu *Programming/Create Palette*.



Figure 32.19: Interface selection



Alternative a control of type **PRG** can be selected from the control palette.

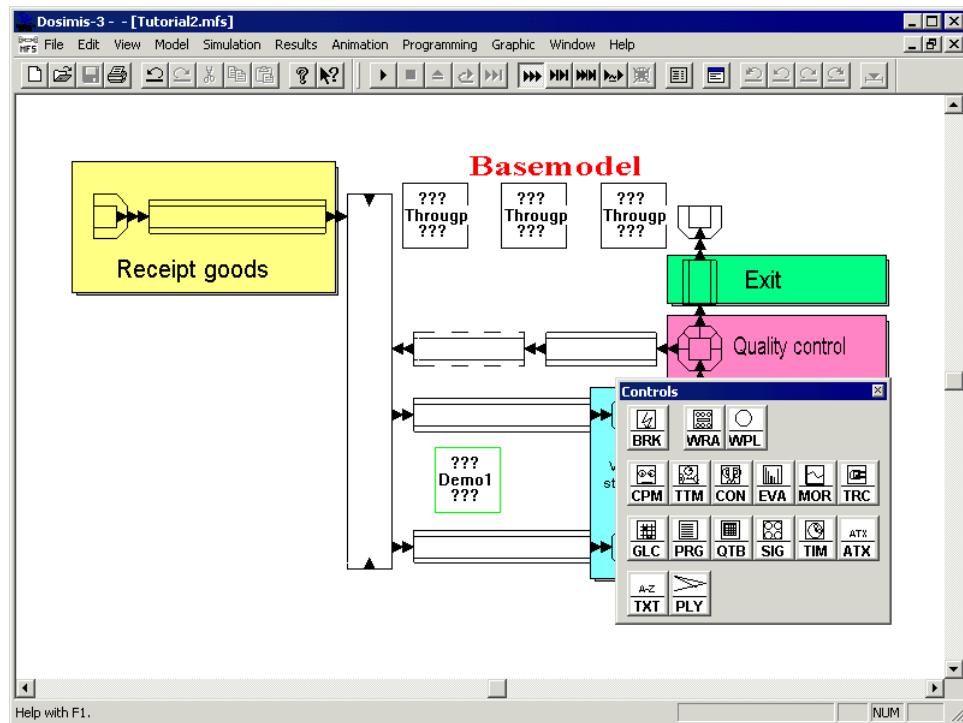


Figure 32.20: Selecting a custom control from the palette

In this case you have to chose which user library serves that control.

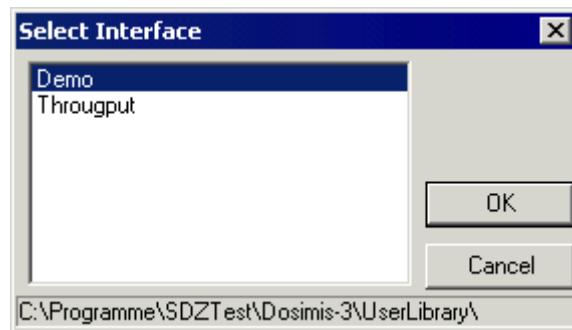


Figure 32.21: Choosing a user library



Then the control can be placed into the model.

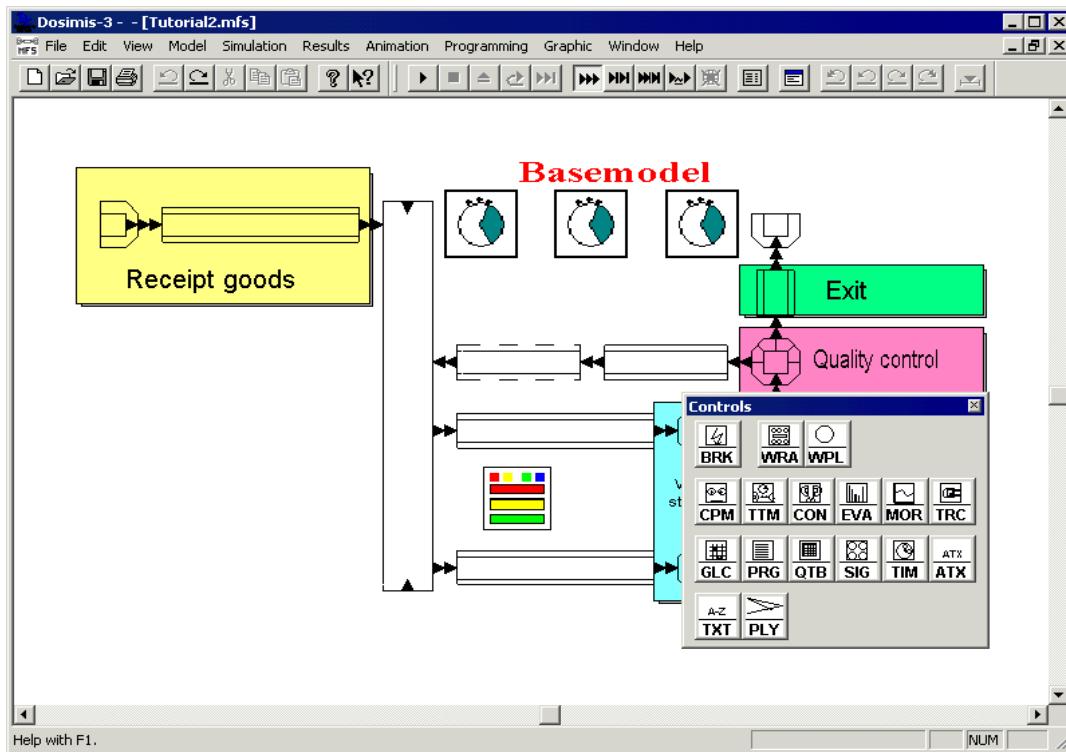


Figure 32.22: Placing a new custom control

Placed controls can be assigned to modules and junctions. This is done, as with all controls, by changing to the *linking active* mode and selecting the corresponding modules and junctions.

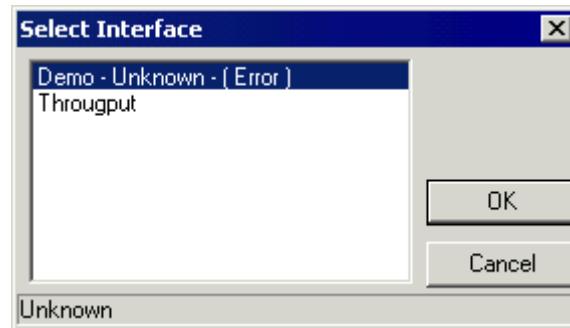


Figure 32.23: Display of missing user libraries as unknown

Assigned user libraries, which do not exist (missing DLL file) are marked as *unknown* in the selection list.



33 Interface for Controls

33.1 API Functions

The newly created class implements the following functions:

```
class CProgStrgDemo1 : public CProgStrg {  
    friend HRESULT CreateProgStrg(IProgStrg ** result);  
protected:  
    CProgStrgDemo1();  
    virtual ~CProgStrgDemo1();  
  
    // User Interface  
    virtual HRESULT Draw(CDC* pDC, int px, int py);  
    virtual BOOL DatenOk(FILE * of = NULL);  
  
    // User Interface Parameters  
    virtual HRESULT Parameter(BOOL * change);  
    virtual BOOL Lesen(const char * key, const char * zeile);  
    virtual BOOL Schreiben(const FILE * fp);  
  
    // Simulator  
    virtual void SimInitialisierung(int step);  
    virtual void SimBearbeitung(baustein * bs, long warum);  
    virtual void SimStatistik(long aktz, long art);  
    virtual void SimBeenden(int step);  
  
    // User specific data follows  
public:  
};
```

They are:

CProgStrgDemo1

The constructor: It will be called each time a custom control served by this user library is placed into the model or if the user library is reloaded.

~CProgStrgDemo1()

The destructor: It will be called before the user library is unloaded, when a control gets removed or when the model is closed.



33.1.1 User Interface

- Draw(CDC* pDC, int px, int py) This function is called each time the custom control needs to be redrawn. So the developer can give an own symbol to controls served by this user library.
- DatenOk(FILE * of = NULL) This function can implement a consistency check for a custom control. Error messages can be printed by using the file *of*. Attention: *of* is not asserted to be a valid file. It might be NULL.

The implementation can be found in *NAMEEdit.cpp*

33.1.2 Parameter

- Parameter(BOOL * change) If the user library has own parameters, it can open a dialog by using this function. **change* should be set to *TRUE* if parameters have changed (e.g. if the dialog was confirmed by clicking OK).
- Schreiben(const FILE * fp) Parameters of a custom control can be written to the MFS-file of the model. This function gets called when the MFS-file is written - one time for each control which is served by this library. You should use the function *ProgSchreiben* for writing data to the MFS-file. *ProgSchreiben* needs the parameters (*const FILE * fp*, *const char * key*, *const char * v*). Key is a unique three-character symbol specifying the parameter to write. E.g. you could use MXT, if your library has a parameter *maximum throughput*. The following write functions are declared in the base class:

```
BOOL CProgStrg::ProgSchreiben(const FILE * fp, const char * key, const char * v)
BOOL CProgStrg::ProgSchreiben(const FILE * fp, const char * key, const int v)
BOOL CProgStrg::ProgSchreiben(const FILE * fp, const char * key, const long v)
BOOL CProgStrg::ProgSchreiben(const FILE * fp, const char * key, const float v)
BOOL CProgStrg::ProgSchreiben(const FILE * fp, const char * key, const double v)
```

All those functions format the value to a string and call

```
BOOL ProgSchreiben(const FILE * fp, const char * key, const char * v)
```

Using this function, it is possible to write complex data.

- Lesen(const char * key, const char * v) The opposite to Schreiben(...). *Lesen* is called each time an MFS-Model is to be read - but once for each parameter which was written to the file. *Lesen* has to evaluate the key and has to set the according parameter with the value contained in *v*. Attention: The value will always be a *char **. It has to be converted to the correct type according to the parameter.

The implementation is found in *NAMEPara.cpp*



33.1.3 Simulator

SimInitialisierung(int step)	This function is called when starting a simulation for every instance of your library - which means it is called for each control served by the library. The initialization takes place in 4 steps <ol style="list-style-type: none"> 1. initialization of table. 2. initialization of modules. 3. Initial actions of decision tables. 4. initialization of objects in the modules.
SimBearbeitung(baustein * bs, long warum)	* This function is called each time an event occurs on one of the linked modules. The parameter <i>bs</i> points to the module where the event occurred. The parameter <i>warum</i> carries the event type.
SimStatistik(long aktz, long art)	This function is called, when statistic data is written. <i>aktz</i> is the time used in the statistics, <i>art</i> will be 1, for an interim-statistic and 2 for an interval-statistic to be written.
SimBeenden(int step)	This function is called for each control, when the simulation ends. The finalization takes place in several steps: <ol style="list-style-type: none"> 1. finalization of controls. 2. finalization of modules.

The implementations are found in *NAMESim.cpp*

33.1.4 Functions of the Simulator Library

Next to the functions of *CProgStrg::** are the functions of the simulator library. For example when in need for accessing the modules linked to your library, the simulator library (*simlib.h*) contains the needed functions. Some of them will be documented here.

```
const char * GetPrgBez(IProgStrg * prg);
long        GetPrgNr(IProgStrg * prg);
bs_liste   * GetPrgBsListe(IProgStrg * prg);

bs_liste * GetBslSuc(bs_liste * bsl);
bs_liste * GetBslPre(bs_liste * bsl);
baustein * GetBslBaustein(bs_liste * bsl);

knoten * GetGknKnoten(gsz_kn * knl);
gsz_kn * GetGknPre(gsz_kn * knl);
gsz_kn * GetGknSuc(gsz_kn * knl);

const char * GetBsBez(baustein * bs);
const char * GetBsKommentar(baustein * bs);
```

With these functions you can iterate over Dosimis-3 modules and junctions. See the following source code for an example how to analyze module names.

```
for (bs_liste * bsl = GetPrgBsListe(this); bsl != NULL; bsl = GetBslSuc(bsl)) {
    baustein * bs = GetBslBaustein (bsl);
    const char * name = GetBsBez(bs);
    /* Analyse the string >name< */
    :
}
```

In analogy to decision tables, controls are having activation modules, which are modules linked to the custom control while in *link-active mode*. If there was an event in one of those modules, the user library serving the control is activated. Reference modules, as they exist in decision tables, are not used in the program interface. Access to modules which are not activation modules can be achieved by the module name. Special search functions are



implemented in the simulator library for that purpose. For efficiency reasons this should be done one single time in the function *SimInitialisierung*. Save the return value of the seek function in a local variable. But do not reuse this stored value when implementing user-interface functionality. This can cause memory access violations, as the desired module could have been deleted or renamed.

```
baustein * suche_bs_nach_bez(const char * bez);
baustein * suche_bs_nach_bez_i(const char * bez);
baustein * suche_bs_nach_kommentar(const char * bez);
baustein * suche_bs_nach_kommentar_i(const char * bez);
```

Using that functions, any module of the loaded model can be found by its name or comment. For further information see the documentation of the simulator library and the reference in the online help system.

33.2 Example

In an independent model the throughput between two distanced modules shall be measured and be written to a file. Independent model means, that the control to be programmed should work with every model. So it is not adjusted to one special situation. For marking the start and target module, the comment of the start module shall begin with S, the comment of the target module will start with T. Whenever an object enters the start module, the actual simulation time has to be recorded within the object itself. When leaving the target module, the recorded time must be read, the difference to the current simulation time must be calculated and written to an output file. The entry time shall be stored in *real.A* of the according object. So this value could also be used by decision tables.

The definition of the control mechanism is quite simple:

*If an object enters the start module, set real.A of the object to the current simulation time.
If an Object leaves the target module, calculate the difference time to real.A, append the result to a file.*

After having created a new user library (see above), a code with basic functionality was created by the library template. This code can now be built to a user library, which can be assigning to a custom control. For giving the user library (and its controls) specific behavior, you have to program certain functionality into the source code. First of all you should give the control an unique symbol.



33.2.1 Symbol

For giving an unique symbol to the control you need to override the *Draw* function of the control class. By default a bitmap is drawn, which is stored as a project resources.

```
HRESULT CProgStrgDurchlaufzeit::Draw(CDC* pDC, int px, int py)
{
    DrawFromBitmap(pDC,px,py, IDB_PROGSTRG) ;
    return NO_ERROR;
}
```

In our example, a stylistic stop watch shall characterize the control. The bitmap can be accessed and edited in the resource editor of Visual Studio.

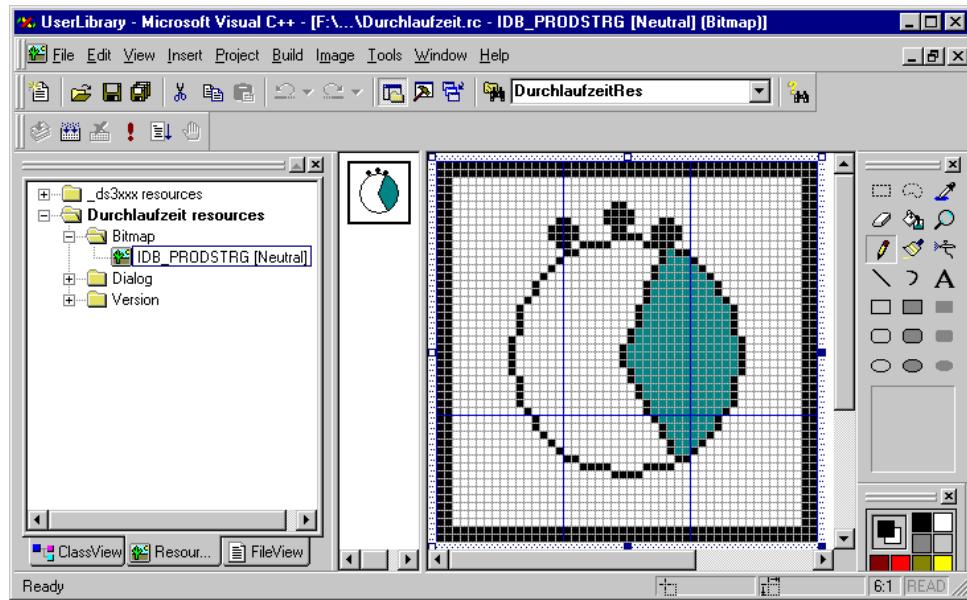


Figure 33.1: Changing a control's symbol



Before compiling the changes to the source code, the library should be unloaded, if it was already used in Dosimis-3. After reloading the rebuilt library, the symbol is visible in Dosimis-3.

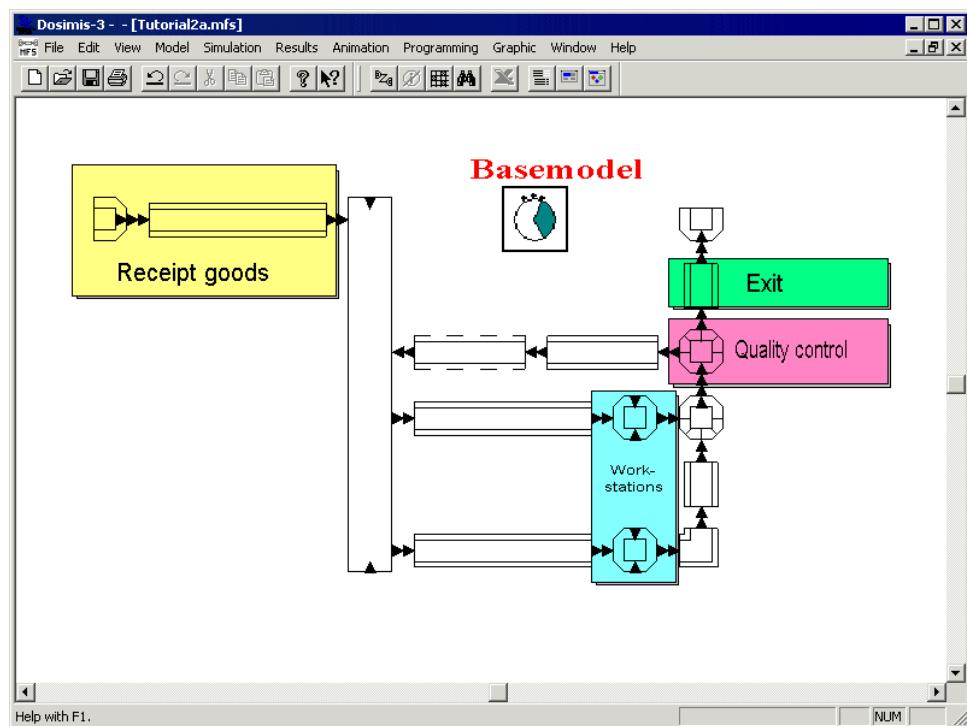


Figure 33.2: First layout

The custom control can be linked, as used with global system states, to Dosimis-3 modules.



33.2.2 Simulator

Now we will take care about the function interface of the simulator.

```
/*! \file
\brief Default-Projekt der Programmierschnittstelle

*****
Projekt: DOSIMIS-3 Oberflaeche unter MS-Windows
(c) SDZ GmbH 1994-2000
\author Klaus Krym,
Datei: DurchlaufzeitSim.cpp
****

#include "stdafx.h"
#include "Durchlaufzeit.h"

// ##### Simulator #####
void CProgStrgDurchlaufzeit::SimInitialisierung(int step)
{
}

void CProgStrgDurchlaufzeit::SimBearbeitung(baustein * bs, long warum)
{
}

void CProgStrgDurchlaufzeit::SimBeenden(int step)
{
}

void CProgStrgDurchlaufzeit::SimStatistik(long zeit, long statty)
{}
```

There are two functions (*SimInitialisierung* and *SimBeenden*) for initializing and destroying the simulation data and model. These functions will be called once each time a simulation run starts (*SimInitialisierung*) or ends (*SimBeenden*). At first we declare a variable of the FILE type and initialize it immediately. FILE variables give access to files on data media like hard disk drives. The connection of the variable to a certain named file is done by the standard C function *fopen*, which needs the file name string as first parameter and a string containing the access method as second. We use “w” for write access - which will create the named file if it does not exist yet. Using the keyword *static* when declaring a variable makes it invisible to other C modules (not to confuse with Dosimis-3 modules). A static variable has the advantage of avoiding name conflicts and raises readability of the program code. When ending the simulation, the file must be closed. This is done by using the C function *fclose*. Closing files is important for flushing cached data to the file. Not closing a file can result in loss of data.



```

// ##### Simulator #####
static FILE * datei = NULL;
void CProgStrgDurchlaufzeit::SimInitialisierung(int step)
{
    if (step == 1) {
        datei = fopen("throughput.txt","w");
    }
}

void CProgStrgDurchlaufzeit::SimBeenden(int step)
{
    if (step == 1) {
        fclose(datei);
    }
}

```

On the next step we will do the main functionality. This happens in *CProgStrgDurchlaufzeit::SimBearbeitung*. This function has two parameters which will be passed to the user library. The first one is the module which was activated by an event. The second says which kind of event occurred. At first we check for object-entry into the start module as recognized by an object entry into an activation module having a comment beginning with 'S'. In that case the current simulation time gets assigned to *real.A* of the entering object. For doing so, several functions of *simlib.h* are needed. We use *GetBsKommentar* (retrieve module comment), *bs_l_obj* (last object in module), *set_obj_et_zus_reell* (set value of additional real-typed object variable), *jetzt* (now - current simulation time). Also we use the constant value *OBJEKTEINFAHRT* (object entry).

```

void CProgStrgDurchlaufzeit::SimBearbeitung(baustein * bs, long warum)
{
    const char * name = GetBsKommentar(bs); // Kommentar des Bausteins
    if (warum == OBJEKTEINFAHRT && name[0] == 'S') {
        objekt * obj = bs_l_obj(bs);
        set_obj_et_zus_reell(obj,'A',jetzt());
    }
}

```

When an object leaves the target module, we have to calculate the time difference to the recorded time and write it to a file. Additionally we append the current simulation time and the object type. The function *fprintf* is a standard C function for writing a formatted output string to a file. The functions *bs_f_obj* (first object in module), *get_obj_et_zus_reell* (get value of additional real-typed object variable) and the constant *OBJEKTAUSFAHRT* (object exit) are, as the functions used before, from *simlib.h*.

```

void CProgStrgDurchlaufzeit::SimBearbeitung(baustein * bs, long warum)
{
    const char * name = GetBsKommentar(bs); // Kommentar des Bausteins
    if (warum == OBJEKTEINFAHRT && name[0] == 'S') {
        objekt * obj = bs_l_obj(bs);
        set_obj_et_zus_reell(obj,'A',jetzt());
    }
    else if (warum == OBJEKTAUSFAHRT && name[0] == 'T') {
        objekt * obj = bs_f_obj(bs);
        double alterzeitpunkt = get_obj_et_zus_reell(obj,'A');
        double aktuellerzeitpunkt = jetzt();
        fprintf(datei,"%d %7.1f %7.1f\n",
                get_obj_typ(obj),
                aktuellerzeitpunkt,
                aktuellerzeitpunkt - alterzeitpunkt);
    }
}

```

After building the user library, start the first simulator run. After that a file *throughput.txt* will exist in the model directory. Viewing it you can see that the implementation is at least plausible.



```

1 314.6 189.7
1 392.1 203.0
1 473.4 225.3
1 559.0 238.5
2 580.9 196.4
1 646.8 203.2
2 774.4 200.7
1 801.7 172.2
2 884.2 198.8
2 968.0 216.3
2 1053.4 240.1
2 1136.5 260.8

```

33.2.3 Consistency Check

Probably the so called consistency check is well known to all Dosimis-3 users. It checks the plausibility of parameters before starting a simulation run. This method can also be implemented for own controls to check their parameters. The function name is *DatenOk* and has a return value of TRUE or FALSE, which tells Dosimis-3 if the control is consistent. In this example we will try if the comments of the linked modules really begin with an ‘S’ or a ‘T’. If the parameter (*FILE *fp*) is initialized (not NULL), an error message can be written to that file. This message will be written to the chk-file, together with all other consistency check messages.

```

BOOL CProgStrgDurchlaufzeit::DatenOk(FILE * fp)
{
    for (bs_liste * bsl = GetPrgBsListe(this); bsl != NULL; bsl = GetBs1Suc(bsl)) {
        baustein * bs = GetBs1Baustein(bsl);
        const char * name = GetBsKommentar(bs);
        if (name[0] != 'S' && name[0] != 'T') {
            if (fp != NULL) {
                const char * bez = GetPrgBez(this);
                fprintf(fp,"Fehler %s: Kommentar %s von Baustein %s: Kein S oder Z\n",
                        bez, name, GetBsBez(bs));
            }
            return FALSE;
        }
    }
    return TRUE;
}

```

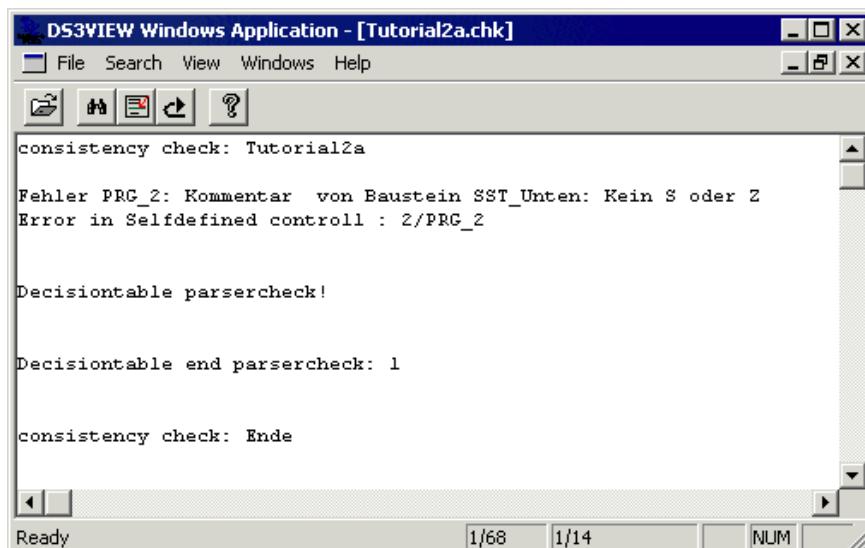


Figure 33.3: Consistency file



The return value can also be used for marking an inconsistent custom control. Using the *Draw* interface function, we paint a little green square to the top left corner of the control if it is inconsistent.

```
HRESULT CProgStrgDurchlaufzeit::Draw(CDC* pDC, int px, int py)
{
    DrawFromBitmap(pDC,px,py, IDB_PROGSTRG);
    if (!DatenOk()) {
        CBrush br(RGB(0,255,0));
        CRect r(px,py-4,px+4,py);
        pDC->FillRect(&r,&br);
    }
    return NO_ERROR;
}
```

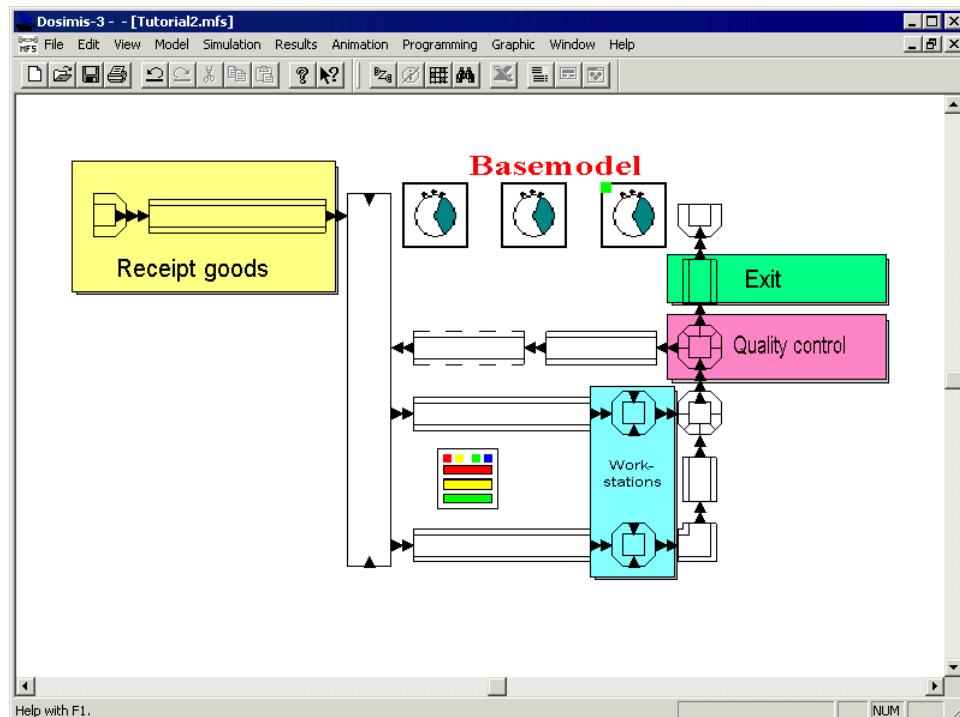


Figure 33.4: Showing up inconsistent controls

33.2.4 Simulator (part 2)

The disadvantage of the control in current state is, that it writes its output to a file with a fixed filename. For comparing two models we would need different output files. An easy way to achieve that is to use an additional function of *simlib*. Instead of using *fopen* for opening the file, use *rewrite_ext*. This will open a file for write access. The file will have the same name as the model itself, but a custom file extension, which must be set as first parameter to *rewrite_ext*. In our example the file extension shall be “*tpt*”.

```
void CProgStrgDurchlaufzeit::SimInitialisierung()
{
    datei = rewrite_ext("tpt");
}
```

This method is useful when comparing different models with different names. But if the same model has several controls served by this library, they all would try to open the same file. So we are in need to test if the file already was opened. Further we want to put a table heading to



the top of the file. Remember, the file variable was declared outside the C++ class. So it is accessible by all instances of the class.

```
void CProgStrgDurchlaufzeit::SimInitialisierung(int step)
{
    if (step == 1) {
        if (datei == NULL) {
            datei = rewrite_ext("tpt");
            fprintf(datei,"Obj Zeit Dauer Baustein\n");
        }
    }
}

void CProgStrgDurchlaufzeit::SimBeenden(int step)
{
    if (step == 1) {
        if (datei != NULL) {
            fclose(datei);
            datei = NULL;
        }
    }
}
```

When reading the output file now, it would be unclear which control wrote the information into it. So we want to add the name of the target module.

```
void CProgStrgDurchlaufzeit::SimBearbeitung(baustein * bs, long warum)
{
    :
    else if (warum == OBJEKTAUSFAHRT && name[0] == 'T') {
        :
        fprintf(datei,"%d %7.1f %7.1f %s\n",
                get_obj_typ(obj),
                aktuellerzeitpunkt,
                aktuellerzeitpunkt - alterzeitpunkt,
                GetBsBez(bs));
    }
}
```

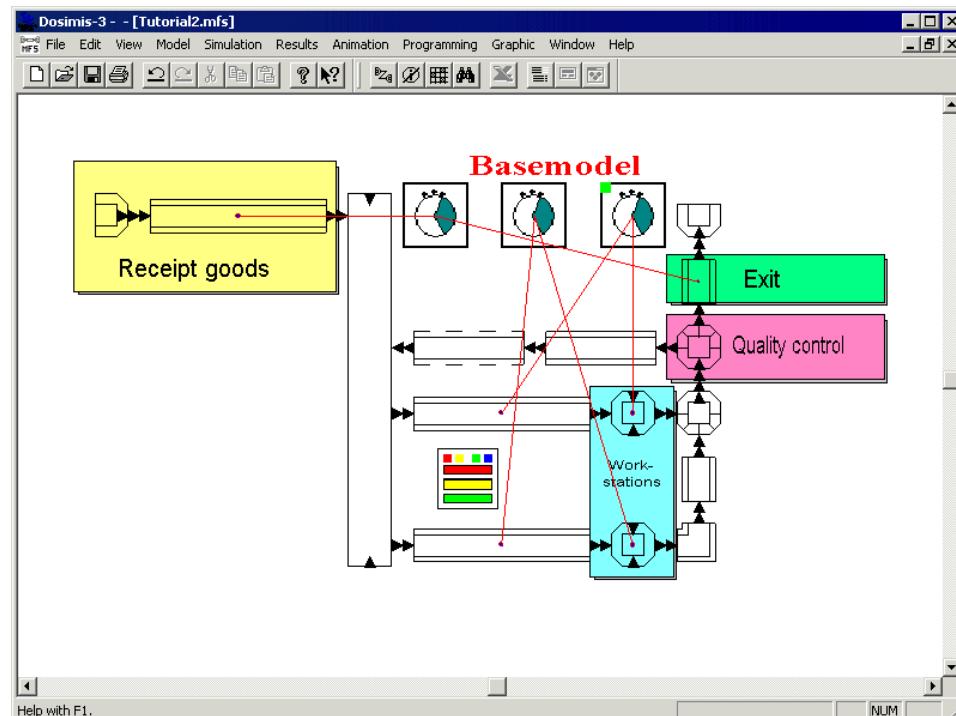


Figure 33.5: Relation of three custom controls



Now the output could look as follows:

```
Obj      Zeit      Dauer Baustein
10     214.8    129.3 Oben
1       304.6     94.7 Oben
1       324.6   114.7 Ziel
1       382.1   113.3 Oben
1       402.1   133.3 Ziel
1       463.4   122.2 Oben
1       483.4   142.2 Ziel
2       545.9   138.1 Unten
1       549.0   229.2 Oben
1       569.0   249.2 Ziel
2       629.8   222.0 Ziel
1       636.8   172.5 Oben
20      660.4   131.2 Unten
```

33.2.5 Using Parameters

Checking the file state can be avoided by giving each instance of the library class an own variable of type *FILE*, which gets initialized when starting a simulation. But when we distinguish between file variables (one per instance), we cannot link them all to the same physical file. So we create a file name from the model name and an unique number for the custom control. Then we open that file when simulation starts. So every control has its own file. Remind that member variables of C++ classes should begin with ‘m_’ to distinguish them from global variables when reading the code. Of course this does not make them a member variable... This can only be achieved by putting the variable declaration within the class declaration in *Durchlaufzeit.h*.

```
class CProgStrgDurchlaufzeit : public CProgStrg {
    :
    :
// Ab hier Anwenderspezifische Daten
public:
    FILE * m_datei;
};
```

Initialize it in the constructor (*DurchlaufzeitPara.cpp*).

```
CProgStrgDurchlaufzeit::CProgStrgDurchlaufzeit()
{
    m_datei = NULL;
}
```

References to the module-global variable *datei* should be removed or replaced.
(*DurchlaufzeitSim.cpp*)

```
void CProgStrgDurchlaufzeit::SimInitialisierung(int step)
{
    if (step == 1) {
        matfluss * mfs = get_matfluss();
        char name[200];
        sprintf(name,"%s_%d.tpt",GetMfsBez(mfs),GetPrgNr(this));
        m_datei = fopen(name,"w");
        fprintf(m_datei,"Obj      Zeit      Dauer Baustein\n");
    }
}
```



```

void CProgStrgDurchlaufzeit::SimBearbeitung(baustein * bs, long warum)
{
    :
    :
    if (warum == OBJEKTAUSFAHRT && name[0] == 'T') {
        :
        fprintf(m_datei,"%-4d %7.1f %7.1f %s\n",
                get_obj_typ(obj),
                aktuellerzeitpunkt,
                aktuellerzeitpunkt - alterzeitpunkt,
                GetBsBez(bs));
    }
}

void CProgStrgDurchlaufzeit::SimBeenden(int step)
{
    if (step == 1) {
        fclose(m_datei);
    }
}

```

33.2.6 Saving and Restoring Control Parameters

The file name of the output file is a parameter to the control. It is kind of a constant name - if the control number does not change - but it can be made an editable parameter. If we want to do so, we need to store and load control parameters to the model file (the so called material flow system file or *mfs* file). For that reason the previously generated name must be known and assigned to a control, when using it in the Dosmis-3 editor. So we declare a variable *m_name* and initialize it with a file name. But this cannot be done using the class constructor, because name and number of the control is not known yet when constructing it. Names and numbers will be assigned later by Dosmis-3, e.g. after loading a control or after initializing a new control. But you can use the consistency check function *DatenOK* for initializing parameters, which is called each time after loading or creating a control.

```

class CProgStrgDurchlaufzeit : public CProgStrg {
    :
    :
// Ab hier Anwenderspezifische Daten
public:
    FILE * m_datei;
    CString m_name
};

CProgStrgDurchlaufzeit::DatenOk()
{
    matfluss * mfs = get_matfluss();
    m_name.Format("%s_%d.dlz", GetMfsBez(mfs), GetPrgNr(this));
    for (bs_liste * bsl = GetPrgBsListe(this); bsl != NULL; bsl = GetBslSuc(bsl)) {
        :
    }
}

```

For saving this name to the MFS file, use the functions *Lesen* (read) and *Schreiben* (write). Avoiding preserved keys we chose the key NAM for storing the name parameter. The code looks like this:



```

BOOL CProgStrgDurchlaufzeit::Lesen(const char * key, const char * zeile)
{
    if (!strcmp(key,"NAM")) {
        m_name = zeile;
        return TRUE;
    }
    // Return TRUE, if INDEX was interpreted
    return FALSE;
}

BOOL CProgStrgDurchlaufzeit::Schreiben(const FILE * fp)
{
    ProgSchreiben(fp, "NAM", m_name);
    return TRUE;
}

```

After saving the model you find an entry for the output filename within the MFS file. All values of custom controls are stored by the key PRG. Next to the reserved sub-keys *NUM*, *BEZ*, *BST*, *KNT*, *DLL* and *FIN* you will find a key named *NAM* which carries a string like *tutorial2_1.tpt* - the output file name of the control. The name was generated by the model name and an unique control number.

```

PRG NUM 1
PRG BEZ PRG_1
PRG BST 2
PRG BST 14
PRG DLL Durchlaufzeit
PRG NAM tutorial2_1.tpt
PRG FIN

```

33.2.7 Changing Parameters from the User Interface

The API has an interface to a parameter dialog, which may be used to show up a parameter mask for a control- just as known by other modules when setting their parameters. This is done by an interface function *Parameter* which handles a dialog being defined in the class **CPrgParameter**. This dialog class displays a dialog resource, which can be edited by using the Visual Studio resource editor. To put an input field into the empty dialog, open the *Controls - toolbar* and select the input field icon (“ab”)).

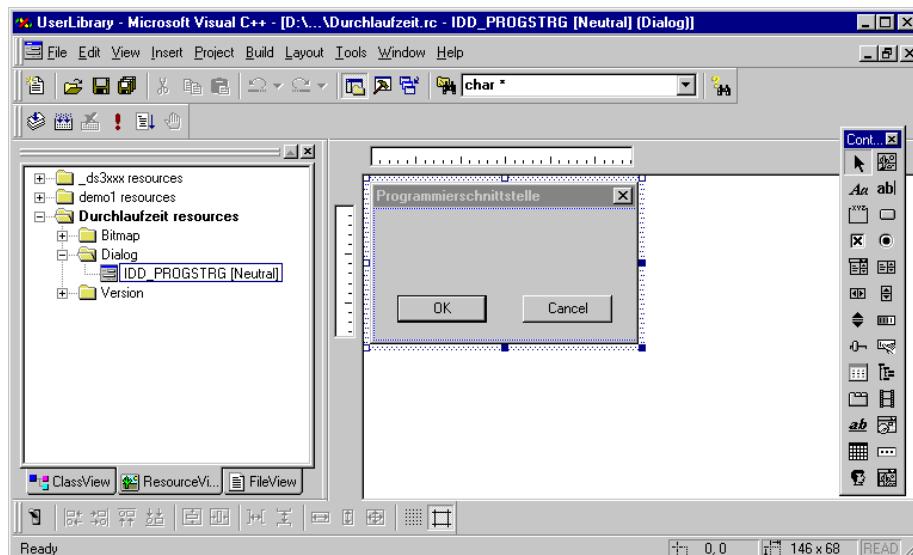


Figure 33.6: Standard dialog

After having placed an edit field (the correct term ‘edit control’ would be confusing here), call the class wizard (i.e. by the context menu of the edit field) and add a variable labeled *m_name*



of the *CString* type. This variable will automatically be declared in the dialog class. Also program code for exchanging data between the dialog class and the dialog resource will be placed into the class implementation (look at *DoDataExchange*).

```
void CPrgParameter::DoDataExchange(CDataExchange* pDX)
{
    CDIALOG::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CPrgParameter)
    DDX_Text(pDX, IDC_EDIT1, m_name);
    DDV_MaxChars(pDX, m_name, 20);
    //}}AFX_DATA_MAP
}
```

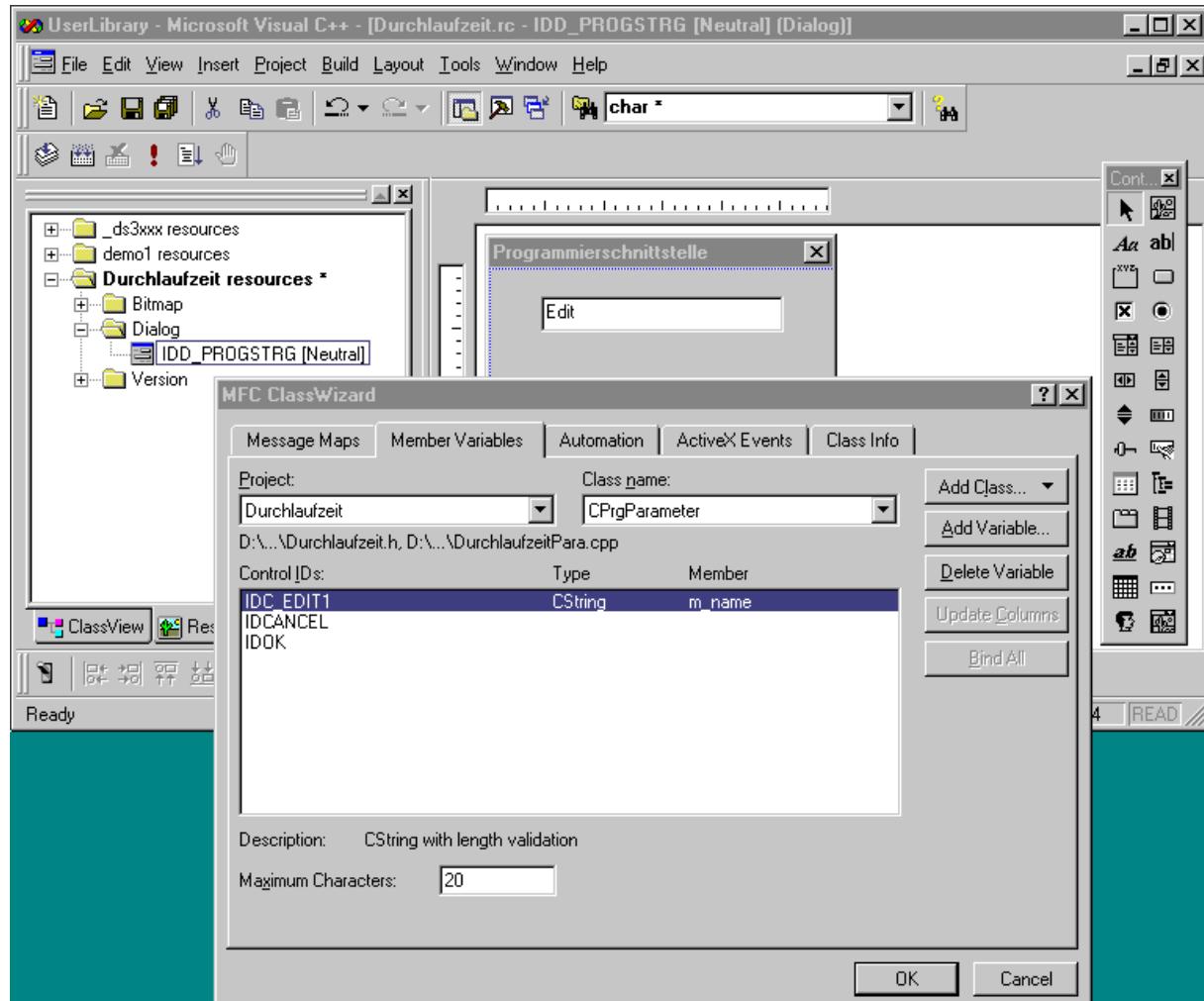


Figure 33.7: Class wizard

As you can see there is a communication path with three stations for handling dialog data. One station is the visible dialog itself, which contains current, editable data. Second station is the dialog class, which retrieves data from or sets it to the dialog when needed. Then there is the user library class, which accesses data from the dialog class and maintains that data, i.e. by loading or storing it to a file or by using it in a simulation. To exchange data between the user library class and the dialog, it is necessary to take two changes: The dialog class has to be initialized with current values. This is done using the constructor. The dialog class function *Exchange*, which is called by the user library class after OK was clicked, has to pass actual dialog data to the caller.



Remove the first two lines of *DatenOk*, which would reinitialize *m_name* with a fixed string when a consistency check is called. This would be bad, as the file name entered by the user would be overwritten each time.

```
CPrgParameter::CPrgParameter(CProgStrgDurchlaufzeit * old, CWnd* pParent /*=NULL*/)
    : CDialog(CPrgParameter::IDD, pParent)
{
    ASSERT(old);
    old = old;
//{{AFX_DATA_INIT(CPrgParameter)
//}}AFX_DATA_INIT
m_name = old->m_name;
}

void CPrgParameter::Exchange(CProgStrgDurchlaufzeit * prg)
{
    prg->m_name = m_name;
}
```

33.3 Away from Theory

By recreating an already existing and well known Dosimis-3 control, we will practice the usage of the program interface. For that purpose we implement the behavior of the capacity monitoring, which is a standard control in the Dosimis-3 module palette.

33.3.1 Data Structure

Our user library will be called *Deadlock Control* and has one single parameter: The number of Dosimis-3 objects, which may exist within a given area of Dosimis-3 modules. This parameter is called *m_limit*.

```
class CProgStrgDeadlock : public CProgStrg {
    friend HRESULT CreateProgStrg(IProgStrg ** result);
protected:
    CProgStrgDeadlock();
    virtual ~CProgStrgDeadlock();

    // Benutzeroberfläche
    virtual HRESULT Draw(CDC* pDC, int px, int py);
    virtual BOOL DatenOk(FILE * of = NULL);

    // Benutzeroberfläche Parameter
    virtual HRESULT Parameter(BOOL * change);
    virtual BOOL Lesen(const char * key, const char * zeile);
    virtual BOOL Schreiben(const FILE * fp);

    // Simulator
    virtual void SimInitialisierung(int step);
    virtual void SimBearbeitung(baustein * bs, long warum);
    virtual void SimStatistik(long aktz, long art);
    virtual void SimBeenden(int step);
public:
    // Ab hier Anwenderspezifische Daten
    int m_limit;
};
```



33.3.2 User Interface Functions

The parameter *m_limit* will be initialized within the constructor and shall be saved to the material flow file using the key LIM.

```
CProgStrgDeadlock::CProgStrgDeadlock()
{
    m_limit = 1;
}

HRESULT CProgStrgDeadlock::Draw(CDC* pDC, int px, int py)
{
    DrawFromBitmap(pDC,px,py, IDB_PRODSTRG);
    return NO_ERROR;
}

BOOL CProgStrgDeadlock::Lesen(const char * key, const char * zeile)
{
    // Return TRUE, wenn INDEX interpretiert wurde
    if (!strcmp(key,"LIM")) {
        sscanf(zeile,"%d",&m_limit);
        return TRUE;
    }
    return FALSE;
}

BOOL CProgStrgDeadlock::Schreiben(const FILE * fp)
{
    // Return TRUE, wenn Einträge gemacht wurden
    ProgSchreiben(fp,"LIM",m_limit);
    return FALSE;
}
```

33.3.3 Dialog

The parameter *m_limit* has to be transferred to the dialog class by passing it as a parameter to its constructor. After the dialog was confirmed by OK, *m_limit* will be retrieved using the dialog class function *Exchange*.

```
///////////
// CPrgParameter dialog

CPrgParameter::CPrgParameter(CProgStrgDeadlock * old, CWnd* pParent /*=NULL*/)
    : CDialog(CPrgParameter::IDD, pParent)
{
    ASSERT(old);
    _old = old;
    //{{AFX_DATA_INIT(CPrgParameter)
    //}}AFX_DATA_INIT
    m_limit = old->m_limit;
}

void CPrgParameter::Exchange(CProgStrgDeadlock * prg)
{
    prg->m_limit = m_limit;
}
```

33.3.4 Simulator Functions

Within the simulator functions, the simulation behavior of the control is implemented. When getting initialized, a list of all relevant entry junctions is created. These will be used for denying entry to an area with exceeding capacity.

33.3.4.1 Initialization

Within the initialization, the list of all entry junctions is created. The special case of the assembly module and the loading station is considered, where only the first entry junction is



of interest - an assembled object gets destroyed immediately and does not affect the number of objects within the controlled area. Further we have to check if a junction is on the edge of the control area. If not, it is not relevant, either - an object entry through such a junction is caused by an exit through a junction also belonging to the controlled area. It does not affect the sum of objects within the area.

```
// ##### Simulator #####
static void fuege_kn_ein(knoten *kn, IProgStrg *gsz)
{
    for (bs_liste * bsl = GetPrgBsListe(gsz); bsl != NULL; bsl = GetBslSuc(bsl)) {
        if (GetBslBaustein(bsl) == GetKnBsAus(kn))
            break;
    }
    if (bsl != NULL)
        return;
    AddPrgKnListe(gsz, kn, TRUE);
}

static void erzeuge_knliste(IProgStrg *lauf)
{
    for (bs_liste * bsl = GetPrgBsListe(lauf); bsl != NULL; bsl = GetBslSuc(bsl)) {
        baustein * b = GetBslBaustein(bsl);
        knotenlist *knl = GetBsEinKn(b);
        if ((bs_typ(b) == AUFNAHME) || (bs_typ(b) == MONTAGE)) { /* Nur den ersten Knoten
einfuegen */
            fuege_kn_ein(GetKn1Knoten(knl), lauf);
        }
        else {
            for (; knl != NULL; knl = GetKn1Suc(knl)) {
                fuege_kn_ein(GetKn1Knoten(knl), lauf);
            }
        }
    }
}

void CProgStrgDeadlock::SimInitialisierung()
{
    erzeuge_knliste(this);
}
```

33.3.4.2 Processing Events

When processing events, we just have to react on object entry or object exit. Otherwise the occupancy of our area did not change. At first the complete occupancy of the whole area has to be determined. Even here we need to process the *assembly* and *loading station* separately. When the capacity limit *m_limit* is reached, all junctions within the previously created junction list get blocked, denying entry of further objects. If the capacity is not reached, all junctions with waiting objects get sorted by their activation time and opened accordingly. So the object with the longest waiting time will enter the area first. By that a FIFO behavior is implemented.

```
static double f_obj_ausloesezeit(gsz_kn *gkn)
{
    knoten * kn = GetGknKnoten(gkn);
    baustein * bs = GetKnBsAus(kn);
    objekt * obj = bs_f_obj(bs);
    return obj_ausloesezeit(obj);
}
```



```

static void fuege_sortiert_ein(gsz_kn *hilf, gsz_kn **neu_knl)
{
    gsz_kn *knl;
    int gefunden;

    if (*neu_knl == NULL) {
        *neu_knl = hilf;
        SetGknPre(hilf,NULL);
        SetGknSuc(hilf,NULL);
        return;
    }
    if (f_obj_ausloesezeit(hilf) >= f_obj_ausloesezeit(*neu_knl)) {
        SetGknSuc(hilf,*neu_knl);
        SetGknPre(*neu_knl,hilf);
        *neu_knl = GetGknPre(*neu_knl);
        SetGknPre(hilf,NULL);
        return;
    }
    knl = *neu_knl;
    gefunden = false;
    while (!gefunden) {
        if (f_obj_ausloesezeit(hilf) >= f_obj_ausloesezeit(knl)) {
            gefunden = true;
            break;
        }
        if (GetGknSuc(knl) == NULL)
            gefunden = true;
        else
            knl = GetGknSuc(knl);
    }
    SetGknPre(hilf,knl);
    SetGknSuc(hilf,GetGknSuc(knl));
    SetGknSuc(knl,hilf);
    if (GetGknSuc(hilf) != NULL)
        SetGknPre(GetGknSuc(hilf),hilf);
}

static void ordne_nach_wartezeit(IProgStrg *gsz)
{
    gsz_kn *neu_knl, *knl = NULL, *hilf;

    neu_knl = NULL;
    for (knl = GetPrgKnListe(gsz); knl != NULL; ) {
        if (bs_ist_bel(GetKnBsAus(GetGknKnoten(knl))) <= 0) {
            knl = GetGknSuc(knl);
            continue;
        }
        hilf = knl;
        knl = GetGknSuc(knl);
        if (knl != NULL)
            SetGknPre(knl,GetGknPre(hilf));
        if (GetGknPre(hilf) != NULL)
            SetGknSuc(GetGknPre(hilf),knl);
        else {
            SetPrgKnListe(gsz,knl);
        }
        fuege_sortiert_ein(hilf, &neu_knl);
    }

    if (neu_knl == NULL)
        return;
    knl = neu_knl;
    while (GetGknSuc(knl) != NULL)
        knl = GetGknSuc(knl);
    SetGknSuc(knl,GetPrgKnListe(gsz));
    if (GetPrgKnListe(gsz) != NULL)
        SetGknPre(GetPrgKnListe(gsz),knl);
    SetPrgKnListe(gsz,neu_knl);
}

```



```

void CProgStrgDeadlock::SimBearbeitung(baustein * bs, long warum)
{
    if (warum == OBJEKTEINFAHRT || warum == OBJEKTAUSFAHRT) {
        long ist_bel, akt_bel;
        gsz_kn *knl;

        ist_bel = 0;
        for (bs_liste * bsl = GetPrgBsListe(this); bsl != NULL; bsl = GetBslSuc(bsl)) {
            baustein * b = GetBslBaustein(bsl);
            akt_bel = bs_ist_bel(b);
            if ((bs_typ(b) == AUFNAHME) || (bs_typ(b) == MONTAGE)) { /* Nur den ersten Knoten
einfuegen */
                if (akt_bel > 1)
                    akt_bel = 1;
            }
            ist_bel += akt_bel;
        }
        if (ist_bel >= m_limit) {
            for (gsz_kn * knl = GetPrgKnListe(this); knl != NULL; knl = GetGknSuc(knl)) {
                sperre_knoten(knl);
            }
            return;
        }
        ordne_nach_wartezeit(this);
        for (knl = GetPrgKnListe(this); knl != NULL; knl = GetGknSuc(knl)) {
            entsperre_knoten(knl);
        }
    }
}

```

33.3.5 Abstract

This example can be found in the directory UserLibrary\Deadlock.



34 Interface for Modules

34.1 Basics

34.1.1 Events

Dosimis-3 is an event-oriented simulator. This means that the event is the basic form during the computation of simulation. These events are processed during the simulation in their temporal order, until either no more further event (e.g. deadlock) exists or the simulation time ran off. The time consumption of the module events ensures that during the simulation the time progresses.

In Dosimis-3 there is an event queue, where all active events are administered. These are sorted in their temporal order. Than in each case the first event is processed. Afterwards there are two possibilities. Either the event takes place at a later time from the renewed waiting queue, or the event switches into a passive condition. Then the event is passive as long as nothing happens, until it is again activated by another process. Dosimis-3 has three functions, which administers the events in the queue.

`void Passivate()` Passivates the event. The event starts at least, when another module activates it (e.g. by `vg_activate` of the successor).

`void Activate (double time)` Activates the event at a fixed time (in seconds). The event must be passive. Otherwise the function `Hold` should be used.

`void Hold (double duration)` Activates the event after the given duration. The event must be active. Otherwise the function `Activate` should be used.

Furthermore an event function belongs to it, which is called each time the time of the event is reached. In this function it must be decided whether the process is to be activated at a later time again (`Hold`) or no further actions are necessary (`Passivate`), until the process is again waked up from another place (`Activate`).

`eventproc()` This function will be called with each event. The function should return true, when the event is recognized and processed. Otherwise an error message occurs.

With start of simulation normally all processes are passive. Only objects are produced in the sources. By the procedure of the object delivery the successive modules are activated, so that as with a domino effect all processes in the simulation model become active gradually.

34.1.2 State Machines

The implementation of the event function is to be designed for the individual requirements. It showed up however with the development of Dosimis-3 that the implementation as a state machine is favorable. A state machine is in the theory something that possesses different states, a function to change these states and an initial state, which is accepted at the beginning of processing.



In a material flow system the states concern usually to types as: *Object entrance*, *Loading*, *beginning of work*, *working*, *driving out* or *module empty*. These states can be again classified into those, which occur at a point of time or those, which last a period. Often these states alternate. Example: After the object occurred (with the head, point like event) in a module, the duration for loading is determined and the states changes into the state *Loading*. Thus the states alternate with another, until the object left again the module.

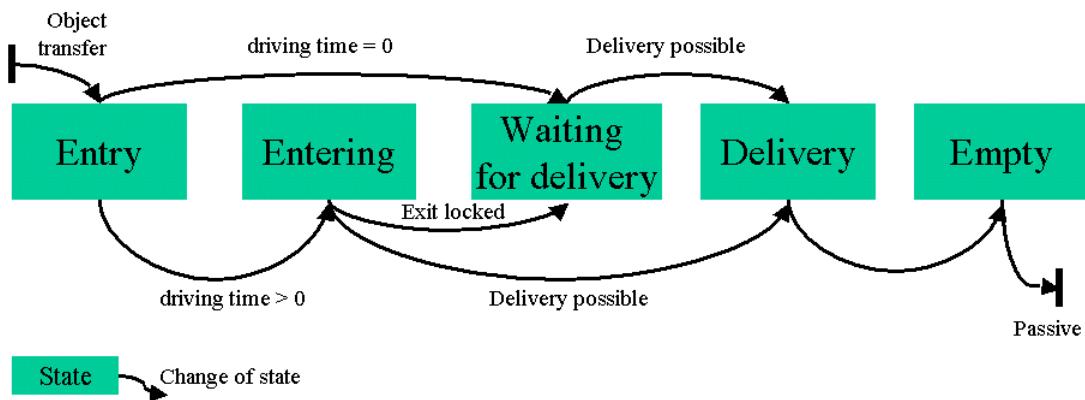


Figure 34.1: Example of a state machine

Theoretically one could step linear from state to state, finding out, that one can reach in zero-time the next state. In the interest of an efficient processing of the events however not necessary states should be jumped over. The number of event calls in the simulation determines significantly the duration of the simulation run.

The states are internal to administer and freely selectable. For this integer values are intended. However a enumerating type should be used, in order to make the source code better readable.

The function for changing the states are:

eventproc()	This function will be called with each event. The function should return true, when the event is recognized and processed. Otherwise an error message occurs.
-------------	---

The initializing function is:

OnSimNotify(PbsNotifyCode code, matfluss * mfs)	This function is called in different places in user interface and simulator. More exact information is to be inferred from the documentation of the enumerating type. Substantial events are start of simulation (multi-level from PNC_SIMINITIALISIERUNG0 to 9), simulation end, reset statistics (after interval statistics at each statistics time and/or intermediate statistics after the pre run) or when the model was read in.
--	--



34.2 Interface Functions

The new created class has the following functions:

```
class IProgBaustein {
public:
    virtual void SetBausteinDaten(CPbsDaten * pbs) = 0;

    virtual HRESULT Draw(CDC* pDC, int px, int py) = 0;
    virtual BOOL DatenOk(FILE * of = NULL) = 0;

    virtual BOOL Lesen(const char * key, const char * zeile) = 0;
    virtual BOOL Schreiben(const FILE * fp) = 0;

    virtual int OnSimNotify(PbsNotifyCode code, matfluss * mfs) = 0;

    virtual bool eventproc() = 0;
    virtual bool aufnahmebereit(knoten *kn, objekt *obj, bool sofort) = 0;
    virtual void eintrag(knoten *kn, objekt * obj) = 0;

    virtual void OnStoRealBeginn() = 0;
    virtual void OnStoRealEnde() = 0;

    virtual bool HatVorfaehrtstrategie() const = 0;
    virtual bool HatVerteilstrategie() const = 0;

    virtual int SelbstProgZiel(objekt * obj, long eing) = 0;
    virtual int SelbstProgEing() = 0;

    virtual bool HatTaetigkeiten() const = 0;
    virtual bool HatManuelleTaetigkeiten() const = 0;

    virtual CPropertyPage * GetPropertyPage() = 0;
};
```

The meaning can be found in the following table:

Draw(CDC* pDC, int px, int py)	These functions have the same meaning as in controls. Further information can be found there.
DatenOk(FILE * of = NULL)	
Schreiben(const FILE * fp)	
Lesen(const char * key, const char * line)	
eventproc()	This function will be called with each event. The function should return true, when the event is recognized and processed. Otherwise an error message occurs.
aufnahmebereit(knoten *junc, objekt *obj, bool immediate)	This function is called each time, an object transfer should take place. The values of <i>obj</i> and <i>junc</i> serve the transferred object and the junction, where the object should enter into the module. The parameter <i>immediate</i> is <i>false</i> , when only the possibility for a transfer is checked. A value of <i>true</i> means, that the object will be transferred successive. The function <i>aufnahmebereit</i> should return <i>false</i> , when a transfer of the object is impossible.
eintrag (knoten * junc, objekt * obj)	This function is called when the object enters the module. Here initializations can be made, which are necessary in this situation.



OnSimNotify(PbsNotifyCode code, matfluss * mfs)	This function is called in different places in user interface and simulator. More exact information is to be inferred from the documentation of the enumerating type. Substantial events are start of simulation (multi-level from PNC_SIMINITIALISIERUNG0 to 9), simulation end, reset statistics (after interval statistics at each statistics time and/or intermediate statistics after the pre run) or when the model was read in.
void OnStoRealBeginn()	This function is called, if the module is disturbed. When overlay failures the call takes place only with the first failure.
void OnStoRealEnde()	This function is called, if the module is undisturbed. When overlay failures the call takes place only with the last failure.
HatVorfahrtstrategie()	This function should return <i>true</i> , if the module possesses a right of way strategy. This is made available automatically in the user interface. Additionally the number of entrances can be increased arbitrary.
HatVerteilstrategie()	This function should return <i>true</i> , if the module possesses a distribution strategy. This is made available automatically in the user interface. Additionally the number of exits can be increased arbitrary.
SelbstProgZiel(objekt * obj, long entrance)	This function will be called during simulation, when the strategy is set to <i>selfprogrammed</i> and a destination is to be calculated for an object.
SelbstProgEing()	This function will be called during simulation, when the strategy is set to <i>selfprogrammed</i> and an entrance is to be calculated.
HatTaetigkeiten()	This function should return <i>true</i> , when the module implements tasks. This will be used at several places in during simulation and in the user interface, for example with the extended statistic.
HatManuelleTaetigkeiten()	This function should return <i>true</i> , when the module implements manual tasks. This will be used at several places during simulation and in the user interface, for example with the scheduling in work areas.
GetPropertyPage ()	If the interface possesses own parameters, then a subdialog can be conveyed in this function. This is indicated within module dialogue at the second position. The further subdialogs serves the parameters, which are common to nearly all modules.



Apart from the central functions for writing and reading of the MFS- file, drawing and data validation in the user interface as well as event processing in the simulation optional functions for the treatment of activities, right of way- and distribution strategies as well as failure events are intended. The optional functions can be overtaken in such a way, as these are implemented in the class *CProgBaustein*.

34.3 Helper for Modules

For simplified processing differently functions are placed in the basic class of each component:

```
class CProgBaustein : public IProgBaustein
{
:
// Informationen zum Baustein
baustein * GetBaustein()
double GetGeschwindigkeit()
double GetLaenge()

// Vorfahrt und Verteilen
knoten * GetVorfEingKn()
knoten * GetVertAusgKn()
int NeuerEingang()
long NeuesZiel()

// Hilfsfunktionen Transprot
void ObjektEingefahren()
void ResetBs()
bool ObjektAbgabe()

// Eventhandling
void VgActivate()
void Passivate()
void Activate(double zeitpunkt)
void Hold(double dauer)

// Hilfsfunktionen Arbeiten
void RuestenAnmelden(double dauer,
                      long min = 0, long max = 0, long qual = 1, long unt = 0)
bool RuestenAktiv()
void ArbeitAnmelden(double dauer, long nt,
                     long min = 0, long max = 0, long qual = 1, long unt = 0)
bool ArbeitAktiv()

// Animation
void AnimateObjekt(EPbsColor col)
:
}
```

These are:

<code>baustein * GetBaustein()</code>	Gets the reference to the module. From this and by the functions of the Simlib all properties of the module can be accessed.
<code>double GetGeschwindigkeit()</code>	Get the value of the speed parameter, which can be set in the basic dialog. By this model constants can be used.
<code>double GetLaenge()</code>	Get the value of the length parameter, which can be set in the basic dialog. By this model constants can be used.



knoten * GetVorfEingKn()	Get the reference to the junction, which was determined by the right of way strategy and from which the next object should enter the module.
knoten * GetVertAusgKn()	Get the reference to the junction, which was determined by the distribution strategy and through which the next object should leave the module.
long NeuerEingang()	The call of this function evaluates the right of way strategy. The return value is the number of the entrance, which was determined. This value remains valid as long as these data is reset by the function <i>ResetBs()</i> . A return value of 0 means, that no entrance is found (e.g. for each entrance no object is waiting).
long NeuesZiel()	The call of this function evaluates the distribution strategy. The return value is the number of the exit, which was determined. This value remains valid as long as these data is reset by the function <i>ResetBs()</i> . A return value of 0 means, that no exit is found (e.g. tailback).
void ObjektEingefahren()	This function is to be called, when an object has entered the module complete. This ensures, the all information at the junction are reset correctly.
void ResetBs()	This function is to be called to reset the parameters of the right of way strategy and the distribution strategy. The cal should be made when the object has left the module and a new entrance is determined.
bool ObjektAbgabe()	This function controls the delivery of an object. The result is <i>true</i> , when the object has left the module, otherwise it is <i>false</i> .
void VgActivate()	Activates the module at the entrance, from which the object has entered the module.
void Passivate()	Passivates the event. The event starts at least, when another module activates it (e.g. by <i>vg_activate</i> of the successor).
void Activate (double time)	Activates the event at a fixed time (in seconds). The event must be passive. Otherwise the function <i>Hold</i> should be used.
void Hold (double duration)	Activates after the given duration. The event must be active. Otherwise the function <i>Activate</i> should be used.



void RuestenAnmelden(double duration, long min = 0, long max = 0, long qual = 1, long intr = 0)	Announces a task for setup. <i>Duration</i> determines the duration of this task. By <i>min</i> , <i>max</i> , <i>qual</i> und <i>intr</i> it can be called for worker. When <i>min</i> has a value of 0, no personal should be used. Further on Dosimis-3 overtakes the control, until the function RuestenAktiv returns <i>false</i> .
Bool RuestenAktiv()	Checks, whether the announced setup has not finished. Returns <i>true</i> , when this is happened.
void ArbeitAnmelden(double duration, long nt, long min = 0, long max = 0, long qual = 1, long intr = 0)	Announces a task for object process. <i>Duration</i> determines the duration of this task. By <i>min</i> , <i>max</i> , <i>qual</i> und <i>intr</i> it can be called for worker. When <i>min</i> has a value of 0, no personal should be used. Further on Dosimis-3 overtakes the control, until the function ArbeitAktiv returns <i>false</i> .
bool ArbeitAktiv()	Checks, whether the announced process work has not finished. Returns <i>true</i> , when this is happened.
void AnimateObjekt(EPbsColor col)	Creates an entry in the trace file, which is interpreted during animation. The enumeration <i>EPbsColor</i> holds the RGB - value of the enumerated colors. This can be extended arbitrary.

34.4 Data Input

The data input with modules is a little more complex than with the controls. While the data input is independent with control, the input takes place with modules in a sub window of the input dialogue. Such sub dialogs are called *PropertyPage*. Such a property page is then placed apart from the standard parameters (e.g. costs, strategies...) in the module dialogue.



```
class IProgBaustein {
:
virtual CPropertyPage * GetPropertyPage() = 0;
:
};

class CPbsPropertyPage : public CPropertyPage
{
    CProgBaustein * m_bst;
DECLARE_DYNAMIC(CPbsPropertyPage)

// Construction
public:
    CPbsPropertyPage(CProgBaustein * bst, UINT id);
    ~CPbsPropertyPage();

:
};
```

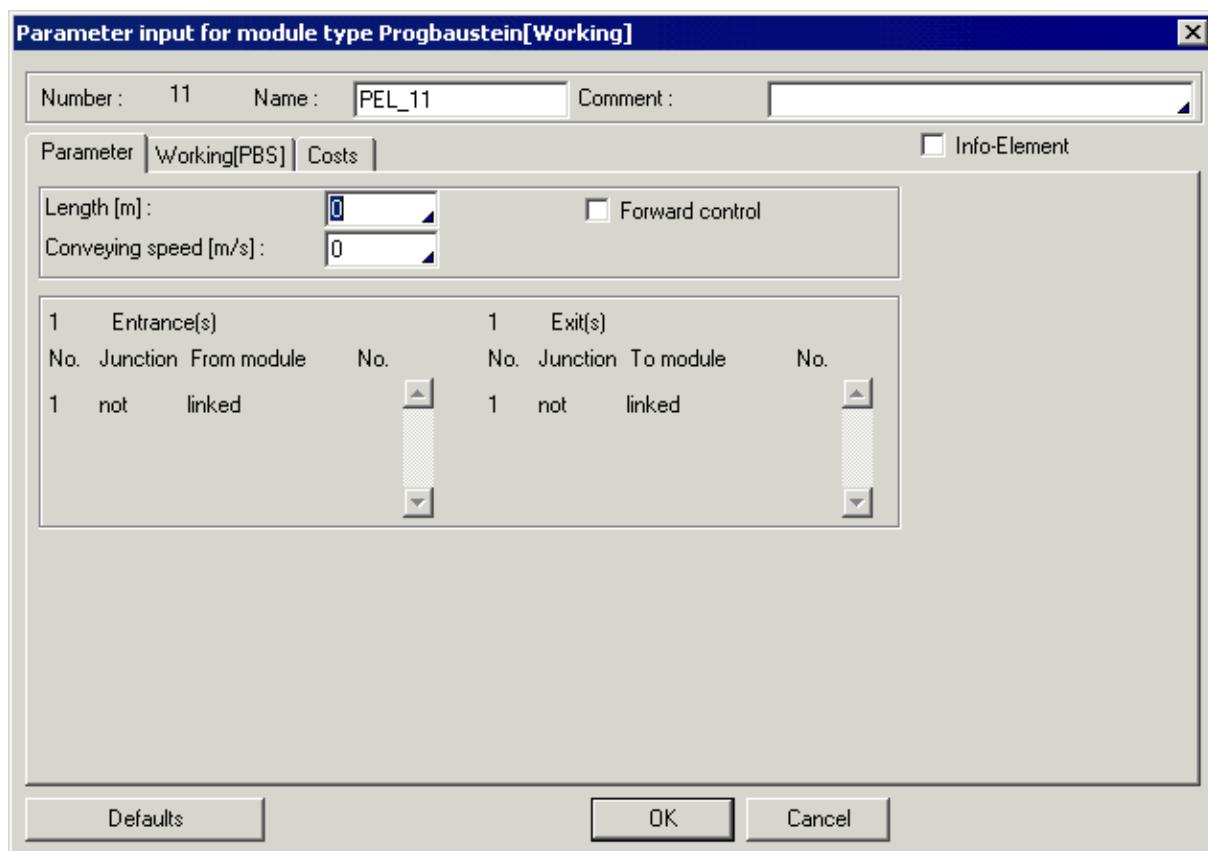


Figure 34.2: Basic parameter of a user defined module



In the basic dialogue shows the connections of the module with its environment. Additionally two parameters for speed and length can be entered. These can be used during simulation. The structure is fixed and cannot be changed.

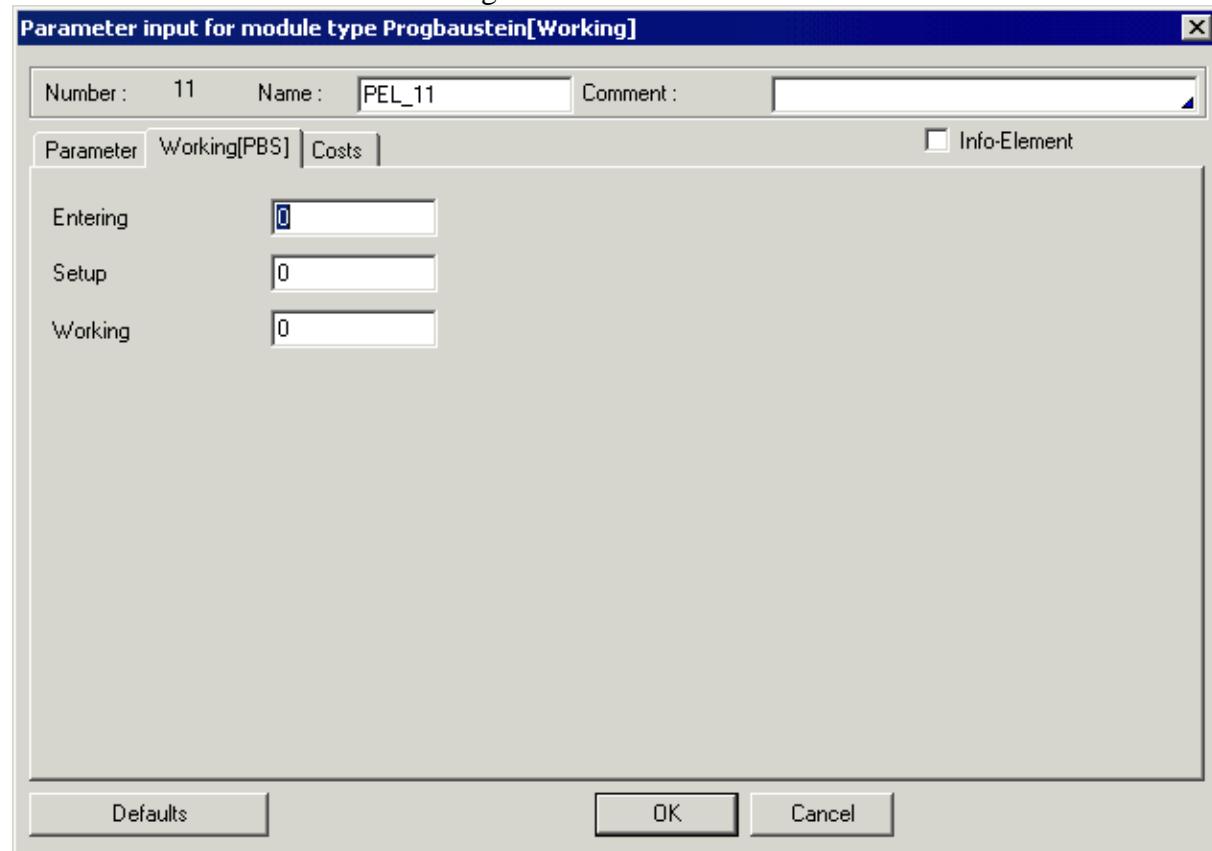


Figure 34.3: Input parameter of the user defined module

The defined input parameters are indicated in their own PropertyPage. These can be extended in Visual studio.



34.5 Templates

Dosimis-3 offers 2 template files, in which components for 2 different applications are produced. This is on the one hand a component with work functionality. Here one shows, how activities in Dosimis-3 are treated. The second template file shows the handling of the right of way and distribution strategies. Here is shown, how and when these decisions are made and how these are integrated into the event flow.

34.5.1 Template Work

```
class CPbsArbeit : public CProgBaustein
{
public:
    CPbsArbeit() {
        m_oldobjtyp = 0;
        m_einfahrdauer = m_ruestdauer = m_bearbeitungsdauer = 0; }

    virtual HRESULT Draw(CDC* pDC, int px, int py);
    virtual BOOL DatenOk(FILE * of = NULL);
    virtual BOOL Lesen(const char * key, const char * zeile);
    virtual BOOL Schreiben(const FILE * fp);

    virtual bool aufnahmefbereit(knoten *kn, objekt *obj, bool sofort);
    virtual void eintrag(knoten *kn, objekt * obj);
    virtual int OnSimNotify(PbsNotifyCode code, matfluss * mfs);
    virtual bool eventproc();

    double m_einfahrdauer;
    double m_ruestdauer;
    double m_bearbeitungsdauer;
protected:
    virtual bool HatTaetigkeiten() const {
        return true; }
    virtual bool HatManuelleTaetigkeiten() const {
        return true; }

    virtual CPropertyPage * GetPropertyPage();
private:
    long m_oldobjtyp;
    bool RuestenNotwendig();
};
```

Apart from the functions of the basic class this class serves three variables for the determination of duration for loading, the duration of setup as well as working time. Additionally still another variable is needed, in which the type of object, which was worked on last, is stored. Hereby in the function *RuestenNotwendig* it is determined whether the type of object changed in relation to the last treatment.

```
bool CPbsArbeit::aufnahmefbereit(knoten *kn, objekt *obj, bool sofort)
{
    UNUSED_ALWAYS(kn);
    UNUSED_ALWAYS(obj);
    UNUSED_ALWAYS(sofort);
    return zustand == PBZ_LEER;
}
```

Returns *true*, when the module is empty (*leer*).



```
void CPbsArbeit::eintrag(knoten *kn, objekt * obj)
{
    zustand = PBZ_EINFAHREN;
    UNUSED_ALWAYS(kn);
    UNUSED_ALWAYS(obj);
}
```

Set the state to *einfahren* when an object enters the module. The activation of the element takes place automatically.

```
int CPbsArbeit::OnSimNotify(PbsNotifyCode code, matfluss * mfs)
{
    UNUSED_ALWAYS(code);
    UNUSED_ALWAYS(mfs);
    if (code == PNC_SIMINITIALISIERUNG) {
        m_oldobjtyp = 0;
        zustand = PBZ_LEER;
    }
    return 1;
}
```

In the initialization the state is set to empty. The variable, where it is stored which type of object was worked on last, is set to 0. This is needed for the regulation whether a setup is necessary.

```
bool CPbsArbeit::RuestenNotwendig()
{
    if (m_oldobjtyp > 0) {
        objekt * obj = bs_f_obj(GetBaustein());
        if (obj != NULL && obj_typ(obj) != m_oldobjtyp) {
            return true;
        }
    }
    return false;
}
```

In this function it is checked, whether the object type has changed. Her the special case during start up is recognized. There is no setup for the first object.



```

bool CPbsArbeit::eventproc()
{
    if (zustand == PBZ_EINFAHREN) {
        Hold(m_einfahrdauer);
        zustand = PBZ_EINGEFAHREN;
    }
    else if (zustand == PBZ_EINGEFAHREN) {
        ObjektEingefahren();
        if (RuestenNotwendig()) {
            RuestenAnmelden(m_ruestdauer);
            zustand = PBZ_RUESTEN;
        }
        else {
            ArbeitAnmelden(m_bearbeitungsdauer, 1);
            zustand = PBZ_ARBEITEN;
        }
    }
    else if (zustand == PBZ_RUESTEN) {
        if (!RuestenAktiv()) {
            ArbeitAnmelden(m_bearbeitungsdauer, 1);
            zustand = PBZ_ARBEITEN;
        }
    }
    else if (zustand == PBZ_ARBEITEN) {
        if (!ArbeitAktiv()) {
            zustand = PBZ_AUSFAHREN;
        }
    }
    else if (zustand == PBZ_AUSFAHREN) {
        m_oldobjtyp = obj_typ(bs_f_obj(GetBaustein()));
        if (ObjektAbgabe()) {
            ResetBs();
            VgActivate();
            zustand = PBZ_LEER;
        }
        Passivate();
    }
    else if (zustand == PBZ_LEER) {
        Passivate();
    }
    else
        return false;
    return true;
}

```

This is the central event function of the module. Here the state transitions are good to recognize. Furthermore this section shows the interaction of the two functions *ArbeitAnmelden* and *ArbeitAktiv*. Only the minimum necessary parameters are registered when *ArbeitAnmelden* is called. All further parameters are optional. This means in this case that no personnel is requested. Additionally the last type of object must be stored before the object is transferred, so that on this fact it can be determined whether a setup must take place next time. It is to be made clear here the object type stored is the type after the treatment. When the type of the entering object is to be saved, the instruction is to be made as the last command in the state *eingefahren*. Then the old value to determine setup is evaluated and can be changed.



34.5.2 Template *Crossing*

```
class CPbsWeiche : public CProgBaustein
{
public:
    CPbsWeiche() {
        m_einfahrdauer = m_verweildauer = 0; }

    virtual HRESULT Draw(CDC* pDC, int px, int py);
    virtual BOOL DatenOk(FILE * of = NULL);
    virtual BOOL Lesen(const char * key, const char * zeile);
    virtual BOOL Schreiben(const FILE * fp);

    virtual bool aufnahmefbereit(knoten *kn, objekt *obj, bool sofort);
    virtual void eintrag(knoten *kn, objekt * obj);
    virtual int OnSimNotify(PbsNotifyCode code, matfluss * mfs);
    virtual bool eventproc();

    double m_einfahrdauer;
    double m_verweildauer;
protected:
    virtual bool HatVorfahrtstrategie() const;
    virtual bool HatVerteilstrategie() const;

    virtual CPropertyPage * GetPropertyPage();
};


```

Beneath the functions of the basic class this class possesses two variables, where loading duration and duration of staying in the module can be set.

```
bool CPbsWeiche::aufnahmefbereit(knoten *kn, objekt *obj, bool sofort)
{
    UNUSED_ALWAYS(obj);
    if (zustand == PBZ_LEER) {
        if (sofort) {
            knoten * kne = GetVorfEingKn();
            if (kne == NULL) {
                NeuerEingang();
                kne = GetVorfEingKn();
            }
            return kn == kne;
        }
        return true;
    }
    return false;
}
```

The decision of the acceptance is somewhat more complicated. If the module is not empty, in each case *false* is returned. Otherwise a simple inquiry is acknowledged with *true*. If however the object is to be transferred, it must be examined additionally, which entrance was selected in accordance with the right of way strategy. Then *true* is returned only if the junction agrees additional with the selected entrance. If the right of way strategy was not evaluated yet (*GetVorfEingKn* then returns NULL, i.e. an entrance was not calculated), this is done and then the selected junction is compared with that, from which the object was to bring in.



```
void CPbsWeiche::eintrag(knoten *kn, objekt * obj)
{
    zustand = PBZ_EINFAHREN;
    UNUSED_ALWAYS(kn);
    UNUSED_ALWAYS(obj);
}
```

Set the state after the entry to *einfahren* (entering). The activation of the event takes place automatically.

```
int CPbsWeiche::OnSimNotify(PbsNotifyCode code, matfluss * mfs)
{
    UNUSED_ALWAYS(code);
    UNUSED_ALWAYS(mfs);
    if (code == PNC_SIMINITIALISIERUNG) {
        zustand = PBZ_LEER;
    }
    return 1;
}
```

The initialization is done by setting the state to *leer* (empty).

```
bool CPbsWeiche::eventproc()
{
    if (zustand == PBZ_EINFAHREN) {
        Hold(m_einfahrdauer);
        zustand = PBZ_VERWEILEN;
    }
    else if (zustand == PBZ_VERWEILEN) {
        ObjektEingefahren();
        Hold(m_verweildauer);
        zustand = PBZ_AUSFAHREN;
    }
    else if (zustand == PBZ_AUSFAHREN) {
        long nz = NeuesZiel();
        if (nz > 0 && ObjektAbgabe()) {
            ResetBs();
            long ne = NeuerEingang();
            if (ne > 0) {
                VgActivate();
            }
            zustand = PBZ_LEER;
        }
        Passivate();
    }
    else if (zustand == PBZ_LEER) {
        Passivate();
    }
    else
        return false;
    return true;
}
```

This is the central event function of the module. Here clearly the implementation is to be recognized as a state machine. After the object has entered the module, with the help of the distribution strategies the new exit (*NeuesZiel*) is determined. If no destination were found or the transfer of the object is impossible, the process is passivated. This can be justified in the fact that all possible successive modules have no more capacity. If in this case something changes, the process is again activated by a successive module and the state, which did not change, will be evaluated again. *NeuesZiel* returns the selected destination, if this was successfully computed at an earlier time. If the object transfer is successful, the strategy entries are put back (*ResetBs*). Afterwards a new entrance is determined immediately and the module at the selected entrance (*VgActivate*) is activated. Afterwards the state is set to empty



and the process passivates itself. Important is also the last else branch. Here the module passivates itself again, if it is activated from the system and the module is empty. This lies in the fact that the methods *Activate* and/or *VgActivate* do not examine, whether this is meaningful for the module concerned. For this the module is responsible.

```
bool CPbsWeiche::HatVorfahrtstrategie() const
{
    return true;
}

bool CPbsWeiche::HatVerteilstrategie() const
{
    return true;
}
```

Both functions return *true*, which means, the module can possess several entrances and exits and that the parameterizing of an appropriate strategy is necessary.



35 Program Interface for Decision Tables

35.1 Introduction

When working with decision tables, the necessity to interact with external programs (i.e. for achieving information controlled by user interface libraries) shows up a need for an decision table based program interface.

The code for interacting with decision tables must be contained within a DLL file, too. For this purpose the project *EtFunktion* exists in the project workspace *UserLibrary* of Dosimis-3. There you can implement functions, which can be called by decision tables.

These functions may have any number of parameters. There is a data structure *CEtParList*, which grants access to all given parameters in C.

Beneath the self defined functions three further functions are available.

```
extern "C" AFX_EXT_API void SetSimlib2(ISimlib4 * pSimlib)
extern "C" AFX_EXT_API void SimInitialisierung(int step)
extern "C" AFX_EXT_API void SimBeenden(int step)
```

These serve for the necessary initialization and finalization. *SetSimlib2* initializes the library of the interface functions of the material flow system.

The function *SimInitialisierung* is called when starting a simulation. The initialization takes place in 4 steps

1. initialization of table.
2. initialization of modules.
3. Initial actions of decision tables.
4. initialization of objects in the modules.

Analogous the function *SimBeenden* is called when the simulation ends. The finalization takes place in 2 steps:

1. finalization of controls.
2. finalization of modules.

35.2 Declaration

Declare decision table functions as follows:

```
extern "C" AFX_EXT_API int TEST(CEtParList * list)
```

Pay attention to the key words, which mean:

- **extern "C"** This function is externally accessible by the C naming convention.
The parameter specific extension of the function name is not applied.



- AFX_EXT_API Exported function of an DLL file, taking care about corresponding naming conventions.
- int The return value is of *int* type. This value can be used in decision table conditions and assignments.
- TEST Name of the function. Case-sensitive.
- CEtParList * list Parameter list of the function.

Example:

```
extern "C" AFX_EXT_API int TEST(CEtParList * lst)
{
    baustein * bs;
    double      d;
    GetValue(lst, 0,bs);
    GetValue(lst, 1, d);
    return d * ist_bel(bs);
}
```

35.3 Consistency Check

If a *call* command is used in a decision table, the consistency check tries to find a DLL function with the given name. If it is not found, an error message will be shown and the simulator cannot be started.

```
Function >Test< does not exist in >EtFunktion.dll<
In decision table`GSZ_2`
```

The parameter list is not checked. Pay attention to parameter types and count when programming and calling external functions.

35.4 List of Parameters

There are several versions of the function *GetValue*, which gives access to the parameters of a decision table function call.

```
bool GetValue(CEtParList * list, int p, int &r)
bool GetValue(CEtParList * list, int p, double &r)
bool GetValue(CEtParList * list, int p, CString &r)
bool GetValue(CEtParList * list, int p, baustein* &r)
bool GetValue(CEtParList * list, int p, knoten* &r)
bool GetValue(CEtParList * list, int p, objekt* &r)
```

These functions can be used for fetching parameters from the parameter list. The parameters in the list are ordered from left to right - the leftmost parameter in the decision table call is the parameter on position 0. Give the wanted position as the **p** parameter to *GetValue*.

The action **call("TEST" , 1, 2)** shall be handled by the following function:



```
extern "C" AFX_EXT_API int TEST(CEtParList * lst)
{
    int d1, d2;
    GetValue(lst, 0, d1);
    GetValue(lst, 1, d2);
    :
}
```

After the second *GetValue* statement, the variable d1 contains 1 and d2 contains 2.

The parameter type cannot be checked by the consistency check. Prove parameter types valid within the C-Function. *GetValue* will return *false* if the given variable pointer does not accept the wanted parameter.

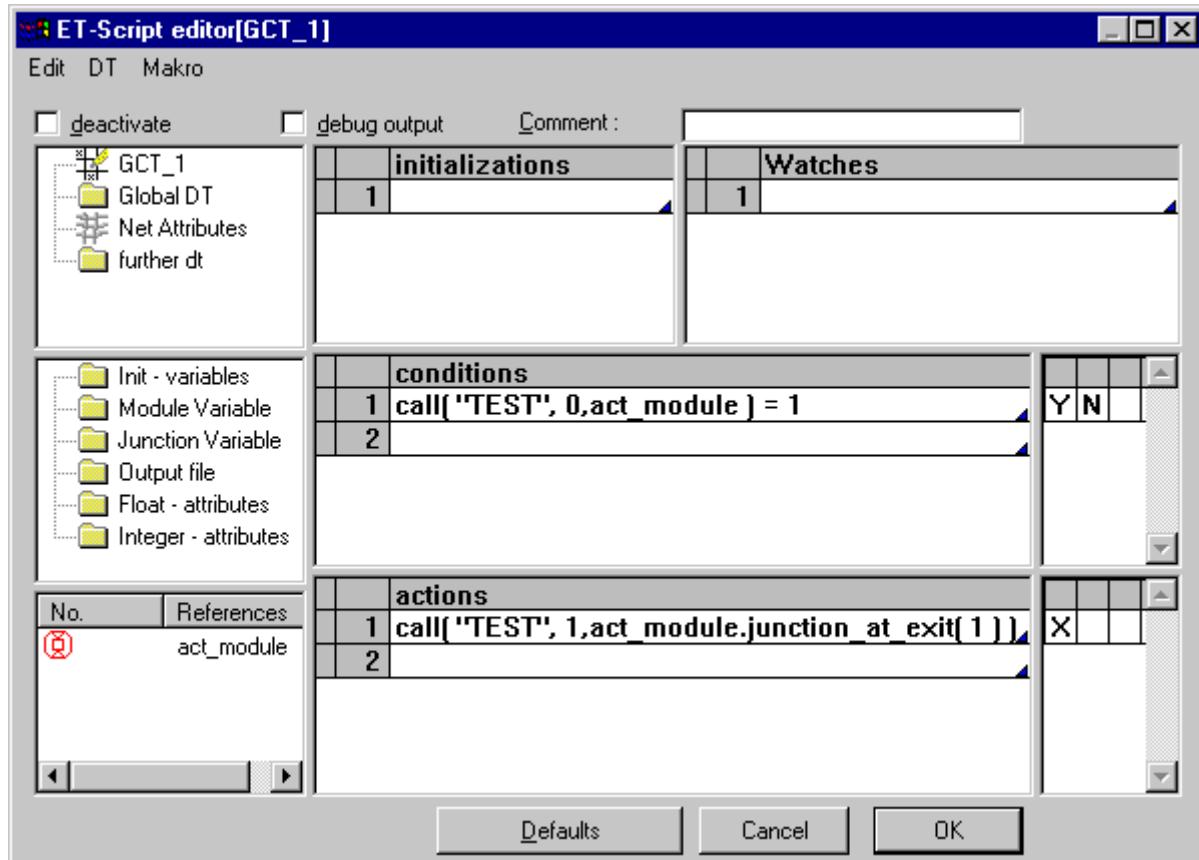


Figure 35.1: Decision table calling a DLL function

It is possible to call the same function with different parameter types. The following function will illustrate how to do so by accepting either a junction and a module as parameter 2.



```
extern "C" AFX_EXT_API int TEST(CEtParList * lst)
{
    int i;
    GetValue(lst, 0, i);
    if (i == 0) { // this is a condition
        baustein * bs;
        if (GetValue(lst, 1, bs))
            return bs_ist_bel(bs);
    }
    else if (i == 1) { // this is an action
        knoten * kn;
        if (GetValue(lst,1,kn)) {
            knoten_sperren3(kn);
            return 0; // not evaluated by dt
        }
    }
    write_meldung("Error: Wrong parameter in >TEST<");
:
}
```



36 English - German Dictionary

According to the German keywords used in the examples and in the data structure, here are some translations of the German keywords. This will be expanded continuously. Look in your online help file for the actual version.

Ausfahren	to exit
Aufnahmebereit	Acceptance
Baustein	Module
Datei	File
Dauer	Duration
Durchlaufzeit	throughput Time
Durchsatz	Throughput
Einfahren	to enter
Einfügen	to insert
Eintrag	Entry
Entscheidungstabelle (ET)	decision Table (DT)
Geschwindigkeit	speed
Jetzt	now (actual moment in simulation)
Knoten	junction
Länge	length
Lesen	to read
Ordnen	to sort
Rüsten	Setup
Schreiben	to write
Sperren	to lock
Tätigkeiten	Task
Verteilstrategie	distribution strategy
Vorfahrtstrategie	right of way strategy
Zeit	Time
Zeitpunkt	Point in Time / moment



37 Installation Instructions

During installations with Windows 2000/XP/Vista you must have **administrator privileges** so that the necessary entries can be made in the system files. Otherwise you may install only a demo version. Standard user may also make no insertions into the general start menu, so that no start icon for DOSIMIS-3 can be registered.

If you do not possess administrator privileges, but you know the administrator password, you can start the setup from the context menu. When pressing the Shift - key and clicking with the right mouse button on the installation program, the context menu offers the entry **execute as** So you can start the installation program under the user identification of the administrator.

The installation of DOSIMIS-3-Windows is very simple.

Installation from CD:

Usually the installation is started automatically from the inserted CD. If this does not fit, you can carry out the installation by hand by starting the program **Setup.exe** from the explorer. This file you will find in the root directory of the CD.

Installation from floppy disk:

Insert the first floppy disk into the disk drive and start **Setup.exe**.

Installation from the network:

Download the DOSIMIS-3 installation file from the Internet (www.sdz.de) and save the file on your hard disk. You should raise these (for example double-click in the explorer). The installation of DOSIMIS-3 is started.

Installation from an e-mail (one part):

Save the file from the attachments on your hard disk. You should start this file (for example double-click in the explorer). The installation of Dosimis-3 is started then.

Installation from an e-mail (several parts):

Save the files from the attachment into one directory on your hard disk. You should start the program **Setup(...).exe**.

Start DOSIMIS-3 once after the installation. The installation is finished then.

After the start of the setup further interrogations are made, in which the basic settings must only be confirmed so that the installation can be carried out. During the first installation it is recommended, to carry out a **full - installation**. Start DOSIMIS-3 once with **administrator privileges**. The installation is finished then.

DOSIMIS-3 requires the operation system Windows 7/Vista or XP. For the version only the least configuration is presupposed of the used operating system. (XP: Pentium 300 und 128MB; Windows 7/Vista: 1-Gigahertz (GHz) 32-bit Processor, 1 GB main memory). For working with the 3D component of Dosimis-3 a DirectX - graphics adapter is recommended. A free hard disk size of approx. 200 MB is recommended in order to examine smaller models without complications. Bigger models achieve however very soon a use of several a hundred



megabyte hard disk size for the statistics. For the graphics a resolution of at least 1024x768 points is expected. However a resolution of 1280x1024 or higher is recommended.

37.1 Software Protection

For the installation of Dosimis-3 you do not need the dongle plug. The availability of the plug is only examined when starting Dosimis-3. If you do not possess a hardlock, a demo version of Dosimis-3 is started.

37.1.1 Single License

Put the dongle into a USB port of your computer and start Dosimis-3. All functions are then available, which are provided by the dongle. The plug must be put in before the start of Dosimis-3. If the plug is taken off during the work with Dosimis-3, working with Dosimis-3 is not possible.

37.1.2 Network License

Install Dosimis-3 on the computer, which should serve as the license server. Put afterwards the dongle plug into a USB haven of this computer. Install Dosimis-3 (without dongle) on a client computer, which is in the same network segment and which can communicate with the server by TCP/IP. Then all functions are available, which are provided on the central dongle. The number of computers, which can work at the same time with Dosimis-3, is limited on 20. The number of Dosmis-3 sessions on individual computer is however unlimited.

37.2 Trouble Shooting

37.2.1 Windows Runtime

Dosimis-3 uses libraries which belong to the standard installation since Windows. If these are not installed on your computer, a fault report of the following type occurs during the start:



Figure 1: Fault report

Copy from the directory *Support* the corresponding file into the installation directory.

37.2.2 Excel 97

The Excel-interface of Dosimis-3 works with Excel-2000. If you have installed Excel97 on your computer and Dosimis-3 cannot create any connection to Excel, you should start the file excel97.reg from the support directory. This will register the corresponding entries in the system files.

37.2.3 DirectX Installation

During the installation of Dosimis-3 DirectX 9.0c version will also be install, if it's not available on the computer. Unfortunately, there are 2 different versions, with the installation



does not recognize. This can cause the error message that the file "d3dx9_30.dll" does not exist.

Right of way DTs determine the entrance, where an object enters a module (Right of way Strategy). In the same way a distribution DTs determine the exit, where an object leaves a

Then install DirectX directly from the CD. You can find it in the directory \\DxSetup\\DxSetup.exe. Afterwards repeat the installation.

37.2.4 Windows Help System and Windows 7 / Vista

The Windows Vista operating system does not support by default, the Windows Help system. Therefore the update Windows6.0-KB917607-x86.msu is provided, which you can find in the support folder. You hereby install the Help system under Vista. The update Windows6.1-KB917607-x86.msu supports Windows 7 (32 bit), the update Windows6.1-KB917607-x64.msu supports Windows 7 (64 bit).

37.3 Installation Files

After a **Full Installation** Dosimis-3 creates the following directory structure. The root directory is **\Programme\SDZ\Dosimis-3** or **\Program Files\SDZ\Dosimis-3**, depending on the configuration of windows. This has as a subdirectory the different languages, which contain directories with examples. Further on the register *Support* is created, in which various files are installed that are needed only in special situations. A further subdirectory is that of the programming interface (UserLibrary).

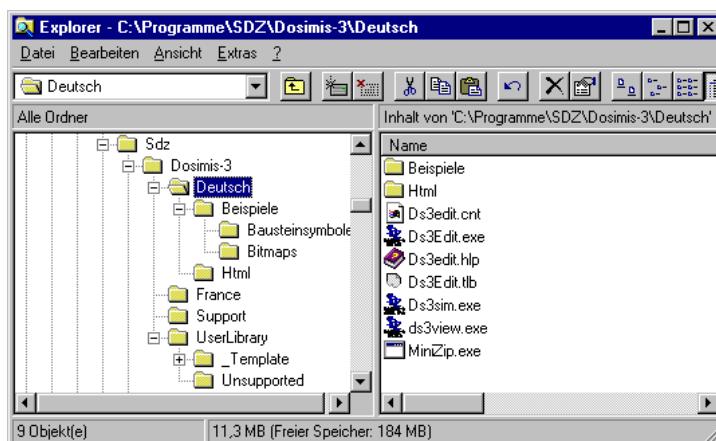


Figure 2: Installation directory

37.3.1 Sample Files

The subdirectory **examples** contains different tutorial models which can be used as basic models of the tutorials for the private study. To do that you find part 1 and 2 of the tutorial in the online help system

37.3.2 Module Symbols

In this directory a Dosimis-3 - model with module symbols differing from the standard representation can be found. These can be placed above *graphics/import* (*ds3edit.dxg*) into every model. Further information should be taken from the documentation on the graphic comments.



37.3.3 Bitmap Examples

This directory contains some bitmaps which can be used for the bitmap animation. If there are 2 versions to the same representation, one contains the supplementary entries for the colored animation (*_mA.bmp) depending on the object state. You can copy these files into the project folder and you rename these in 0.bmp or similar. For this purpose see the documentation of the bitmap animation.

37.3.4 HTML

These register contains the documentation of the libraries for the programming interface and the description of the main - data structures of Dosimis-3. On the one hand this is through the html - possible for files and an Internet browser (Netscape or Internet Explorer). Alternatively the file *index.chm* for the disposal, a compiled help file that provides for compact form into these information, stands. Precondition for the use of the HTML You understand that one from help file following information:

Distribution of the HTML-help:

Precondition to the considering of the HTML-help file is a program the *Microsoft Internet Explorer4.0* and a newer versions. If this is not available on your computer, the file *hh.exe* can be found in the support folder. You create a link from the help file to this file and register as a destination by the following command:

```
C:\Programme\SDZ\Dosimis-3\Support\HH.EXE C:\Programme\SDZ\Dosimis-3\Deutsch\Html\index.chm
```

If you chose another installation path, the paths are to be adjusted in the above line correspondingly (is valid only for the HTML - help file of the programming interface).

37.3.5 Support

In this directory you find some system libraries, programs and further files which are normally not needed.

37.3.6 User Library

In this directory the modules of the user interface lie for self programmed operations. For every defined operation a subdirectory is created here.

37.3.6.1 *Template*

This directory contains the pattern files from which the source text of a new user interface is created.

37.3.6.2 *Deadlock*

Example of a deadlock control.

37.3.6.3 *EtFunktion*

Project files of the *functions calls* of decision tables.

37.3.6.4 *Unsupported*

This directory contains the data structures of Dosimis-3. These can be used during the programming. It is recommended however to use the methods of the simulator library since the data structures can be changed.

There are 3 more files are in this folder. These can be used in the environment Visual-C++.



- Usertype.dat: In this file you find all keywords of the data structure Dosimis-3. If this file is copied into the installation directory by Visual Studio (Standard: C:\Programme\Microsoft Visual Studio\Common\MSDev98\Bin), these keywords are highlighted in color.
- Autoexp.dat: This file contains instructions for the Visual C++ Debugger. Here it is fixed, which additional information is shown in the variable window, without expanding complex data structures. This file is to copy to the installation directory of Visual Studio.
- Sysincl.dat: This file contains the names of the system include files to avoid warning during compilation.

More information on these files will be found in the documentation of Visual Studio.

37.3.7 X-Files

This folder contains some examples of machines using in the 3D visualization.



38 Update Instructions

Updating prior versions of DOSIMIS-3 can be done by two steps

1. Installation of DOSIMIS-3 according to the installation instructions
2. Updating the license on your CodeMeter USB Stick

38.1 Additional Note for Update Installations

Perform installation as described in the installation manual. If there is yet an existing version of DOSIMIS-3 on your computer, it gets replaced by the new version. You may prevent this by choosing another installation directory. So you are able to use both versions.

38.2 Updating the CodeMeter USB Stick

The highest version number of DOSIMIS-3 that you are allowed to start is stored on your CodeMeter Stick. If your currently installed version is newer than the one licensed on the stick, DOSIMIS-3 will run in demo mode only.

To update the license information on your CodeMeter stick, plug the stick directly into a free USB plug at your PC. An update of your CodeMeter stick is only possible, if the stick is plugged in at the PC, where Dosimis-3 is running.

Now start DOSIMIS-3 and choose *Update* within the *Help* menu. Choose *CM-Stick Update* from the update wizard. With an existing internet connection select **Internet update**.

Your system will now send information about the plugged CodeMeter stick to SDZ GmbH. Despite from the yet existing information on the stick, no other content will be transmitted. The servers at SDZ GmbH will now check if you are permitted to update. If so, the necessary update information will be sent back immediately. The CodeMeter stick then gets updated within a few seconds, activating your license.

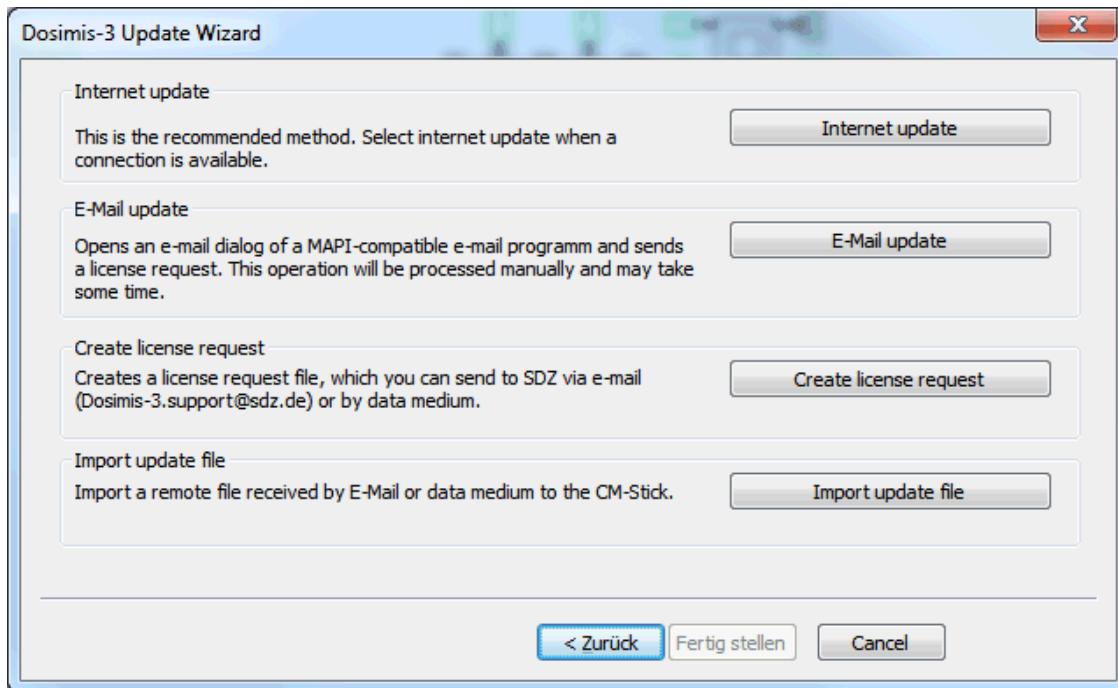


Figure 3: Update wizard

Alternatively, choose one of the following update mechanisms:

By choosing **E-Mail update**, DOSIMIS-3 will try to open your MAPI enabled e-mail client with a prepared mail to SDZ GmbH. The mail will have the necessary license information from your CodeMeter stick attached. Your request will be checked manually, so be prepared that this process may take a day or two. You will get a response e-mail with a file attachment for updating your CodeMeter stick. The stick can be updated by clicking **Import update file**.

If your e-mail client is not Microsoft MAPI enabled, you may simply store the license file to your hard disc and send it to us using either some transport medium like CD or USB stick, or by attaching the file to an e-mail manually. To do so, choose **Create license request** and select a directory for saving the license file.

The license file sent to us by e-mail or data medium needs to be worked on manually, so please be patient. This process may take a day or two. We will send back an update file, which then has to be transferred to the CodeMeter stick by using the **Import update file** button in the update wizard.



39 Release Notes 6.2

This is a list of all relevant changes from version 6.1.

39.1 Standard Functions

39.1.1 User Interface

The version number is 6.2.

The dongle must be updated for the current version. For this purpose a server program, which makes this automatically in demand, is provided at SDZ.

Alternative Linking Mode.

Less restrictive parameter checks in dialogs.

39.1.2 Modules

For each assembly list in an assembly station it can be determined, whether the objects should not be destructed.. Then the objects remain in the system with all their data. These can be brought back to the system in a Disassembly Station.

The Shuttle can accept more than one object. The behavior can be controlled by several parameters.

The Conveying Circuit has a restriction table. Before every delivery of an object it can be determined, whether the object should really leave the module.

The type of a module can be changed subsequently.

39.1.3 Controls

39.1.3.1 Capacity Monitoring

The Capacity Monitoring can control the objects in transit between two areas.

39.2 Decision Tables

The linking of a Timer with a decision table has been changed. It takes place in the same way as the linking with activating modules.

When calling another decision table Parameters can be passed to the sub decision table.

There are further data types like File, Animation Text or Transport order.





40 Release Notes 6.1

This is a list of all relevant changes from version 6.0.

40.1 Standard Functions

40.1.1 User Interface

The versions number is 6.1.

One focus of this year's development was the expansion and unification of decision tables (as well as the continuation of the redesign of the user interface).

It has begun to revise the result graphs. The three main types: occupancy, condition and throughput graphs are included. The remaining types, as well as a restructuring of the result menus and dialog, are subsequently confirmed in individual automatic updates.

Automatic notification and installation for possible update of the program (this can be disabled on request).

Passive timers are highlighted in the model with a square layout.

In occupancy diagram of maximum value which a source is determined in accordance with the specified capacity. Only if the capacity is not specified in the source, the default value of 100 is used.

Besides the animation can now also use the online simulation are performed proportionally.

The pretty print output of decision tables now includes the Default column. Further, the *indifferent symbol* (\emptyset) has been replaced by a minus (-).

Before any use of the new version 6.1 the dongle must be updated. Select in DOSIMIS-3 from the menu Help / Update item CM-Stick update. In the dialog that different options are available for update.

40.1.2 Modules

The Shuttle with several tables has a new strategies. A currently release table can be preferred when several tables are valid for the next transport.

The parameterization of the Multiple Work Station has been is sophisticated according the process results.

40.1.3 Controls

The linking mode has been simplified, the usage of the shift key is not necessary anymore. When a control is active (blue), all controls selected next will be assigned to that control. Another control can be made the active control only, if no element is selected before. This mode can be changes at *Model/Configure*.



40.2 Decision Table

Initial actions have been extended to initial decision tables.

The net attributes have been renamed to *global variable*. The variable view has been redesigned.

The decision table dialog has an additional view. This contains the call stack of the decision tables.

The reference view support beneath the modules and junctions all controls used in the decision table.

The decision table dialog has a new button *Step Into*. This invokes the stop in the next decision table invoked by the current actions.

Several conditions and actions can be selected and copied simultaneously. Rules can be moved horizontally.

Multiple references in the syntax dialog (Ctrl-E) have been eliminated.



41 Release Notes 6.0

This is a list of all relevant changes from version 5.1.

41.1 Standard functions

41.1.1 User Interface

The version number is 6.0.

The user interface is redesigned. The models can be arranged more easily. Docking of windows is more intuitive. The whole design is revised.

The dongle must be updated for the current version. For this purpose a server program, which makes this automatically in demand, is provided at SDZ.

41.1.2 Modules

The Assembly possesses in addition to the single process time a total process time, which is used at the end of the assembly process.

The random functions are extended by the triangle distribution.

The parameterization of varying distribution functions is simplified.

Many modules possess a efficiency factor. So the performance can be diversified very easy.

41.1.3 Controls

41.1.3.1 Transportsystem

Transport system can be passivated. Alternative strategies can be examined very easy. Additionally there are three neu results statistics: State diagram of the orders, throughput time diagram of the order and waiting queue diagram.

A global transport control can be put into a model as a control element. If there are no connections to any module, it is treated as global control.

The unloading stations do not have to be unique assigned to one transport control. The common usage of an unloading station is now possible.

41.1.3.2 Work Area

Work areas can be parametrized, even they are not connected to any module, failure, . . . All corresponding elements of the material flow are offered in the dialog. This happens for compatibility reasons, because e.g. failures do not have to be connected with the work area. The elements should be connected with the area further on for clarity reasons.

41.2 Decision tables

Deletion of all break points of a decision table can be made with a mouse click.



There are 2 new functions for the removal of objects from storage. Thus objects can be release from stock by an attribute value.



42 Release Notes 5.1

This is a list of all relevant changes from version 5.0.

42.1 Standard Functions

42.1.1 User Interface

The version number is 5.0.

The elements can be selected by characteristics. Thus further utilities are provided to navigate comfortably in the model and validate the model.

The model can be searched for the use of model constants. In addition the dialogue was extended.

The simulation parameters have been restructured.

42.1.2 Modules

The storage module can now be initialized by an input file. In this context also the reading in routine of the sources was revised. So now also attributes and further parameters can be assigned to the object with the initialization.

Somit können die Aufnahme von mehreren Transporteinheiten einfach abgebildet werden.

The shuttle possesses now a capacity. Thus several transportation units can be simply carried with one cycle.

In the destination oriented distribution now any Integer attribute can be selected as criterion apart from the type of object and the destination parameter.

In the modules now also conditional break points can be defined.

The storage module can be analyzed more exactly in online simulation with the help of the 3D-Visualisation.

42.1.3 Controls

42.1.3.1 Transport System

In a transport system vehicles can transfer several loads. Now it is also possible that partial loaded vehicles can take up further loads at loading stations.

Furthermore with the initialization values for acceleration and braking can be assigned to the vehicles. These can be considered when driving through blocking sections.

42.1.3.2 Work Area

In a work area each group of workers can have an individual habitual place. Additionally the scheduling rules were extended and the handling is more intuitive.



42.2 Decision tables

All decision tables possess now beside the initials actions also finale actions. Thus now also the instructions necessary with simulation end can be implemented.

The string - operations were extended significant. Now also initialization variables can be used for these operations. Additionally there are several new assignment operators, which make the definition of decision tables more efficient.

42.3 3D-Animation

In the 3D-Animation now curves and corner conveyor can be visualized and animated.



43 Release Notes 5.0

This is a list of all relevant changes from version 4.3a.

43.1 Standard functions

43.1.1 User Interface

The version number is 5.0.

When copying of single elements (individually, Ctrl + E), the references are no longer copied. This is still available by the method *Copy (Group)*.

Moving of elements can be made with cursor keys.

The deletion of junctions has been improved. When you delete a junction by the context menu the parameters of the module will be adjusted.

The parameters of an element can transfers to a similar element with the help of Collective parameterization.

Modules which are initialized can be selected by the on the selection menu. The initialization may be displayed in the layout.

43.1.2 Modules

The storage space module can be accurately and with precise cycle times. The storage statistics has been expanded.

The multi-functional work station has been extended with a further process, the cleaning.

The conveying circuit can be steered that way, that he waits, even if the last object is not yet fully exited.

43.1.3 Controls

43.1.3.1 Work Area

Failures and capacity monitoring, which area may be affected from a work, must be connected with the work area. (Note: In connecting mode the CTRL button must be pressed when the control is to be inserted).

43.1.3.2 Storage Control

The new storage control supports the storage disposition in the selection of suitable storage areas.

43.1.4 Results

The main services 3 result can be defined individually in the properties of the results. These are provided for quick access in the result menu and the context menu of the elements.

43.1.5 Animation

In the continuous animation the change the workplace of the worker is animated continuously.



43.2 Decision Table

The possibilities of animation texts are significantly expanded. These texts can be displayed in multiple lines. In addition, model parameters can be displayed. There is the method *CmdGetParameter (Field, par1, par2, ...)*, which works like the Excel interface, support the evaluation of parameters or results.

The methods of quick tables 2 additional features have been added. The random process of the timer can be examined by the decision table will be evaluated. This allows the usage of any random process.

The storage module now supports the removal by the unique the number of the object. This allows individual objects specifically to be removed.

The variable of initialization of the main decision table can be used in all sub decision tables. The data type string is supplemented by further methods.

In the online simulation, the time consumption of the decision tables and quick table can be examined precisely (profiling).

43.3 Programming Interface

43.3.1 COM-Interface

The document supports the method **IncludeModel** method (**name As String, x As Single, y As Single, prefix As String**). This allows building automated generated models with predefined sub models. There are also several new methods to generate 3D views.

43.4 3D-Animation

The 3D visualization is extended by an animation.

43.5 Changes and Corrections

43.5.1 Excel-Interface

The initialization of modules is not properly transferred if the same object was used several times.



44 Release Notes 4.3a

This is a list of all relevant changes from version 4.3.

44.1 Standard Functions

44.1.1 User Interface

The version number is 4.3a.

The conversion of presentation length into the capacity of modules has been supplemented for the modules conveyor, block section, accumulating conveyors 2 and Lifo buffer.

44.1.2 Modules

The combination of shuttle with several tables and sensors does not work reliably. This is prohibited by the consistency check.

If a table had to discard a transport order because the entrance, the empty table is driven to, was disrupted during the empty run, orders got lost. This is corrected.

44.1.3 Controls

44.1.3.1 Work Area The completion of activities is been revised. Besides the object processing are now also failure processing generated automatically, if the CTRL key is pressed while clicking.

In viewing the work breaks reference failures were not displayed properly, if at the same time fixed or periodic failures were selected.

44.1.3.2 Shuttle Control In the shuttle control the junctions were all too often locked and unlocked. This is implemented now much more efficiently.

44.1.4 Results

The results of global transport controls was erroneous. This is solved.

44.1.5 Animation

The continuous animation block sections and accumulating conveyors 2 did not work correctly. This is corrected.

In the last section of the accumulating conveyor (analogous conveyor, ...) the current occupancy is shown only, if the value is > 10.

The bitmap animation was limited on the canvas 10000x10000. This is has been extended on the entire working area.



44.2 Programming Interface

44.2.1 Modules Programming

When copying crossings via copy and paste the parameters of the right of way strategy and distribution strategy got lost.

44.3 3D-Visualization

The initialization of the 3D visualization was clearly too restrictive. Now is the visualization on computers much more possible.

Circles could not been placed.



45 Release Notes 4.3

This is a list of all relevant changes from version 4.2.

45.1 Standard Functions

45.1.1 User Interface

The information view supports beneath name, number, speed, ... also cycle time and throughput time of modules.

For simplifying modeling a reference length for capacity of the modules can be defined. For new modules the capacity will be calculated automatically.

Single elements can be passivated without deleting them from the model.

45.1.2 Modules

There are 2 new right of way strategies: percentaged - and Bauschuld.

In the secondary strategy priority of exits the priority can be defined for each object type individually.

The change of object types in the workstation can be calculated by 3 different algorithms

A shuttle can be defined with several tables on one rail. 3 alternatives for calculating the travel time are provided. Efficiency factor, sensors

45.1.3 Controls

45.1.3.1 Work Area

For simplifying the determination of the distances between the modules, each module can be defined by a coordinate. With the help of the worker speed the way time between 2 modules can be calculated very easy. 2 different algorithms are provided.

45.1.3.2 Failures

The definition of a reference failure is extended by a further parameter. The counting module can be defined separately in case of the need of such a special module for each failure. Additionally there are 3 different algorithms for calculation the behavior in case that the factor is > 1.

45.1.3.3 Shuttle Control

The shuttle control synchronizes a cycled sequence of modules.

45.1.4 Results

The parameter of the results are extended significantly. The axes can be scaled individually. Additionally the event for cycle time measuring can be selected arbitrary.



The occupation diagram supports mixed selections of elements with similar representation (capacity monitoring control, worker employment, ...). Further methods of the buffer analysis are provided now for these elements.

Beneath the information of modules the production diagram shows the states of workers.

Parameterization of evaluation modules has been simplified significantly. The parameter of a representation can be taken over to an evaluation module.

The simulation results of different simulation models can be compared in the same diagram.

45.1.5 Animation

The orientation of the bitmaps can be suppressed in the bitmap-Animation.

45.2 Decision Tables

45.2.1 Usage

Every decision table possesses a default rule. This will be executed, if no explicit rule fits.

45.2.2 New Functions

The list of mathematical functions is extended with 17 methods.

The scheduling of transport systems can be activated by a decision table.

The right of way and distribution strategies without parameters (FIFO, minimal occupation, ...) can be called from a self defined right of way and distribution strategy.

The shuttle supports sensors, which lead to a call of decision tables. Further on the driving direction, the state of movement and the position of entrances/exits can be evaluated by decision tables.

45.2.2.1 Progress Indicator

A new element called *progress indicator* is provided. Percentile values can be visualized during animation.

45.2.2.2 Quicktable

Sorting of quick tables becomes stable. This means that the sequence of lines, which are identically according the sorting criteria, remain in their order.

45.3 Programming Interface

45.3.1 Excel-Interface

Display of connecting modules (number/ name) with modules with many entrances/ exits when showing distribution strategies, positions,



45.3.2 COM-Interface

The COM-interface supports the request of the actual version number. COM scripts can be implemented version safe. Further on the method *StartSimulation* possesses a return value which can be evaluated. Therefore series of simulation runs can be interrupted safely.

45.4 3D-Visualization

The actual version contains a beta-version of the 3D-visualization.

45.5 Changes and Corrections

For compatibility to previous versions corrections, which affect the order of the processing of the events, can be switched off in the configuration of the simulator. So it can be guaranteed in many cases that old models behave exactly the same way as with old versions.

Modules:

- The unloading station calls the decision table, when the object is ready to leave at exit 2.
- The disassembly calls the decision table, when a disassembled object is ready to leave at entrance > 1.



46 Release Notes 4.2

This is a list of all relevant changes from version 4.1.

46.1 Standard Functions

46.1.1 User Interface

Models can be structured with the assistance of layers and views.

Parameters can be marked as test value.

In the definition of the model constants formulas can be used.

Junctions, which are attached at special points of certain modules (distribution car, conveying circuit), can later be moved.

When connecting failures with modules, the components must be selected no longer individually. With the assistance of a selection rectangle (rubber band) whole areas can be assigned to the controls.

46.1.2 Modules

In an accumulating conveyor 2 objects with different lengths can be administered.

The conveying circuit received a clearly improved animation. Additionally an extended statistics for this component is led.

46.1.3 Controls

46.1.3.1 Failures

In failures single modules can be disturbed by random.

Reference failures possess a factor, from which the referenced failure can be included several times when calculating the availability.

46.1.3.2 Transport System

Vehicles can be directed through the storage module. Thus a simple turbulence of the vehicles is possible.

Tracks can be marked as waiting position. This simplifies the definition of waiting sections.

46.1.4 Work Plans

The work plans can be examined by decision tables.

46.1.5 Animation

In the bitmap animation it can be selected between covering representation and transparent representation.



46.2 Decision Tables

46.2.1 Usage

Within the initialization area arbitrary actions can be implemented.

The editor has received further window, in which expressions can be supervised.

The expressions can be of several lines.

With the output into files (datprint) the separators for fields and decimal points can be parameterized.

46.2.2 New Functions

46.2.2.1 Quicktable

In the instruction *SearchTable* it can be looked for texts.

46.2.2.2 Work Plan

There are several functions, with which information of work plans can be called up.

46.3 Programming Interface

46.3.1 Excel - Interface

The command set to inquire simulation results was extended. So the results can be queried from transport systems as well as from object turn-around times.

46.3.2 COM - Interface

The COM interface possesses methods to affect the representation of junctions while generating models automatically.

46.4 Changes and Corrections

For compatibility to previous versions corrections, which affect the order of the processing of the events, can be switched off in the configuration of the simulator. So it can be guaranteed in many cases that old models behave exactly the same way as with old versions.

Decision tables:

- The determination of the entrance/exit by decision tables supplies the first entrance/exit with the disassembly and/or storage, from which an object enters/leaves the module. Since this can happen at the same time parallel through several entrance/exits, this could lead to errors. Now the regulation takes place accurately.
- If in the simulation parameters all stochastic cycled times should be fixed, now also with the random functions of the decision tables the average value is returned. This does not apply to the random function *dice*.

Animation:

- The algorithm for recording the AVI - file has been revised. Recording now is five times faster and the created file is approximately five times smaller.



47 Release Notes 4.1

This is a list of all relevant changes in version 4.0b/c (Without engagement on completeness).

47.1 Standard Function

47.1.1 User Interface

The navigation in the model is simplified clearly by the use of the mouse wheel. With pressed mouse wheel the work area is shifted. Turning the mouse wheel zooms in or out of the work area.

47.1.2 Modules

The module multi-functional work station combines the behavior of work station, assembly and disassembly. Depending upon work procedure a different behavior can be defined.

Modules can be assigned attributes in a parameter dialogue. Thus self defined parameters in the simulation can be simply used.

47.1.3 Controls

47.1.3.1 Measuring Element

The measuring element offers the possibility of measuring values in different kinds of evaluating and of plotting them statistically. So also user-defined statistics can be defined by the use of attributes.

47.1.3.2 Transport Systems

Different speeds can be assigned to the vehicles in a transport system. These can be used then when driving through tracks instead of the module specific speed. To the correction of areas, in which different speed is driven, these can be corrected in tracks over a percentage value.

In loading station now charges can be loaded with different goals. These are then driven off in the order of the loading. At the point of unloading it can be specified whether all charges of this goal will unload there or only those, which are accessible in the order.

The number of transportation to a goal can be limited.

47.1.4 Working Plans

Working plans, which could be used so far only as ASCII - file, can be parameterized now in the user interface of Dosimis-3.

47.1.5 Animation

The data flow in the animation can be logged as AVI - files. These files can be played then with an external Player (Windows Media Player, Power DVD...), without Dosimis-3 on the computer installed.



47.1.6 Miscellaneous

For compatibility to previous versions some correction, which influence the sequence of work at the events, can be deactivated by the configuration of the simulator. So it can be guaranteed in many cases that old model behave exactly the same as calculated with the old version.

- With the shuttle and turntables with only one entrance this is selected automatically.
Those drive into the basic position takes place then also automatically.

47.2 Decision Tables

47.2.1 Usage

In the syntax dialogue the selected path is indicated. But it is not taken over with a double click, since this led frequently to an unintentional data transfer. A taking over of the data is made now by the Button *insert*.

47.2.2 New Function

47.2.2.1 Quick Table

In the instruction SearchTable can be defined comparison operation as text („==“, „<=“, ...). This makes the instruction more transparently.

There are 2 new instructions, in order to copy and/or move lines of a Quicktable into another.

47.2.2.2 Log Window

Within the log window can be searched for texts. Additionally a filter can be activated, so that only texts, which contain a given search text, are represented.

47.3 Programming Interface

47.3.1 Excel-Interface

The name of the model can be queried. This can be evaluated with the regulation by experiments.

The instruction set was supplemented to the parameterizing of controls. The parameters of connector, turn-around time measuring module, monitoring and decision tables can be exchanged now over the interface.

47.3.2 COM-Interface

The COM-interface of Dosimis-3 was extended. New functions were added to creating and placing of modules and controls.

47.3.3 Programming Interface

In the interface library more than 50 new functions for the treatment of the Quicktable, attributes, for the component construction and the programmed controls were added.



48 Release Notes 4.0b/c

This is a list of all relevant changes in version 4.0a (Without engagement on completeness).

48.1 Changes and Corrections

Error corrections:

- After an experiment, which was started from the Excel - interface, the simulation results were not read in again.

The determination of the new type in the multiple work station led to a crash, if this object is not defined in a subsequent work procedure.





49 Release Notes 4.0a

This is a list of all relevant changes in version 4.0a (Without engagement on completeness).

49.1 Changes and Corrections

The dialogs of controls have been unified. These posses an input field for comments and can be switched of as info elements.

Elements, which are reference by a global decision table, are shown in a special way when activating info connection.

When searching for files the type of file can be selected. This affects for example quick tables d files in sources.

In analogy to the transfer of throughput time in assemble and disassembly stations this situation can also be mentioned in the turnaround time measurement.

With serial simulation all opened models can be calculated sequentially.

The simulation can be started from the simulation parameter dialog directly.

During online simulation or animation breakpoints can be enabled. This can be activated later on.

The colors of the results diagrams can be adjusted in a separate dialog.

In the editor of the decision tables the parameters of functions are overtaken also with the context sensitive selection..

The decision tables are evaluated after each event in a activating element. In the interest of short simulation duration these calls can be disabled. For a better efficient a call of decision tables is added after the complete entrance of an object.

In the log window a dialog exists, from which the output can be searched. Additionally the search text can be used as a filter, so only that output is displayed, which refers to the search criteria.

Initialization and finalization in the programming interface is done in several steps, because only such a sequence can be determine during initialization, that special information's are available. Further on the interface of decision table calls has got two additional function for initialization and finalization.

In interface library is extended with functions for using quick tables.

Error corrections:

When using varying times for object generation in a source it could happen, the distances can become very small. This effects specially intervals with no distribution in combination with failures during this time.



After re initialization of the display, for example after activating the screen saver, it could happen, the module palettes are nor shown properly.

In zip archive files with special characters have been added by failure.

The animation the worker did not work properly, when a prerun time is set. These have been drawn first after their first event.

For compatibility to previous versions some correction, which influence the sequence of work at the events, can be deactivated by the configuration of the simulator. So it can be guaranteed in many cases that old model behave exactly the same as calculated with the old version.

- The additional decision table call after the object entrance can affect the events during simulation.



50 Release Notes 4.0

This is a list of all relevant changes in version 3.2b (Without engagement on completeness).

50.1 Standard Function

50.1.1 User Interface

The old module pallet was replaced by a two new module pallets. For a better overview a separation was made between flow of material modules and controls.

Dialogues can be configured by a selection of characteristics of the module. This was realized first by the example of the work station.

By the properties to the user interface warning stages can be specified for the consistency check. Those supplies additional referring to possible model errors during the parameter examination. Additionally module properties can be specified (personnel, transport system). In the simulation properties it can be specified that error situation are logged in the trace file. Thus the analysis of the error situation was substantially simplified.

50.1.2 Modules

In the assembly different mounting methods are possible. Additionally wildcards are supported.

There is a new module of the type accumulating conveyor 2. This corresponds essentially to the function of a accumulating conveyor. It has however a separate length of the last segment and loading path. Additionally sensors can be defined, which release an event similarly to light barriers, from which decision tables are activated.

There is a new module of the type multiple workstation. This essentially corresponds to a work station, however it possesses a capacity. This serves the parallel treatment of several objects. As long as objects are brought in, start of work cannot not be begin. The first object determines the operating time. After the treatment all objects leave the station. Parallel to it the next objects can enter.

50.1.3 Controls

50.1.3.1 Failures

The mean time between failure of random failures can be referred to active time. This means that in the reference module an object must be found. Alternatively this can be limited also to the operating time.

Throughput-dependent failures can be defined stochastically. Instead of a firm number of items this can be randomized now by a stochastic process.

In a sub dialog a separate point of starting time can be indicated for random and throughput-dependent failures.



50.1.3.2 Work Area

With the parameterization frequent actions can be made much easier by the switches completing and/or taking over. For each module, which can implement object processing, an entry in the task list is made. The change conditions for homogeneous tasks are taken over from the current.

Interrupted tasks can be assigned with priority (e.g. after a break).

By the limited number of workers in breaks the number worker, which pause at the same time, can be limited. The remainder must shift the break, until the minimum number is again reached (e.g. personal allowances).

Fixed and periodic pauses can be visualized in an overview (by zoom also in detail).

In groups of workers a work area several teams with their own shift model can be defined. Thus the modeling is simplified clearly by multi-shift work with different personnel strength.

By the new activity supervising personnel can be requested for capacity monitoring. If an object enters the area, the supervising of this area is activated. If the area is again empty, the supervising is waived. For supervising the indicated number of workers is requested. The worker can be called up however at any time to another activity.

Work areas can be deactivated by failures. The personnel starts then immediately the break. Special strategies are not permitted.

The work area dialogue is also opened, if no module is added. Some activities cannot be completely parametrized. Later on changing's are however possible.

50.1.3.3 Capacity Monitor

In capacity monitoring also a lower bound can be defined apart from the capacity. The release of the entrances takes place only when the total allocation falls below the lower bound. Additionally personnel can be requested for supervising.

50.1.3.4 Monitoring

A protocol for variables and statistics can be selected. The expenditure takes place in the protocol window every time, the value changes.

50.1.3.5 Transport System

By the local transport controls independent transportation areas with their own controls (administration of orders and vehicle scheduling) can be provided. Each area has its own way matrix and vehicle/order administration.

In the statistics the processing time is separately proven. This is meant that the duration of the processing of load in a work station is seized in its own state.

50.1.4 Statistics

When assembling and disassembling the turn-around time can be transferred. When assembling the basic object receives the smallest entrance time of all assembly objects and the basic object as the entrance time. In the disassembly each disassembled object receives the entrance time of the basic object as an entrance time.



In the turn-around time measurement there is a new action *transfer*, which forces this for the measured values when assembling and disassembling.

The results of modules with work portion can be represented common. The result diagrams of work station, assembly, disassembly, loading station and delivery station can be analyzed now together in a diagram.

50.1.5 Miscellaneous

Entries can be shifted within lists. Since often only the first entries are visible, the most important data can be shifted to the beginning of the list.

The cycle time of 0 is permissible, if the type of distribution is fixed, in nearly every situation (e.g. not valid in sources).

Various external calls are incorrect, if the full path contains blanks (call of zip archives...). This is corrected.

With certain configurations of Windows XP an error message came up at the start of Dosimis-3 because of an erroneously reading of entries in the system file. This is repaired.

For compatibility to previous versions some correction, which influence the sequence of work at the events, can be deactivated by the configuration of the simulator. So it can be guaranteed in many cases that old model behave exactly the same as calculated with the old version.

- Disturbed sources had frequently produced a new object at the end of the disturbance. This is corrected, so that the production time is shifted also according the breakdown duration.
- When module are disturbed by decision tables, this could lead to a wrong breakdown statistics of the module, because this passed there frequently during a course of actions within the module (e.g. during the object delivery). This was changed in the regard that the disturbance enters into force only at the end of such a chain of events. This has the consequence that for example the breakdown semaphore is not set immediately after the execution of the action.
- In work areas it could occur, that the new worker are not assigned in accordance with the arrangement rules, but also with alternatives the first found worker of the work was assigned.

50.1.6 Cost Simulation

Costs of various states can be assigned to modules. These divide on into a fixed portion and a time-dependent portion. This is statistically collected for each module.

The costs are summed and given additionally in each case to the object, which leaves the module. Toward end of the simulation the values will not be transferred, if the module is empty. The object costs are evaluated for each object type.

Special cases cost simulation of modules:

Disassembly: For each exit it will be fixed, which proportional portion of the value of the basic object are assigned to the objects leaving these exits. It is possibility to assigned the entire value to just one exit.



Storage: A storage possesses costs of operations and storage cost for each object type. The proportionately operating cost are assigned to all objects in the storage with each exit of an object from the storage.

In work areas personnel costs for each state and qualification can be defined. The costs are summed and collected for each work area separately. If a work is terminated, the costs are assigned additionally to the object, which was worked on.

With the cost simulation several result graphics are provided:

Costs per object (min/max)

Costs pro module (per state)

Costs pro worker

50.2 Decision Tables

50.2.1 Usage

The DT-dialog was revised. This realizes the view onto all decision tables of the model. Additionally the coordinates of the representation can be transferred from one decision table to another decision tables.

50.2.2 Configuration / Parameter

Looking for parameters in decision tables can be activated in the dialog for model/searches. If the searched text is found in the initializations, conditions or actions, every of these decision tables can be recalled and worked on step by step.

In the configuration of the simulator the decimal separator can be specified. The output of real numbers does not take place then with a decimal point, but as fixed in the parameterization of the clipboard properties.

50.2.3 New Functions

By the method Stop work(<type: int>) in modules with processing the current work procedure (kind 0) or the entire treatment (kind 1) can be broken off.

A sub table can now have a return action with return value. The call of a decision table returns this value ($a := \text{subtable}(\text{name}).\text{execute}$).

There are 2 variants of the foreach -instruction. To the one hand to iterate through all modules of a decision table, on the other hand to analyze all objects in a module.

50.2.3.1 Work Area

Work areas support restriction - decision tables. Thus the combination of workers with a work can be released or prevented. For this there are two data types (workers and work) as well as in each case one method on these data types (act_worker.qualification and act_work.qualification).

50.2.3.2 Quick Table

Quicktables can be referenced now also by numbers. Columns of a quicktable can now also be referenced by names.



50.2.3.3 Timer

Timer can activate itself after given intervals (e.g. in accordance with a distribution function). The timer activates all connected decision tables. A timer can be activated also from a decision table. This can be done additionally with a delay: (*timer(name).activate(time)*).

50.2.3.4 Sensor with Accumulating Conveyor 2

For an accumulating conveyor 2 sensors can be defined, which will release a DT while passing. The inquiry is made by a method of the module (*akt_module.sensor*).

50.2.4 Searching Errors

Actions can be deactivated purposefully and simply by marking.

The initializing values are indicated directly during the online simulation.

The function errprint writes data into the error file (*modelname.err*). These error log can be rerouted into the simulation protocol.

50.3 Programming Interface

50.3.1 COM-Interface

The COM-interface from Dosimis-3 was extended clearly. New classes were added for the usage of modules, work areas, controls and decision tables.

The class IDs3Document possesses new functions for creating and connecting elements as well as setting parameters of the elements. Additionally simulation parameters can be set, so that complete simulation models can be generated.

50.3.2 Programming Interface

50.3.2.1 Integrated Editor

An integrated editor serves the treatment of smaller corrections from the user interface. It supports editing and translating. The editor possesses a syntax coloring as well as the possibility to call the translation process directly. With incorrectly translated sources the erroneous line of code can jump to and correct directly.

50.3.2.2 Palette for self defined Elements

The self defined elements possess their own toolbar. From these the elements can be pulled directly on the work area. The selection by name is not necessary any longer. For this there is a new function for exchanging the information for the representation. Additionally an exchange of the version number as well as the kind of translation (Release/Debug) takes place, so that a compatibility between differently versions can be ensured.

50.3.2.3 Module Construction

Dosimis-3 offers the possibility of module construction. For this 2 templates are available. The necessary methods are made available for reading and writing from parameters, consistency check, drawing, object transfer, initialization and event processing.

Additionally there are functions, which inform about the beginning and end of failures, right of way strategies, distribution strategies as well as (manual) activities.



The template work realizes work times, worker requirement and setup times. The sample shows the use of these methods and can be changed afterwards.

The template crossing shows, how modules with several entrances and exits convert object transport by right of way and distribution strategies. The standard strategies are directly usable thereby. Additionally programmed right of way and distribution strategies can be realized.



51 Release Notes 3.2b

This is a list of all relevant changes in version 3.2a (Without engagement on completeness).

51.1 Changes and Corrections

The transport statistics supplements the new type operating time.

Receiving the waiting period by Excel interface for all modules.

The consistency check of the formulas not accomplished, if this refers directly to a Quicktable, which is however not loaded.

Consistency check for the parking place in transport systems takes place only, if a follow-up module exists. This could lead to an incomplete consistency check. The model however was inconsistent anyway.

If no transport parameters are defined, the initialization of the model could not be archived correctly. This is repaired.

Very long module names or model names could lead to a crash during the result representation in the layout.





52 Release Notes 3.2a

This is a list of all relevant changes in version 3.1a/b/c (Without engagement on completeness).

52.1 Changes and Corrections

The statistic of capacity monitoring and Turn-around Time Measurement haven't been closed in some cases. This results in an error during reading these statistics.

Old models with work places haven't been changes correct to the new working place.

During the startup of Dosimis-3 some entries of the registry have been queried in read/write mode. This lead in some cases to error messages with windows XP.

The shortest length of modules is fixed to 24 units. "Degenerated" modules could cause a crash during animation.





53 Release Notes 3.2

This is a list of all relevant changes in version 3.1a/b/c (Without engagement on completeness).

53.1 New Functionality

Many elements possess a standard method. This can be activated by the menu option Edit/Execute. Alternatively this can be activated by a double-click on the element with pressed SHIFT-key.

The dialogues have been structured. The substantial parameters are in direct access, secondary parameters can be supplemented when required.

The input of the parameters can take place by attributes or formulas. All further information you will find in the chapter of input fields.

There is the possibility of limiting the distribution functions. Additionally the distribution can be varied during simulation with source and sinks.

The crossing offers additional functions, so that lift times or circuit times can be considered.

For more flexible export into the clipboard the separators can be parametrized.

There is a new graphic element: Bitmap. Thus for example logos or an entire layout can be placed in the model.

With the parameters of the cost simulation the capital commitment in a system can be pursued.

There is an additional result diagram of the cycle time analysis, by which the model can be analyzed more exactly.

The evaluation module offers the possibility of parameterizing constantly returning result representations so that these are fast in access.

For failures the point of starting time can be defined for a better transient behavior. These are available for throughput-dependent failures and for random failures.

In the decision tables the possibilities have been extended. Beside the integer and real - variables (a-j) further variables can be assigned to modules and objects. These are called attributes and can be used not only in the decision tables, but also directly with the parameterizing of modules.

For the use of larger data sets during the simulation there is the possibility of using simple spread sheets, called Quick tables. By operations on these tables the data can be evaluated, sorted or it can be looked for within these data.

For shuttle and turntables the basic position can be changed dynamically by decision tables.



The element signal display serves similarly the animation texts to visualize conditions in the layout. Contrary to animation texts this is done however by color information.

53.2 Changes and Corrections

Placing of modules is changed. Before the definition of the first base the adjustment can be still corrected. For modules with variable size during placing the entire module is indicated, not only the way points. The expansion ends with the mouse position, not *a little bit* behind it.

With transport systems the manual disposition is extended. After working in a station the further behavior of the vehicle can be substantially better affected by the successive disposition.

The work place in the work area became more flexible. So an allocation can be made, for which work areas and which modules it should be permissible.

In the employment of workers also model constants can be used. This concerns the minimum and maximum number of workers on the one hand. Additionally there is a special constant type, which indicates the worker qualification. This can be used during processing and in the work areas.

The format of the statistics file is changed. The width of the tables is limited by 80 letters, so an output on DIN A4 paper is possible.

The result parameters are distributed for clear representation on several sub dialogs.

The throughput statistics logs additionally entrance and exit, by which the objects enter and/or leave the module. So in modules with several entrances and exits the material flow can be analyzed in detail.

In the online simulation apart from the module parameters also the information of the controls is indicated. These are the list of the vehicles and orders for transport systems as well as the list of the workers and tasks for work areas.

During the input of formulas in the decision tables the context sensitive selection is improved. Thus the selection takes place, if model modules are demanded such as variable, quick tables, model constant....



54 Release Notes 3.1a/b/c

This is a list of all relevant changes in version 3.1 (Without engagement on completeness).

54.1 New Functionality

The conveying circuit is extended by a further parameter. Beside *Circuit stops if output is impossible* the switch *Circuit stops, until object has left complete* exists.

In the animation the first window is centered, if a breakpoint is set and the appropriate module not visible in any window.

The manual activities are visualized separately in the production diagram and status diagram from the automatic work.

With the placing of texts the first set point was interpreted as starting point of the text. This was changed to the left lower corner of the rectangle, independently of how the rectangle is drawn up.

Further functions for the analysis of the simulation parameters are available in the COM interface.

54.2 Changes and Corrections

When the disassembly and assembling the operating points were not correctly announced for analysis in the decision tables.

In the animation disturbed modules were not correctly updated, if the appropriate disturbance were not in the visible area.

In the programming interfaces the initialization of the Simlib was missing before the recording procedure, which leads to an error message.

When in a work area *another activities* was defined, but not immediately be parametrized, this has lead to an inconsistent model.

The deletion of entries in the variable window could lead under certain circumstances to a crash.

Main disturbances, which affect subsidiary failures, operated only correctly, if the subsidiary failure had had an event during the breakdown duration.

Disturbances by decision table were not correctly animated.

When opening the parameters of the source the switch the *determined sequence* was overwritten by *random sequence*.

In some cases the combination of automatic delivery and delivery via DT in a storage causes errors.



With the definition of positions in the conveying circuit it could lead to an inconsistent model.

In crossings with only one input/output the parameter *objects are allowed to stop* was not correctly initialized with simulation start.

After Undo - operations the administration of the programming interface was not reinitialized.

The animation was substantially accelerated, because empty modules are not always been drawn again.

Work areas with inconsistent break were not marked as inconsistent.

The wildcard in working times have not been considered by the excel - data exchange.

In the turn-around time measurement it could occur that objects, which did not achieve the endpoint of the measuring section produced faulty measurements.

The statistics " waiting for assembly " was not led correctly when assembling with multiple objects.



55 Release Notes 3.1

This is a list of all relevant changes in version 3.1 (Without engagement on completeness).

55.1 New Functionality

Multi Document Interface (MDI): Several Dosims-3 models can be opened at the same time. Between these models or between two Dosimis-3 applications sub models can be copied by the clipboard.

The user interface possesses properties, to switch the display of minute values from DD:HH:MM to minutes. Additionally the file size can be defined, where a safety query takes place before reading simulation results.

Elements can be selectively **frozen or hidden**. Frozen elements are further be drawn but not considered during selections.

Object types can be defined with **objecttype variables**. Additionally the restriction of these types to 32000 is cancelled. The limit is now 999999000.

Modules with the length of 0 are possible for the module types workstation, assembly, disassembly, distributor, combining station, crossing, discharger and infeed element.

All Dosimis-3 elements can be assigned with a comment. This can be extended by a multiline note. This note can also be assigned to every parameter of the model, where a variable can be used.

New distribution strategy **Bauschuld**. This forces a deterministic behavior. This strategy is the replacement of the percentaged distribution, if *cycle time fixed* is activated in the simulation parameter.

In the workstation several **work procedures** can be defined. Several different handlings with different properties in one station can so be comfortable defined.

In the module storage stored object can be assigned with a mean storage time. Then an automatic removal takes place and the delivery by decision table not necessary.

In the disassembly each object of the disassembly list can have an individual disassembly time.

The destinations in an automatic transport system can be defined by destination variables.

The loading/unloading times in an automatic transport system can be **manual and by random**.

Unloading stations and workstations possess the possibility, to stop vehicles at the end of an order in a **bypass**. Then these vehicles do not stand on the route and following vehicles can pass this area.



Failures can be defined by a link to a reference failure. The parameters are taken from that reference.

Failures can affect failures. Therefore the user can define, whether e.g. the utilization is defined according the total time or the active time (without pauses) of the machine.

A new statistic module is created, from which the throughput time can be measured. Additionally statistics and graphics are available.

For the capacity monitoring its own statistics and an internal message log are written. As a special case a capacity of 0 for pure statistics is admissible over the total of the modules.

A simulation directory can be defined, where Dosimis-3 creates the files, which are used during the simulation. This is in particular an advantage with data intensive simulations in a network, since the relatively small models can be situated in the network and the data intensive outputs take place locally.

There is a new statistic representation, in which the standard results (actual occupation, throughput...) are represented during the entire statistics period at each statistics point in time.

The results of an optimization run are plotted in the optimization diagram.

During the animation the possibility exists of animating the last steps again backwards.

For presentation purposes there is a further animation mode. In the continuous animation the objects in the conveyors are animated exactly, so that the impression of flowing is supported.

Texts are not defined only by the enclosing rectangle, but can be scaled also by the font size.

In the decision tables the new dialog was made the standard. Additionally several new operations, functions and actions are available:

- **&&, || and !:** The logical operation *and*, *or* und *not* permit to define conditions more easily and reduce the number of rules.
- **For-Loop** for easy iteration,
- **open** to release a junction independent of the number of blockings.
- **disturb_time, block_time:** Restricted activities according the time, which are released automatically.
- **datprint:** Text output on file.
- **dt_failsem, maintenancesem, sum_failsem:** Querying the new semaphores.
- **intpar, floatpar:** Access to the default parameter of a model
- **callfor_wk:** Calling for worker by a decision table.
- **act_object und act_entrance** with self defined distribution,
- **C - function call**

55.2 Changes and Corrections

The new decision table editor is raised to the standard. The old editor is available however further (inclusive documentation). If during pressing the button *edit DT* the CTRL key is pressed, this is activated. This is however no longer continued to maintain.



Items can be moved **without grid**. For this the CTRL - key is to press with the operation. Furthermore graphic diagram operations can be aborted (move, rotate,...) with the ESC - key.

For simplified handling new or extended **Tools** (e.g. processing) are available.

New **search - dialog**! The criterion of the type of item (module, failure, ...) is situated now in the dialog and can be corrected subsequently.

Wildcards are admissible in addition to the object oriented distribution now also during the input of type of object of the handling (WST, ASS, DAS, ...). To different interpretations see the documentation of the type of module.

Infeed element and discharger may have more than 2 in/outputs.

Maintenance is new type of failure, which is separately provided in the statistics. So also intended downtimes (maintenance) can be analyzed separately apart from down-times (failure) and layer times (to trace).

Undefined items are selected after the explicit **consistency**.

The selection in the trace and statistics lists is possible also for work areas, failures and further controls.

The average value of the **AGV- statistics** in the result graphic was always according all vehicles. This is corrected for the area limited by the type of object. Order statistics are now vehicle-referred. Thus different vehicle fleets can be better analyzed.

If **additional information** is activated, in continuous diagrams the breakdown times of the fixed and periodic failures are displayed, if these are selected.

Refined error message during result representation.

Continuous diagrams (e.g. allocation diagrams) can be copied into the **clipboard**, if only one module is selected.

During the definition from references to decision tables tool tips are supported. Thus the items can be better identified.

It exists a new instruction **keyword** in the Excel interface. Since the way of writing is not always unique, so the keywords can be recalled. This is first intended for the module results and work area results, since these keywords cannot be recalled here via export as for example with the parameters.

With the data export via Excel interface a warning takes place, if the name of the item (module, failure...) is not unique. This could lead beforehand to irritations, because the data is possibly taken from the false item.

The functions of the **ISimlib** were completed. The **IEditlib** is omitted. The few functions are implemented in the **ISimlib**, which are available both in the editor and in the simulator.



56 Release Notes 3.0a

This is a list of all relevant changes in version 3.0a (Without engagement on completeness).

56.1 Added Functionality

The input of animation text will be checked for syntax direct after the input.

The model cannot be saved, when spaces are in the name of the material flow system. Otherwise this will lead to problems with the simulation.

After the Online - Simulation the parameters of distributions will not be reset, when the switch *cycle time fixed* is activated. The Online - Simulation cannot be started, when this switch is set.

Pretty - Print continued with failures dependent on throughput, alternating distribution,

The Ergosim - Interface adjusted to the actual version.

56.1.1 International Version

Function Translate for translating the text of the decision tables into the language of the current version of Dosimis-3 (in Global-DT-Data).

Die Excel - interface is expanded with English keywords.

The Viewer analyses the statistic-file correct, even it is written with another language version of Dosimis-3 than the current.

56.1.2 Corrections

Initialized storages didn't work correct.

The pallet changer runs into a deadlock.

The events *exiting* and *entering* of failures depending on throughput have been changed in the selection box.

With throughput dependent failures the format of the throughput was transformed to HH:MM, when the value is greater than 60(Internally the calculation was correct).

During the combination of models with connector or decision tables some conflicts have not been solved correct.

Sometimes the automated path finding was not read correctly.

In Bitmap - Animation only object type lower than 10000 have been allowed. This value is raise up to 32000 (the maximum value of Dosimis-3 object types).

The Online - Simulation crashes sometimes, when the Variable - Window was active.





57 Release Notes 3.0

This is a list of all relevant changes in version 3.0 (Without engagement on completeness).

57.1 Added Functionality

57.1.1 Interaction

Context menus

Toolbars for Zooming, Modeling, Results

Short-Cuts in Simulation: F7,CTRL+F7, Alt+F7

57.1.2 Representation

The elements can be replaced by self defined Symbols.

Triangles in Nodes and elements are filled for a better visualization in the animation of the nodes.

Placed elements with more than two way points (conveyor, ...) can be replaced.

57.1.3 Default

Variables are allowed for defaults.

57.1.4 Dynamic Element Linking

The count of entrances and exits increase automatic, if more than 2 neighbors have to be connected.

57.1.5 Automatic Guided Vehicles

Display cont of vehicles in the parameter mask and selection of the initializing elements.

Search element according to AGV-destination.

Show destination address in the layout.

57.1.6 Export

Version 2.3

Create Zip-Archive

Mail model

57.1.7 Failures Dependent on Throughput

With failures dependent on throughput the possibility exist, to define the event can be defined, which is responsible for counting.

57.1.8 New Parameter with Right of Way and Distribution Strategy

right of way strategy

distribution strategy

57.1.9 Validation

Elements can be added complete to the action protocol by a switch, without changing the trace list.



57.1.10 Documentation

In addition to the new function the description of the Source is structured new and continued.

57.2 New Functionality

57.2.1 Decision Tables

Copy and move of decision tables.

The decision tables work with the parser. This means, that some decision tables become inconsistent, e.g. the keywords are not allowed at naming of variables.

New windows for text output

Connection to the online simulation

57.2.2 Decision Tables Dialog

This version includes a beta version of a new decision table editor. This can be activated by clicking on the button *edit DT* and pressing the control key (STRG/CTRL).

57.2.3 New Disposition of Worker

For a faster disposition of the worker a new algorithm is implemented.

57.2.4 Undo

Operations can be undone.

57.2.5 Online - Simulation

For analyzing the behavior the simulation can be done in an online-mode. All relevant information are during the simulation available for analyses.

57.2.6 Animation

During animation break points can be defined, where the animation can be suspended. Further methods for analyzing the model are available.

57.2.7 Programming interface

For additional definition of controls modules can be implemented. This works with Visual-C++ and the programming language C/C++.

57.2.8 Excel interface

New export Standard

New function set randomnumbers for changing the start value of all random processes.

57.3 Corrections

57.3.1 Start Value of Random Numbers

Every decision table, the net attributes, the work plans and the automated guided vehicles (only with Battery - charging) got their own start value. Event accord to these processes have been initialized with the global mfs - start value. This will lead to different simulation results, especially after extensions to the model.



Therefore the simulation results may differ to the results of the last version, because the new version behaves, as is (once a time) all start values of the random process of work plan and decision tables have been changed.

57.3.2 Statistic Shuttle and Turntable

The unload time of the element (Relation of unload path and (unload -)speed) will be added to the state unloading time. Till now this time was added to the transport time. In the state Unloading time was only entering of the following element. *Unloading time* contains from now on both parts *unloading time of the element* and *Entering the following element*.

57.3.3 Continuation of Generating Events

The behavior of the elements at identical events (reaction on waiting for leaving, exiting,...) is made unique. Therefore some differences in activating of the elements can occur, which should invoke only small differences of the total behavior. In principle the result should not differ. But it can lead to a for example activating of decision tables at states, which not have been recognized during the implementation.



58 Release Notes 2.3a

This is a list of all relevant changes in version 2.3a (Without engagement on completeness).

58.1 Added Functionality

58.1.1 Interaction

During the placement of elements with more than 2 way points these can be deleted.

In the mode Info/connections the relations of workplaces are visualized.

Short-Cuts for search functions: CTRL+F: Search, F3 redo search.

Delete all way points instead of change way points, if SHIFT-key is pressed.

58.1.2 Representation

Output of DT initializing in pretty print.

Black legend, when no data available (occupation diagram).

Display worked objects also for shuttle and turntable,

Marking of pre run time in diagram via option addition information.

Representation of initializing of net attributes in global DT-Data.

The simulation window from now on not top most on the desktop.

58.1.3 Dialogs

Viewing of the source file

Without Statistics also for pauses and work area pauses available.

58.1.4 Calculation

Consistency check for all elements, because for example after the copying of a model the model is probably inconsistent (e.g. source file).

58.2 New Functionality

58.2.1 Strategy

Right of way strategy with check entrance.

In the distribution strategy it is selectable, which successors should be examined for the new destination. Elements, which are connected with an unblocked junction, elements without failure, element with free capacity and elements in the state ready to overtake. This decision have been made the whole time, but implicit and in dependence of the strategy, without selection.

58.2.2 Dialogs

View vehicles and selection of agv-init-elements

Export Version 2.3

Search elements according to agv destination

58.2.3 Animation

Protocol windows for output of animation text

View objects in element during animation.

Animation of conveying circuit



58.2.4 Results

Viewer

Tree view of the SLG-structure in the viewer.

Copy of areas(**Ctrl+C**, or. Right mouse click in the Arrangement) in the Viewer (at the time 100 lines), into the clipboard for further use for example in Excel.

Printing from the viewer.

Arrangement of the diagram according the selection

Selection box for moment of statistic

Representation of mean values for several diagrams.

Expanded time representation in the result graphics

Results graphic with Muster or Date - input.

Language independent reading of the slg - file.

Selective reading of the trace - file (CRTL - key pressed)

Worker employment diagram

Production diagram

diagram of absolute occupation

58.2.5 Excel interface

New option with keyword start: Start batchfile.

Output of Integer and Float - Variables.

Export some parameter of failures

58.3 Corrections

58.3.1 Random Processes of DTs

Random processes have been initialized with the random number start value of the MFS. This leads to different results, when the mode is expanded.. From now on each decision table has its own start value.

58.3.2 Several Order in Agv Strategy

With several orders for one agv the calculation of the driving time with the strategy shortest path only the secondary destination of the actual order was taken in advance. This is corrected to secondary destination of the last order.



59 Release Notes 2.3

This is a list of all relevant changes in version 2.3 (Without engagement on completeness).

59.1 New Functionality

New element "connector" for combining parts of models.

59.1.1 Extension to Failures

Without statistics

passive

easy input

59.2 Easy Handling

View of relations in the model

Pretty Print extended

Selective Export with excel interface

59.2.1 New Selection of Elements

By property

By name (3 modes)

59.3 Decision Tables

The combination of modules at including of models more flexible.

Support in editor

Debugging-Information

Animation

Detailed information in case of error situation

59.4 Support of Validation

Random times fixed

Passive failures

File in case of simulation errors

temporary simulation

expanded consistency check

59.5 Support of the Clipboard

Graphics

Results

59.6 Support During Experimentation

Module for Optimization

Results via Excel interface



59.7 Graphic

59.7.1 New Graphic Types

[Filled polygon](#)

[Arrow heads](#)

[Line types](#)

59.7.2 [Graphic-Toolbar](#)

59.7.3 [Preview for Groups](#)

59.8 [Bitmap-Animation](#)

59.9 [Element Symbols](#)



60 Release Notes 2.2

This is a list of all relevant changes in version 2.2 (Without engagement on completeness).

60.1 New Functionality

60.1.1 Decision Tables

- Editor
All conditions and actions are total editable. Corrections can be made later on without much efforts.
Error check extended
- Inconsistent during simulation are announced
Output of the call stack in case of errors in the .chk-file.
- Exchange of elements and nodes, completion after deletion of elements

60.1.2 Interaction

- Shortcuts
F9: Linking active
F10: simulation start
F11: animation
- Information will be shown (ToolTips)
- Fast Interaction in the animation with the help of Toolbars.

60.1.3 New Strategy in Source

Bauschuld algorithm

Passive Source

60.1.4 Results

State diagram

Work plan statistic

Results in the layout (additional with head and foot area)

Output in file redesigned (and corrected).

60.1.5 Work Area

- New strategies for change of tasks (working with the work item)
- Next task in area
next working operation according the work plan.
Worker remain at the station (according the strategy), even if the next object does not need a worker. This may cause a deadlock. The worker will be released, if at the station the of work (without worker) starts.
- Work plan
If an object with work plan reaches a disassembly and a disassembly list exists, the work plan will be assigned to the first object in the disassembly list
If an object with work plan reaches an assembly, the work plan will be assigned to the base object.



60.2 Corrections

60.2.1 Random Numbers

The random numbers have been recalculated when reading the model. This can lead to different results.

The random numbers of 2 different elements could be identical after the copy process. This leads to same behavior of both elements.

60.2.2 Animation

The animation with pre run crashes, when a storage exists in the model.

60.2.3 Work Areas

Pauses could not be parameterized with worker. Therefore maintenance could not be parameterized.

60.2.4 Elements

The automatic pathfinding at distributors was erroneous. This leads to a deadlock.

60.2.5 AGV

Blockades could be greater than 100 %. Blockade parts of agv's are recalculated at each statistic moment.



61 Introduction to the Tutorial (Part 1)

61.1 Structure of the Tutorial

Using simulation technology to answer logistic questions demands powerful simulation Tools, which are simple to use. DOSIMIS-3 measures up to both demands and therefore it is deployed in several ways in industry - simulation technology is nowadays moving towards education.

The educational version of DOSIMIS-3 contains a reduced scope of service regarding both the size of a simulation model and the supply of interfaces and modules to create extensive control logics. Information about the complete scope of service will be supplied on demand.

DOSIMIS-3 requires the operation system Windows 7/Vista or XP. For the version only the least configuration is presupposed of the used operating system (Windows 7 or Vista: 1-Gigahertz (GHz) processor with 1 GB RAM; XP: Pentium 300 und 128MB RAM). For working with the 3D component of Dosimis-3 a DirectX - graphics adapter is recommended. A free hard disk size of approx. 200 MB is recommended in order to examine smaller models without complications. Bigger models achieve however very soon a use of several a hundred megabyte hard disk size for the statistics. For the graphics a resolution of at least 1024x768 pixel is expected. However a resolution of 1280x1024 or higher is recommended.

This tutorial shall enable the user to create models himself, to edit values of parameters and to modify the model. Furthermore this tutorial acts as guide for designing a seminar of a training course in simulation, including the run of experiments using a predetermined model in chapter 4.

Chapter 2 will provide a short overview about the philosophy of creating a model, chapter 3 describes all steps from the creation of a model till the initial results. Furthermore a practical example will be illustrated.



61.2 Installation of DOSIMIS-3

The installation of DOSIMIS-3-Windows is very simple.

Installation from CD:

Usually the installation is started automatically when the CD is inserted. If this does not happen, carry out the installation by starting the program **Setup.exe** from the Explorer. It can be found in the root directory of the CD.

Installation from floppy disk:

Insert the first floppy disk into the disk drive and start **Setup.exe**.

Installation from the internet:

Download the DOSIMIS-3 installation file from the Internet (www.sdz.de) and save the file on your hard disk. To start the installation of DOSIMIS-3, run this file by double-clicking it in Explorer for example.

Installation from an e-mail (one file):

Save the file from the attachment on your hard disk. Start this file by, for example, double-clicking it in Explorer to start the installation of DOSIMIS-3.

Installation from an e-mail (several files):

Save the files from the attachment into one directory on your hard disk. Start the program **Setup.exe**.

Start DOSIMIS-3 after the installation. The installation is then completed.

While installing with Windows 7 / Windows Vista / Windows XP you must have **administrative rights** so that the necessary entries can be made in the system files. Otherwise you may install only a demo version. Standard users are not permitted to make any entries in the general start menu, such that no icon for DOSIMIS-3 can be created.

After starting the setup, further queries may appear, in which basic settings usually have to be confirmed. During the first installation, it is recommended to carry out a full installation. Then start DOSIMIS-3 with administrative rights. The installation is then completed.





62 Task

62.1 Philosophy of Modeling

Logistic systems can be analyzed very easy with the help of a simulation tool, if the tool offers the necessary elements to create a model. In DOSIMIS-3 this elements are

- **modules** and
- **objects**.

The underlying object-oriented philosophy of modeling assumes that buffers and conveyor belts, work stations, reject gates will be represented by **modules**, which reproduce the behavior of this elements within the required accuracy. A module in DOSIMIS-3 has a specified process logic, respective parameter records and predefined standard statistics.

Objects are used to describe movables like work pieces or palettes (even maybe information). They are marked by numbers, that identify an object type. A model is created in several steps:

- place modules in the DOSIMIS-3 window
- edit module parameters (data)
- connect the modules
- define parameters of simulation
- run the simulation
- look at the results either by watching the animated material (object) flow or see the result statistics.

Handling of objects is defined through the parameter-input mask. Every source contains information about the objects entering the system. All other modules have information on how to handle each object inside the module. In this manner, data input is possible in a clear and compact manner.

62.2 Exercises

The following example is taken from a project and it contains all features of a simulation study. The base for the model is the layout of a planned production hall. A lot of assumptions have been made that have to be checked by simulation.

62.2.1 Model

Consumer goods are produced on the production line of an electrical manufacturer. There is also a sub-area where the goods are inspected and where minor repairs can be carried out. The material flow is illustrated in Figure 2.1. A preliminary production area supplies our production line with two different kinds of product groups (randomly distributed) in intervals of one minute. Both kinds of goods enter the system at a receiving buffer and are fed to the two work stations by shuttle. But there is a fixed assignment of each kind of goods to the individual work station, because a full flexibility cannot be operated economically due to high investment needs.



Due to the processes being technologically difficult to control, a high quantity of rework is accumulated. The parts to be repaired are fed back to the work. Setup times of considerable importance arise in this case.

Layout of the production system

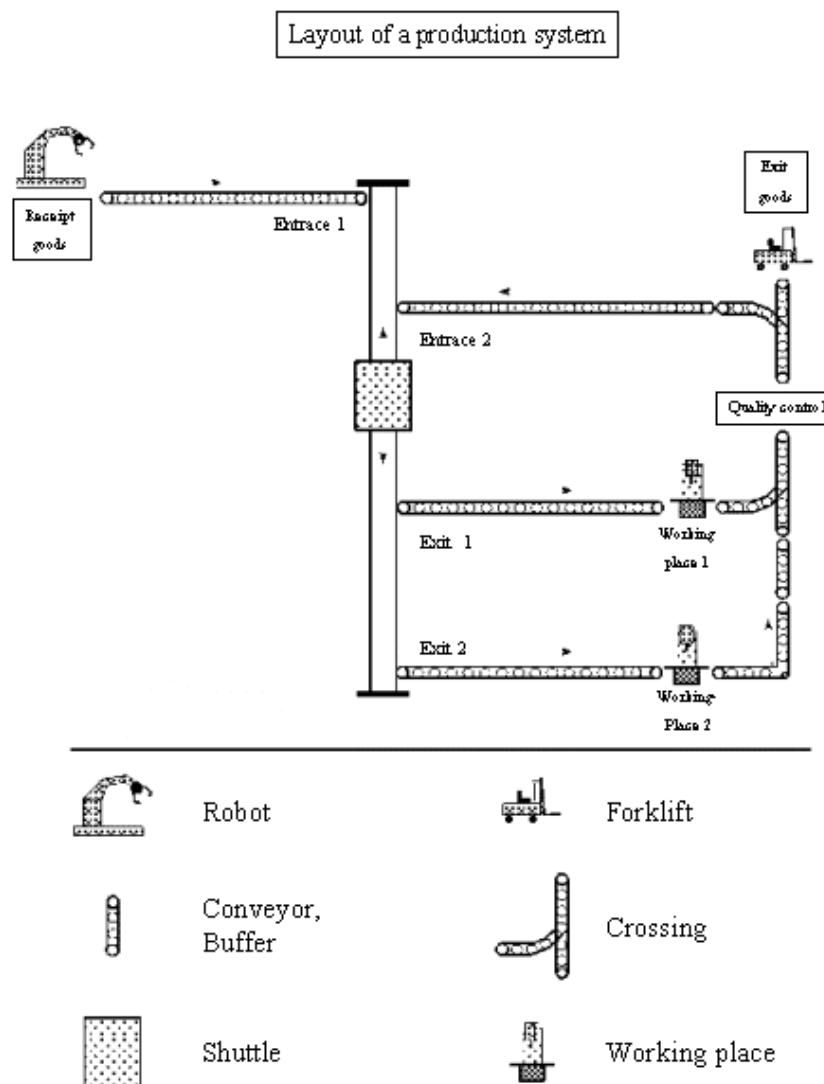


Figure 2.1.: Example of a practical application

62.2.2 Question

Following questions have to be answered:

- Is it possible to achieve the desired throughput of 60 parts per hour (on average)?
- Where are weak spots in the system?
- How high is the utilization of the work stations?
- How much is the utilization capacity of the shuttle?
- Does the mode of operation of the following production line have effects on the utilization of the work stations?



The questions are supposed to be examined under the assumption that failures of the conveying system do not matter. The employed machine operators are supposed to be constantly available. The usage of a standby-employee system may be necessary.

62.3 Database

Data showed that it is possible restrict our consideration to two kinds of products. The layout can be created in small steps. There is a definitive range of supplementary buffer space available. However, it is not possible to forge the use of the shuttle, because the receipt of goods and the two work stations are placed in two halls.

Following data were collected:

62.3.1 Source

The Source (SOU) is an interface that feeds products into the production area under consideration. Products of type 1 and 2 appear in random sequence. Both products are equally frequent. On average products arrive every 60 seconds (normally distributed with a variation of 5 seconds).

62.3.2 Accumulation Conveyor

The accumulation conveyor (BFS) is a conveyor and buffer element. Each accumulation conveyor has a velocity of 0.2 m/sec. The length of a component carrier is 1 m.

There is a different buffer capacity for each conveyor.

- 10 parts after the source,
- 2 parts before the work stations,
- behind work station 2 there is an edge conveyor and a buffer with a capacity of 4,
- 3 parts on the feedback conveyor (for rework).

62.3.3 Shuttle

The shuttle (SHU) is a transport element (vehicle) moving on rails. It picks up loads at a loading station and carries them to unloading stations by means of a vehicle. The loading distance for component carriers is 1.1 m, the unloading distance is 0.1 m, and the velocity for loading and unloading is 0.2 m/sec.

Velocities are specified as follows:

- Maximum speed: 1 m/sec
- average velocity to position: 0.1 m/sec

Following local controls are planned:

- Source has priority
- Product assignment to the exits.

62.3.4 Work Station

The work station (WST) allows reproduction of both manual and automatized working environments (for example robot). Work stations have a length of 1 m. Conveying speed to feed palettes into the station is 0.2 m/sec.

The following intervals are planned:

- Processing time: 80 seconds normally distributed (standard deviation 5 seconds)
- Set-up time: 120 seconds total (necessary for rework)



Rate of rework is 15 %.

62.3.5 Combining Station

The combining station (COM) is used for reproduction of points that merge different material flows. The combining station is 1 m long and conveying speed is 0.2 m/sec. A FIFO (first in first out) strategy is suggested.

62.3.6 Distributor

The Distributor (DIS) is used for reproduction of points that separate one material flow into several different flows. It contains the same conveying parameters as the combining station. The distribution strategy results from the destination of the product (sink or rework).

62.3.7 Sink

The sink (SIN) is an interface to the environment, where products leave the system and are delivered to further production areas. The average departure time at the sink is 55 seconds and exponentially distributed.



63 Modeling of a Production System

63.1 Entry

If installed, you will find DOSIMIS-3 in the Program folder of the “Start” menu of your Windows desktop. Please select **Start/Programs/DOSIMIS-3** from your Windows desktop to start the simulation tool. This simulation tool was implemented according to MS-Windows conventions as much as possible - that is why this tutorial will contain additional remarks regarding handling of Windows.

The first step in creating a new simulation model is to select **File/New** from the menu bar. This opens a new window for the new model which is named “Dosimis-3-1” by default. You can change this name by saving the model with a new name if using the **Save as** option from the **File** menu. Please do not forget to save your model regularly.

The size of the main window of DOSIMIS-3 and of the model window (Dosimis-3-1) can be changed by using the window buttons “Maximize” or “Reduce” in the upper right corner of the window.

If smudges or incomplete pictures appear in your DOSIMIS-3 window, please use the “F5” button on your keyboard to refresh the screen.

As next step select **View/Modules palette** from the menu bar. A new window containing “arrows” and “symbols” subsequently appears. The arrows point in the direction of material flow (orientation) and the symbols represent the modules. Now there are two opened windows, with which the user has to work, the “*module window*” (small) and the *work area* named “Dosimis-3-1” (large) inside the main window of DOSIMIS-3.

To place the source (SOU) in the workspace window of your model, first select the orientation (right: →) on the module window and then click on the source symbol. (When pointing at a symbol with the mouse, the name of the module appears under the mouse pointer and a short description of the module is displayed in the status bar of the main window.) Now place the cursor at the designated point inside your workspace and press the right mouse button to release the source and to fix it on the workspace. The source is now selected and therefore its symbol is displayed in red color. The work area contains an invisible grid, so modules only can be placed depending on this grid. So if your action failed, please try again. If the wrong module or orientation has been chosen, then you have to delete the last entry. Please select the module if necessary (red symbol => selected), press the “Delete” button on your keyboard or select **Edit/Delete** from the menu bar and confirm this action by clicking the “OK” button of the dialog window that appears.

Important: There is a grid on the work area, which means that modules can be put on the work area only at fixed distances. If you want to move a module, then click the module, keep the mouse button pressed while moving and release the button when the module reaches its final position.

The next module to be placed in the model is an accumulation conveyor (ACC). The accumulation conveyor is placed similar to the source - but the ACC is a module with a variable shape - therefore act as follows: select the desired orientation and the ACC symbol



on the modules palette, and then click the source in your model to fix the initial point of the ACC. Now place the mouse cursor a little bit right of the desired end point of the ACC and click the left mouse button to fix this point. After this, please click the right mouse button to leave the graphical edit mode of the ACC. As you have seen, the initial point of the ACC was placed a little bit right (two grid points) of the source you clicked on. In the same manner the end point has been placed (two grid points) left of your cursor. This behavior is to help you to place an ACC between two stations.

Now select the orientation down (\downarrow) and the shuttle (SHU) on the modules palette and place the initial point of it at the right end of the ACC. Now place the cursor about 5 grid points below the initial point and hit the left mouse button to fix the end point of the shuttle and to leave the graphical edit mode. The following modules are now placed in your work area. (Figure 3.1):

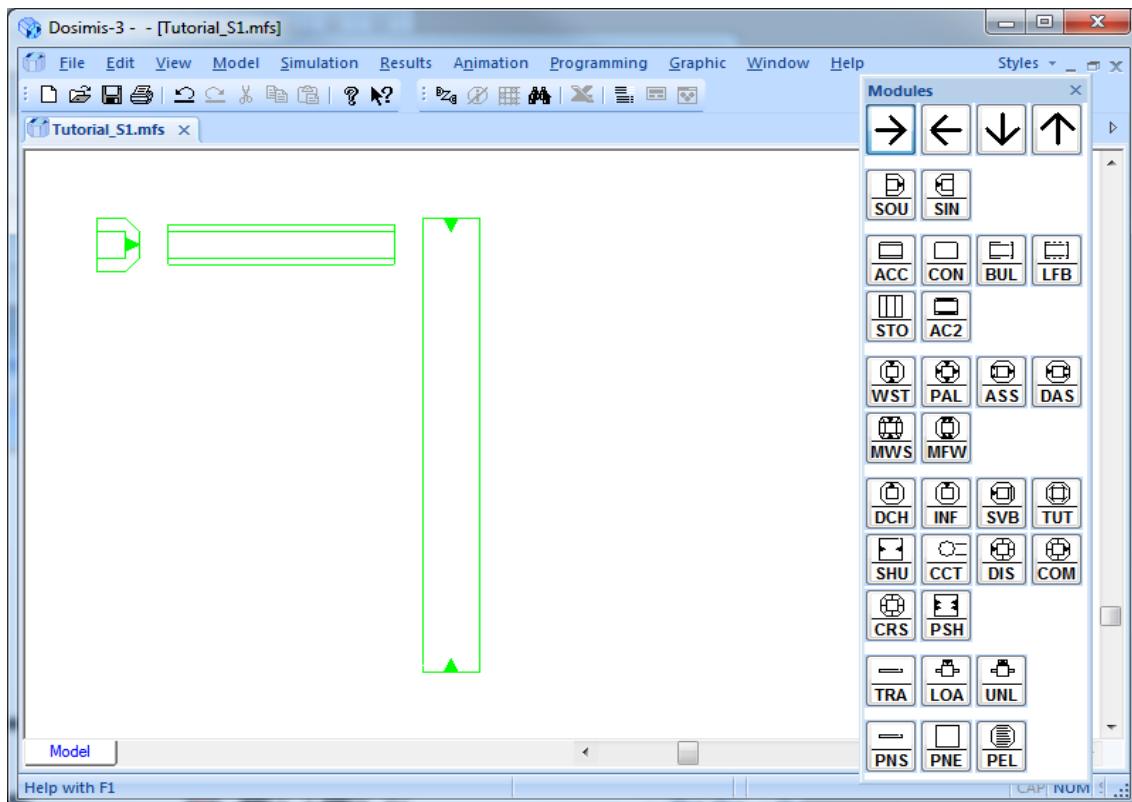


Figure 3.1: Placing the first modules

Other modules will be placed the same way:

- one accumulation conveyor (ACC) at the right side of the shuttle - close to the middle,
- one work station (WST) at the right end of this accumulation conveyor,
- one further accumulation conveyor (ACC) at the right side of the shuttle - close to its end point,
- one work station (WST) at the right end of this accumulation conveyor (below the first WST),
- to create the edge conveyor at the right side of work station 2, perform the following steps: select orientation right (\rightarrow) and the ACC module on the modules palette, then click with the left mouse button on WST 2, next place the cursor in the middle of the green-colored ACC symbol in your model and press the left mouse button again, now



place the cursor one grid point above and press first the left and second the right mouse button.

Now your model looks like figure 3.2:

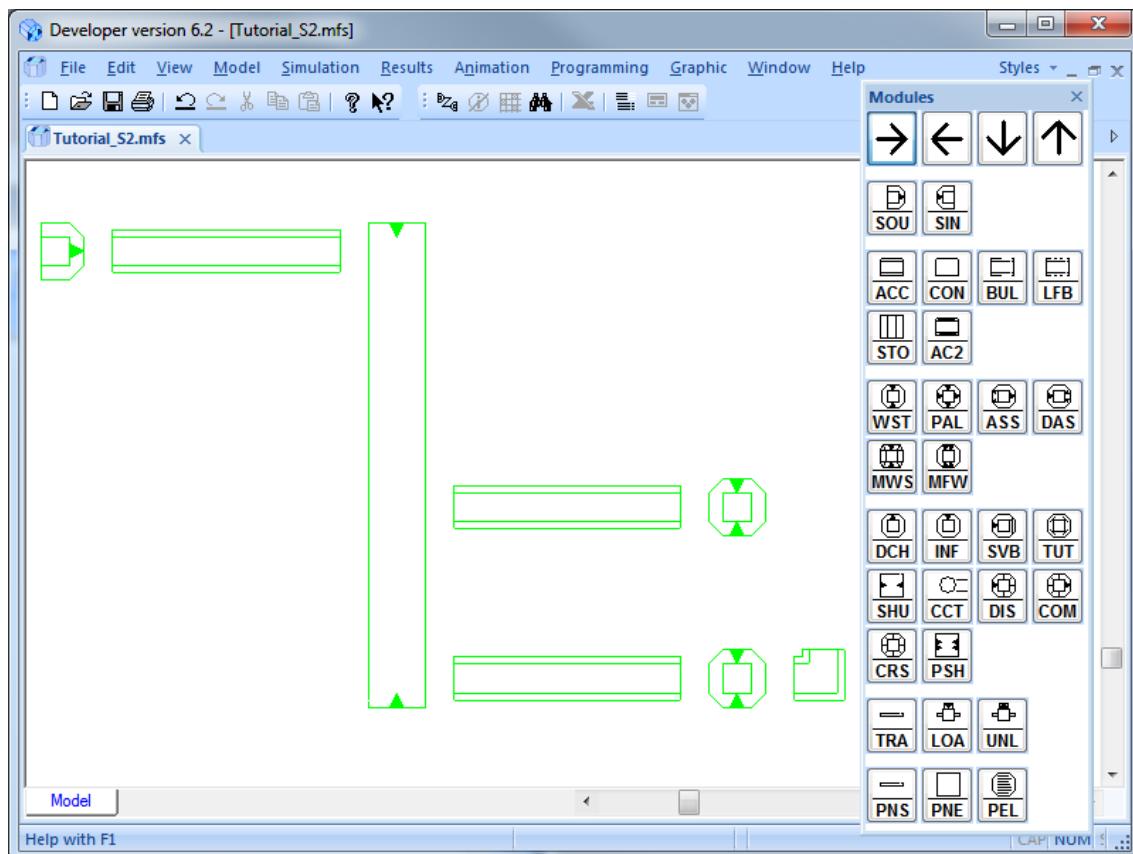


Figure 3.2: Placing of further modules

To optimize the view of your model, select **View/Zoom .../Model** from the menu bar of your DOSIMIS-3 window.

Before placing the next 4 modules, select orientation up () on the modules palette.



Then place the following modules one after the other into your model:

- Accumulation conveyor (ACC) (length: 1 grid element)
- Combining station (COM)
- Distributor (DIS) and
- Sink (SIN)

Now your work area should look like Figure 3.3

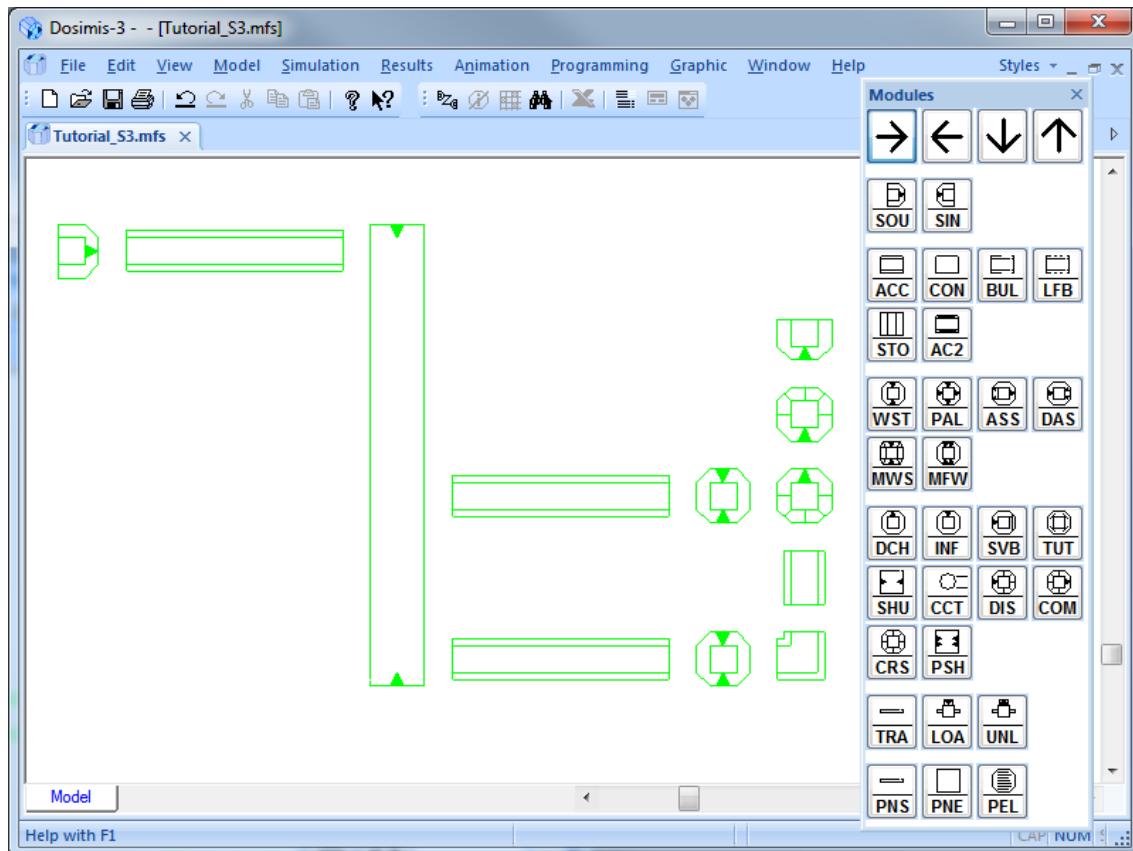


Figure 3.3: Placing of modules with orientation up



The last module to be placed (feed back) requires changing the orientation of material flow to left (). Then select the ACC symbol on the modules palette and place the accumulation conveyor between the distributor and the shuttle. The final Figure (Figure 3.4) displays all necessary modules on your work area.

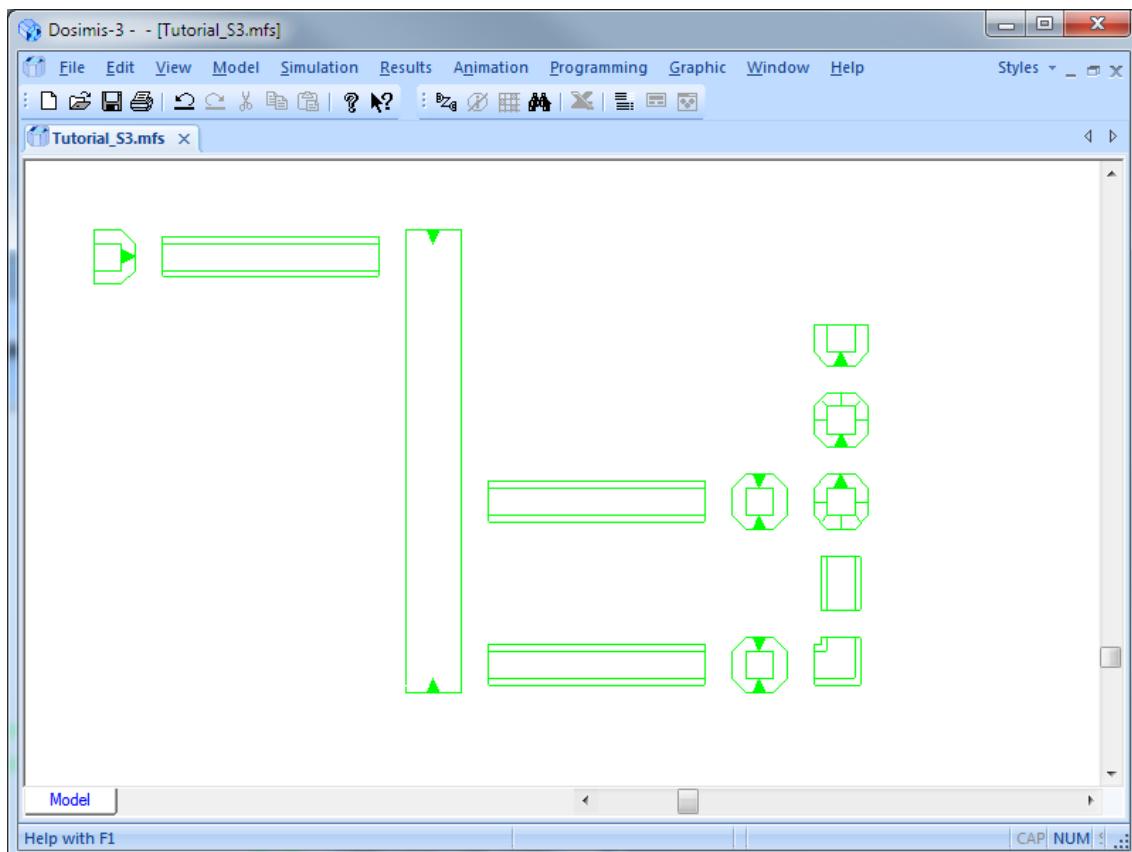


Figure 3.4: Layout containing all modules

Now you should save your model. Therefore select **File/Save as** from the menu bar of your DOSIMIS-3 window. A new window appears, where you can enter the filename for your model. Default is “Dosimis-3-1.mfs”. Change the filename to “Tutorial_S5.mfs” and click on the “Save” button.

63.1.1 Linking of Modules

Our next step will be to connect or to link the modules. Please select **Model/Linking active** from the menu bar. Now linking mode is active (displayed by a node symbol at the bottom of the mouse cursor) and you can enter links between the modules representing material flow connections. Therefore please click with the left mouse button on the start module, then select the destination module by click on the left mouse button and fix the link by pressing the right mouse button.

The fixed link is displayed as a line with two arrowheads. The display color of the start module (source) should have turned to black. The black color indicates that all necessary data entries for the module are done. Please repeat the procedure above until all entrances and exits of all modules are linked. But pay attention to the sequence of connecting entrances and exits and assigning the entrance/exit number to an entrance/exit!



Important!

For modules with more than one entrance/exit the sequence of linking has an influence on parameter settings, because the sequence of connecting entrances/exits assigns the entrance/exit number to an entrance/exit! In our model this modules are: the shuttle, the combining station and the distributor.

Linking rule:

Entrance 1 is assigned to the first input link, entrance 2 is assigned to the second input link and so on. Same applies to the exits. In case of the shuttle that means: connect the receiving buffer to the shuttle first (entrance 1), connect the feed back conveyor afterwards (entrance 2). To link the exits, first connect the ACC above to the shuttle (exit 1) followed by the ACC below (exit 2).

Please link the distributor to the sink at first (exit 1) followed by the feed back conveyor (exit 2).

Now you have finished your model and the model window on your computer screen looks as follows:

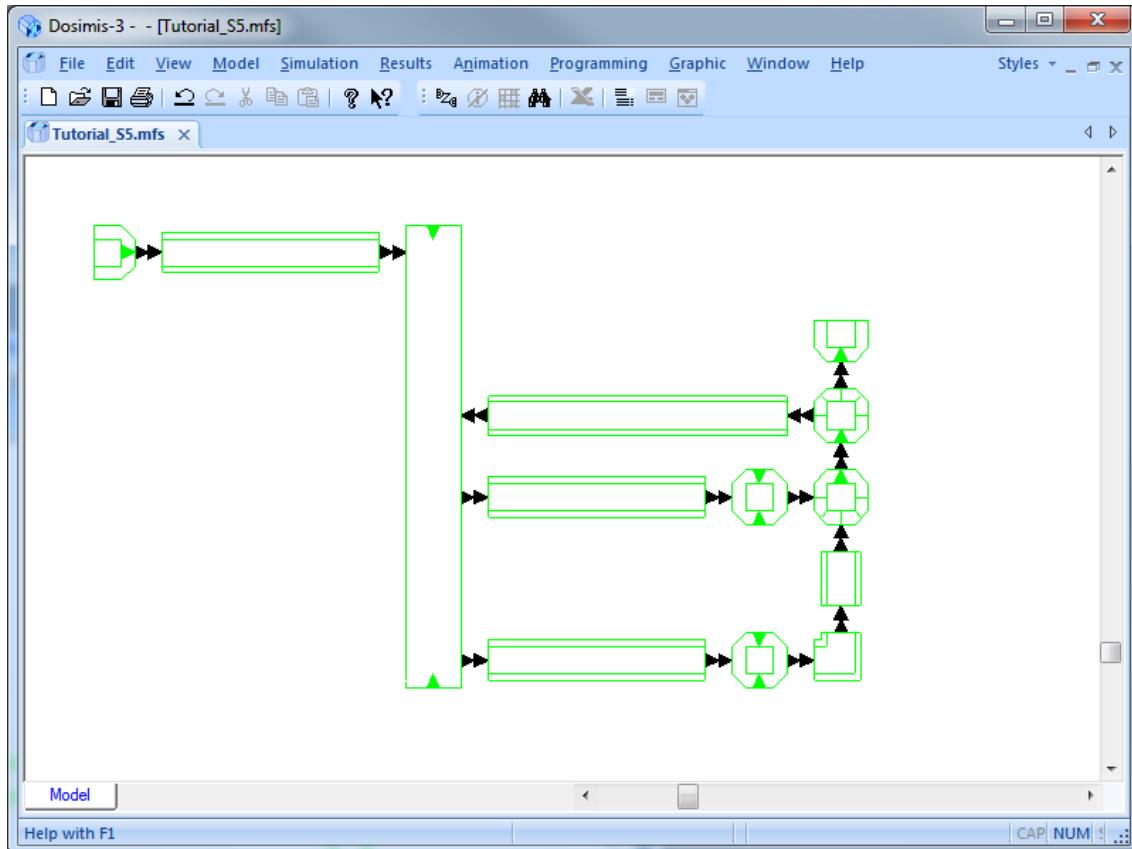


Figure 3.4a: Finished model layout

Linking mode has to be disabled now. Therefore please select **Model/Linking active** from the menu bar. As sign that the linking mode has been disabled, the node symbol below your mouse cursor will disappear.



63.2 Input of Data with the Aid of Parameter Masks

63.2.1 Parameter Source

In the next steps, module data will be entered. Double-click the source to open the following window (parameter-input mask):

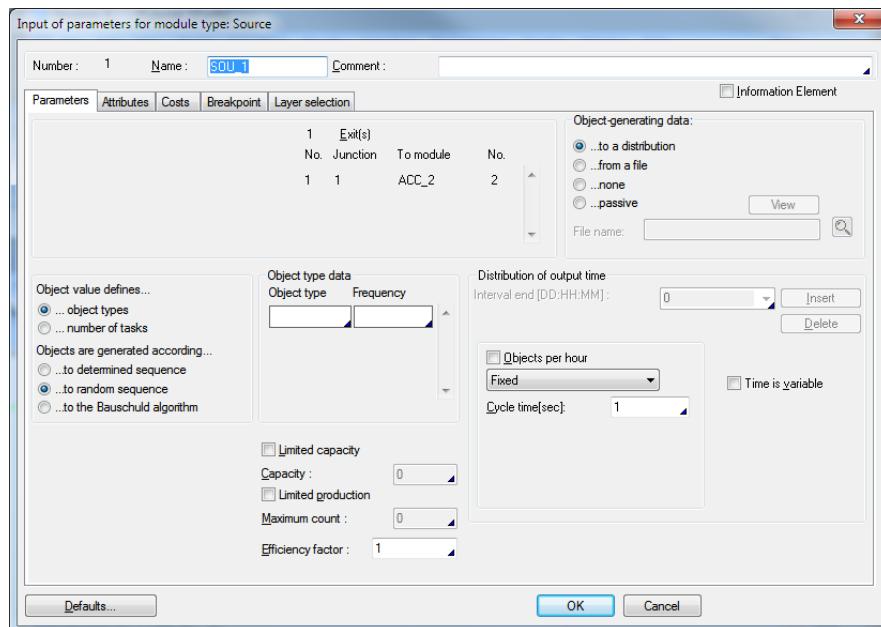


Figure 3.5: Parameter-input mask of the source

Every marked input box (circles: radio buttons and white boxes: input arrays) normally has to be edited. You can use the “Tab” button on your keyboard to jump forward to the next input box. Use “Shift” and “Tab” button to jump backwards. Or select the input box directly by clicking it with the left mouse button.

The Module Number is assigned automatically, Name and Comment can be skipped.

a) **Object Generation Data**

Objects shall be generated in random sequence- on average a defined distribution will result, however, only after a sufficiently large number of samples. So, please select the radio button named “...according to a distribution”.

b) **Object Type Data (object type and frequency)**

The relative frequency of objects (here product groups) to appear, is entered according to the following table. Please use the “Tab” button on your keyboard to jump to the next input array and to add a new line of input arrays.

object type	frequency
1	50
2	50

This means, that both types will occur within a probability of 50 %.



c) Distribution of Output Time

By a click on the button at the right side of the combo box a drop-down list opens which contains all available types of distributions. Please select “normally distributed” from the combo box. Now enter 60 seconds as “mean” (expected value) and 5 seconds as deviation. All other settings remain given by default. In the end the parameter input mask should look as follows:

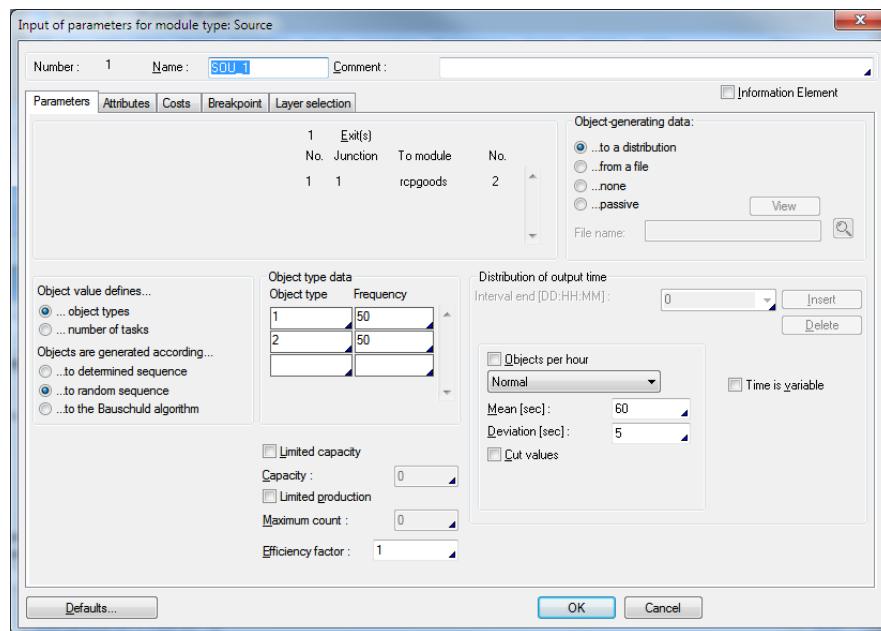


Figure 3.6: Completed parameter input mask of the source

To leave a parameter input mask and to confirm the entries made, please hit the “Enter” key on your keyboard or click on “OK” button of the mask.



63.2.2 Parameter Accumulation Conveyor

The parameters of the accumulation conveyor are entered in the same way. To open the parameter input mask, double-click on the module. Our model contains six accumulation conveyors.

Parameters are partly different. All accumulation conveyors of interest receive a special module name. These module names will help you to understand and to interpret the results.

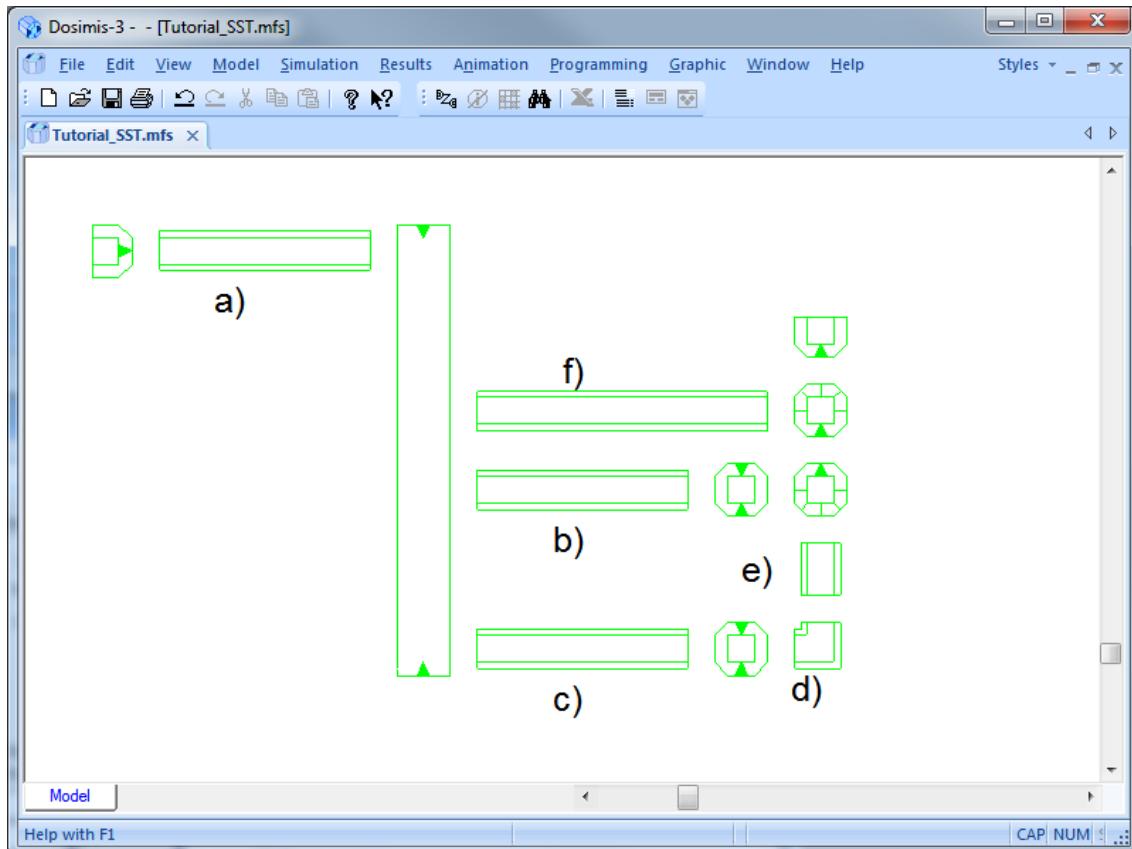


Figure 3.7: Assignment of the accumulation conveyors

To a): Buffer at receipt of goods

Only the first four lines of the parameter input mask are of interest for this model. These lines contain the conveying system data. The number of the module is generated and cannot be changed. Please enter the values of the following input areas as shown below:

Name:	rcpgoods (default: ACC_2)
length of segment (length of a component or pallet):	1 m
conveying speed:	0.2 m/sec
capacity (number of segments):	10 parts



The following figure results from these entries:

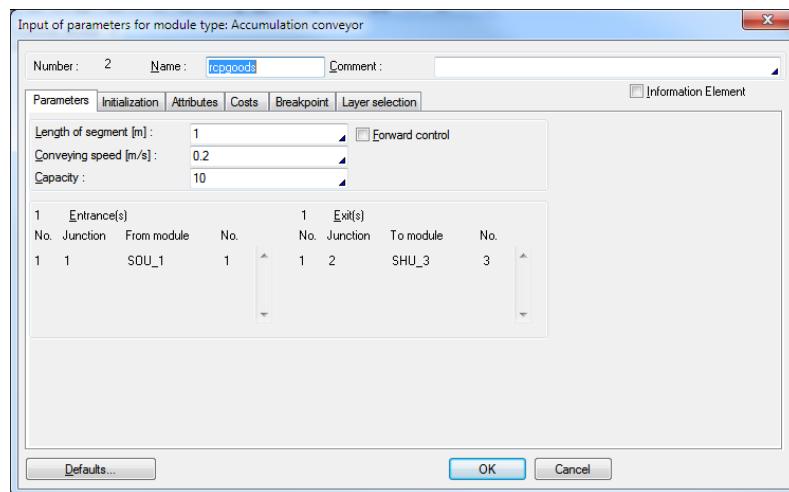


Figure 3.8: Parameter-input mask of the ACC after the source

To b) and c): Buffer before work station

Please use the same parameters as in a) except for capacity and module name. The capacity (number of segments) of the ACC is 2.

Module names: ACC_Above, ACC_Below

To d): This is an edge conveyor

Same data as in a) except capacity of 1 and the forward control has to be activated by clicking on its check box. A new pallet is only allowed to enter the conveyor after its preceding pallet has left the edge conveyor.

To e) and f): Connecting buffer and feed back conveyor

Same data as in a) except capacity of 4 in ACC e) and 3 in ACC f). Please remember to save your model occasionally.

63.2.3 Parameter Shuttle

Enter of shuttle data.

The following data sets, gathered in a box with light gray borders, have to be entered for the shuttle. From Top to bottom the shuttle parameter-input mask contains:

- Conveying data
- Entrance and exit assignment
- Dimensioning (on right side of entrance/exit assignment)
- Right-of-way strategy
- Distribution strategy

Conveying data:

- loading path for one palette: 1.1 m (10 cm longer than palette length)



- loading speed for one palette: 0.2 m/sec
- unloading path per palette: 0.1 m
- unloading speed per palette: 0.2 m/sec
- calculation of travel time: slow/fast
- slowly driven path for the car: 0.5 m (acceleration/deceleration distance)
- speed fast: 1 m/sec (maximum speed of the car)
- speed slow: 0.1 m/sec (average positioning speed of the car).

The remaining conveying data will not be changed.

Dimensioning:

Please use the values of following table:

Entrance	pos. [m]	height	Exit	pos. [m]	height
1	0.0	0.0	1	20.0	0.0
2	15.0	0.0	2	25.0	0.0

This dimensioning data result from the creation of a coordinate system at the shuttle path. Its origin is close to the receipt of goods (Entrance 1), the end is close to the “ACC_Below” (Exit 2).

The height is used to reproduce a lifting movement and is not needed here.

All other settings remain as given by default. The completely filled up parameter-input mask for the shuttle should like shown below:

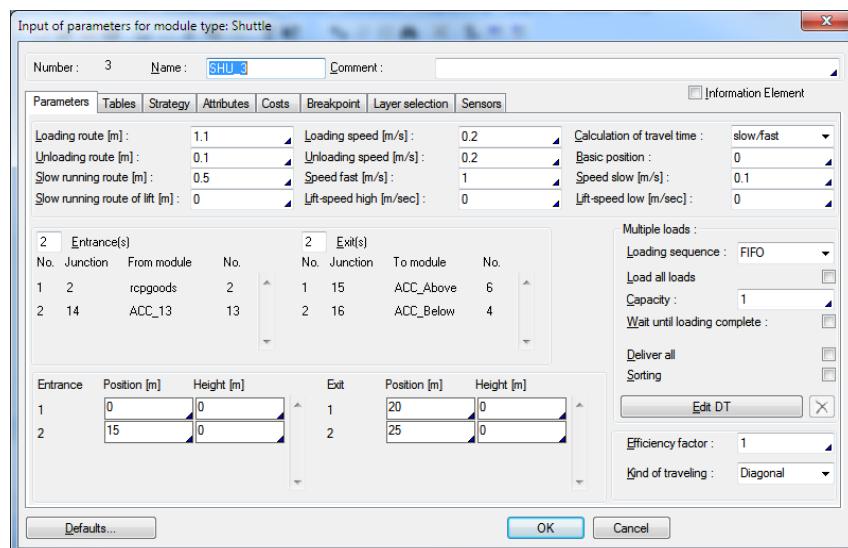


Figure 3.9: Parameter-input mask of the shuttle

Entrance and Exit assignment:

Entrance(s): 2 (receipt of goods and feed back)
Exit(s): 2 (to both work stations)



Right-of-way strategy:

The right-of-way strategy makes the decision which palette will be carried as next one. Select “priority of entrances” from the combo box.

Priority of Entrances:

Entrance	Priority
1	1
2	2

Assigning of entrance numbers will be done later when connecting the modules. Following assignment will be done:

- Entrance 1: receipt of goods buffer
- Entrance 2: feed back conveyor

Priority 1 is highest, therefore priority 2 is of lower priority.

Distribution strategy:

The distribution strategy defines the exit, to which a pallet will be moved. Please select “destination with” from the combo box, enter object type 1 at first, then use the “Tabulator” button on your keyboard to jump into the input area of the second object type and enter 10, now click on the “next exit” button and act in the same way for exit 2 using the values of following table:

Destination with	
exit	object type
1	1
	10
	(next exit)
2	2
	20

The object type is one type of product (group). Product type 10 is assigned to rework of product type 1, 20 is assigned to rework of product type 2.

The completely filled-up strategy dialog for the shuttle should finally look like shown below:

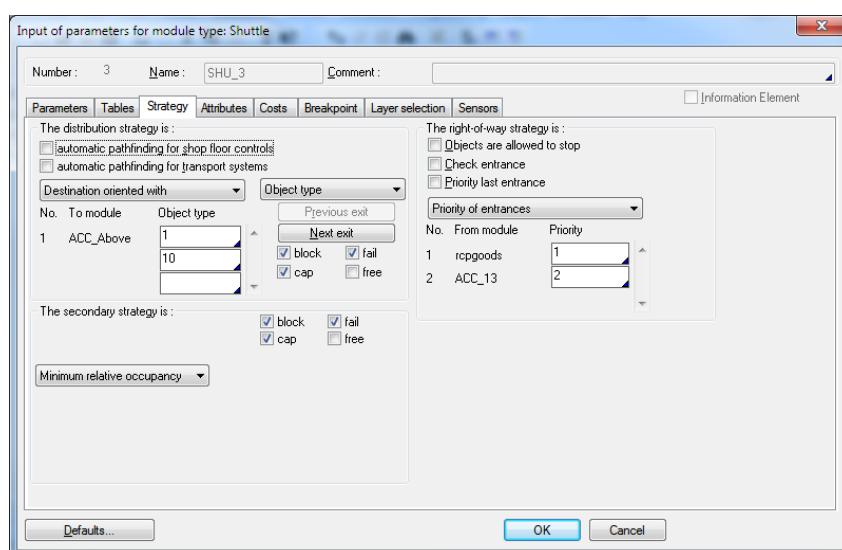


Figure 3.9a: Strategy input mask of the shuttle



63.2.4 Parameter Work Station

When entering the parameters of the work station (WST), you have to edit the following data sets:

- Conveying data
- Entrance and exit assignment (no data input!)
- Transport parameter
- Work procedure
 - Distribution of working time
 - Initial object
 - Set-up times

“Transport parameter” data sets are of no importance, because they are only required for transport systems. In this tutorial we use “Above” as the module name for the upper work station and “Below” for the lower one.

Conveying data:

The length of work station is equivalent to the entry length of one palette.

Length: 1 m
 Conveying speed: 0.2 m/sec

Distribution of working time:

Select “normally distributed” from the combo box. This sets “normally distributed” as default for the following distribution settings. Now first enter object 1, then use the “Tabulator” button on your keyboard to jump into the input area for distribution. As you can see the preselected item (normally distributed) is highlighted. So please use the “Tabulator” button to jump into the next input area called mean and enter 80. Hit the “Tabulator” button again and enter 5 as the deviation. Now press the “Tabulator” button to jump into strategy and leave the setting on “none” as it is. To open a new line, please hit the “Tabulator” button again and repeat the inputs for object type 10 in the same manner according to the following table. All time values are in seconds.

Important: To adopt the settings of a line, you have to enter the input area “strategy”!

normally distributed			
Work Station	objects	mean	deviation
Above	1	80	5
	10	80	5
Below	2	80	5
	20	80	5

Initial Object:

You will find all objects defined previously in the initial object combo box. The initial object combo box contains the numbers of objects that enter the work station. As you remember two types of objects enter each work station (“OK objects” and “rework objects”). Now we must define for each kind of object the probability of leaving the work station as an “OK object” or as a “rework object”. Object numbers 1 and 2 are assigned to “OK objects”, object numbers



10 and 20 are assigned to “rework objects”. Please enter the number of the “new object” leaving the station with its “probability” for each object selected from the “initial object” combo box according to following four tables.

For work station “**Above**”:

initial object 1 (entering)		initial object 10 (entering)	
new object (leaving)	probability	new object (leaving)	probability
1	85	1	85
10	15	10	15

For work station “**Below**”:

initial object 2 (entering)		initial object 20 (entering)	
new object (leaving)	probability	new object (leaving)	probability
2	85	2	85
20	15	20	15

Set-up Times:

The set-up time is executed only if there is a change of object type. Please enter the values from following table in the same manner as described in “Distribution of working time” (see above) using the “**fixed**” distribution here:

Work Station	from object	to object	cycle time
Above	1	10	60
	10	1	60
Below	2	20	60
	20	2	60



Same data are used for both work stations. It is the shuttle that controls which product is carried to which work station. By this time the work station has received the necessary information to proceed. Now the parameter input mask of one work station should look like shown in figure 3.10:

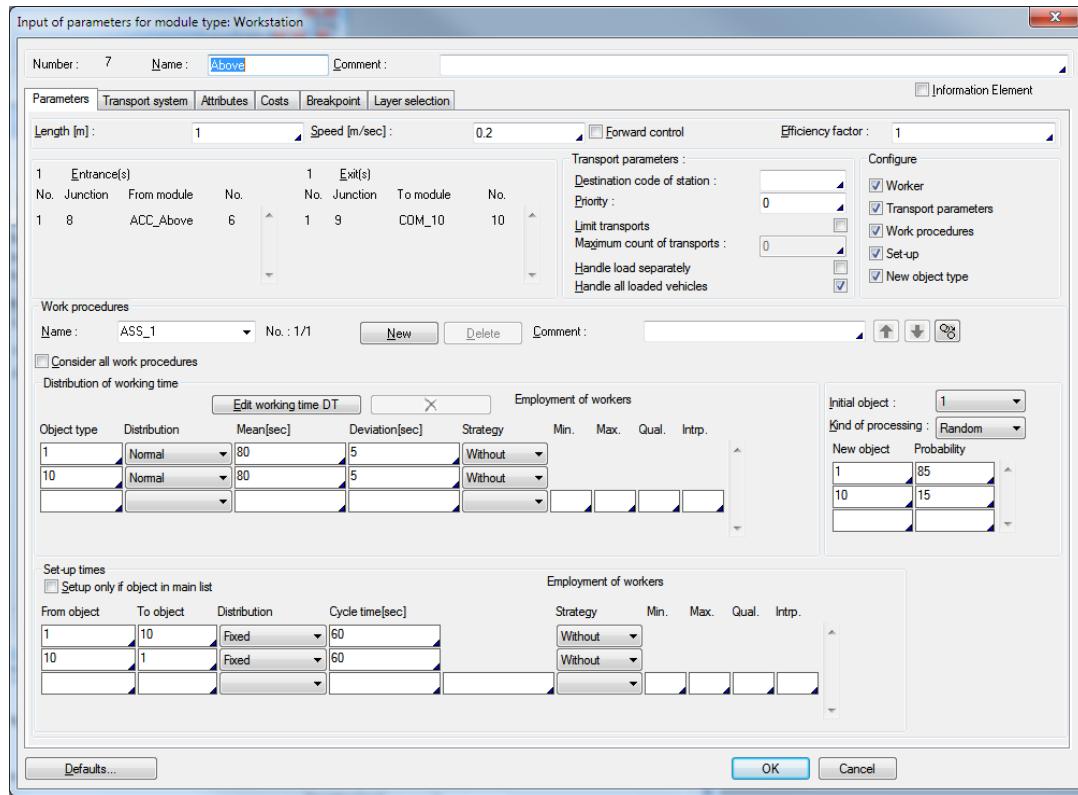


Figure 3.10: Parameter input mask of one work station



63.2.5 Parameter Combining Station

There is not much data left to be entered in the remaining modules. In summary, the following data sets are to be entered:

Combining station:

- *Conveying data*
 - conveying path: 1 m
 - speed: 0.2m/sec
- *Entrance- and. exit assignment*
 - Entrance(s): 2
- *Right-of-way strategy*
 - FIFO

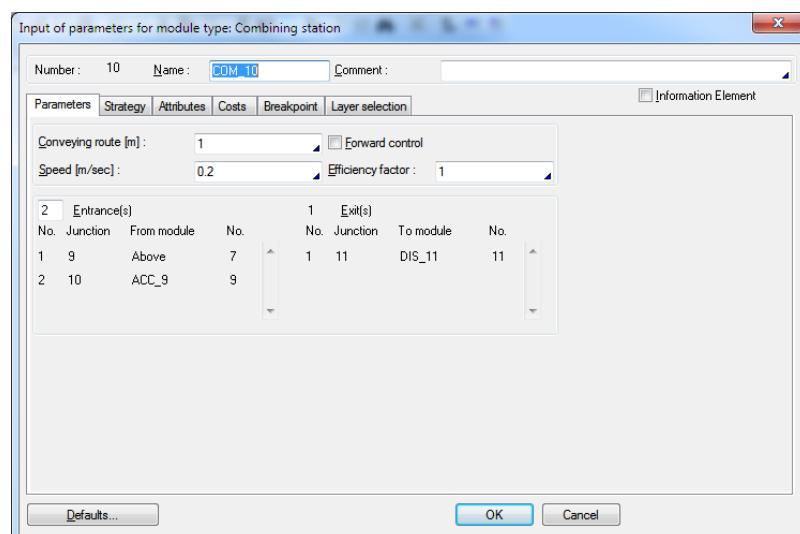


Figure 3.11: Parameter input mask of the combining station

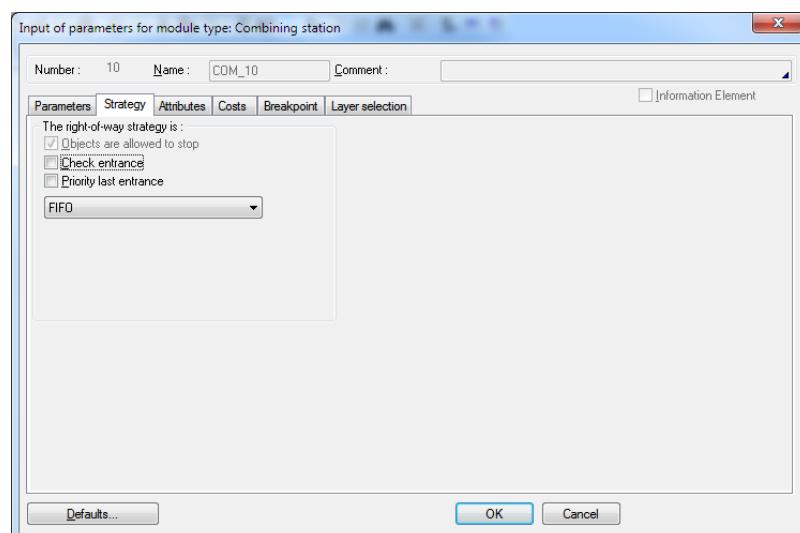


Figure 3.11a: Strategy input mask of the combining station



63.2.6 Parameter Distributor

Distributor:

- *Conveying data*
 - conveying path: 1 m
 - speed: 0.2m/sec
- *Entrance- and. exit assignment*
 - Exit(s): 2
- *Right-of-way strategy*
 - destination with (combo box)

Exit	Object type
1	1
	2
(next exit)	
2	10
	20

By using the distribution strategy “destination with (object type)” object types are assigned to one exit. The secondary strategy has to be edited if one object type can use more than one exit, because then the assignment is not unique.

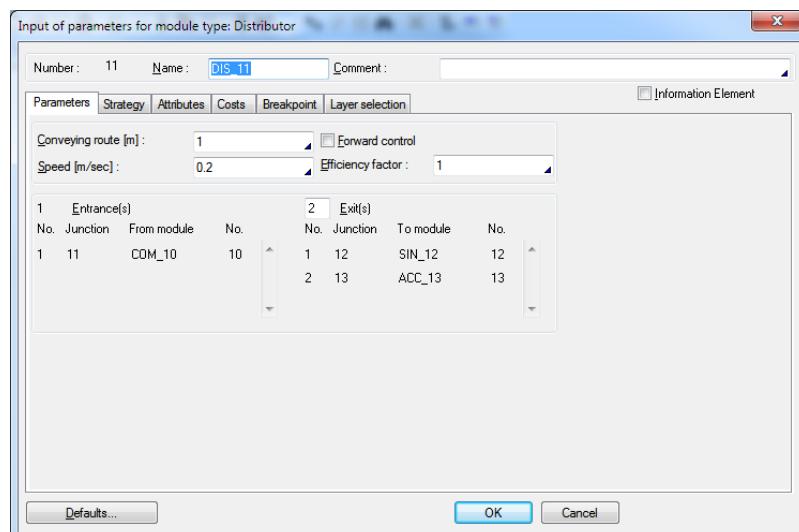


Figure 3.12: Parameter input mask of the distributor

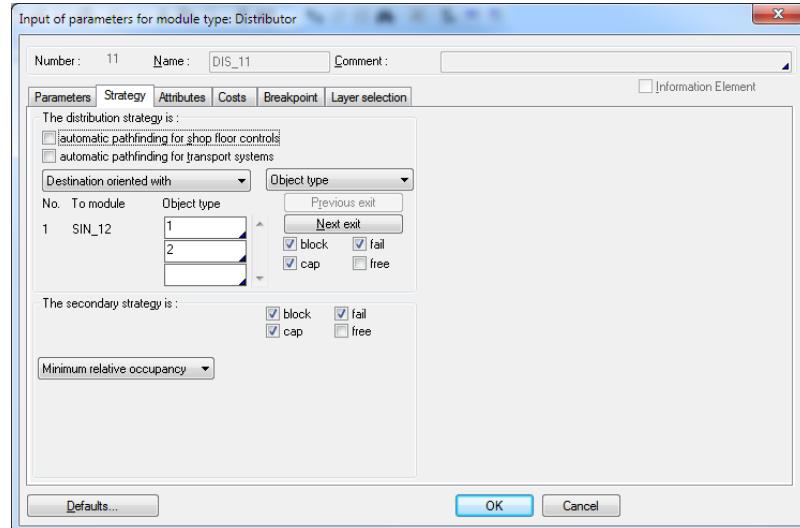


Figure 3.12a: Strategy input mask of the distributor

63.2.7 Parameter Sink

Sink: Please select “expo. distributed” from the combo box as distribution of leaving time and enter 55 sec. as mean.

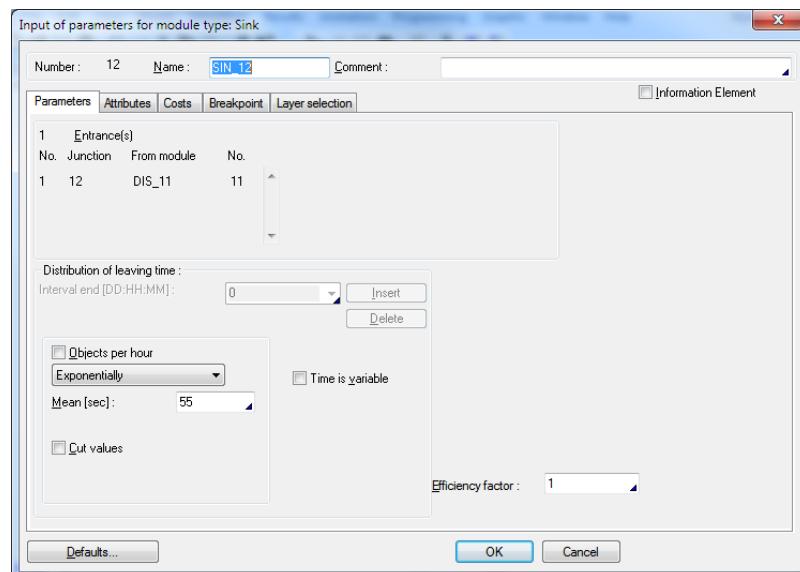


Figure 3.13: Parameter input mask of the sink

That completes the entire input of module parameters, so please save your model.



Now you have finished your model and the model window on your computer screen should look as follows:

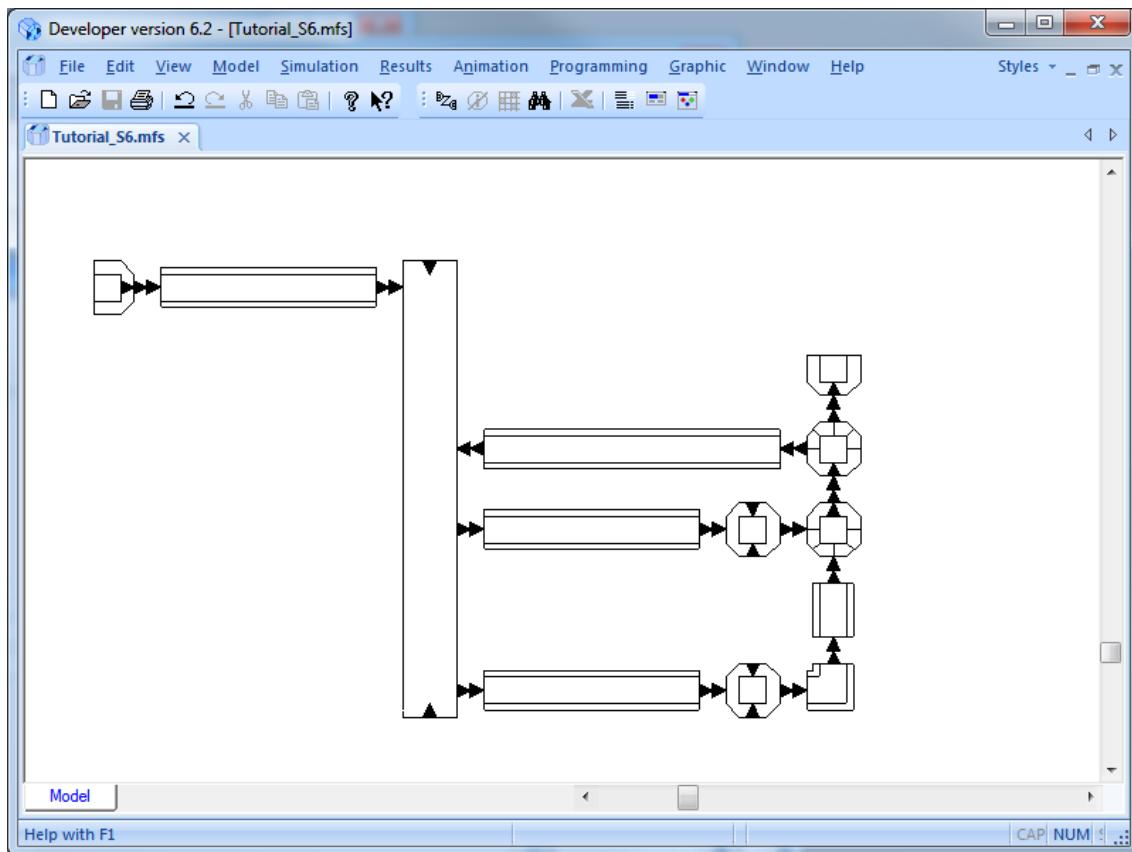


Figure 3.14: Finished model layout

63.3 Starting a Simulation Run

Before starting a simulation run you have to enter the simulation parameter. Please select **Simulation/Parameter** from the menu bar. The simulation parameter window appears. You can enter all time values in minutes. Please enter the following values:

simulation time	300	(results to 5 hours)
pre-run	0	(no pre-run for statistics)
statistic interval.	60	(every 60 minutes statistical data are collected)

All other settings remain as set-up by default. Before starting the first simulation run, please select **Simulation/Consistency check** from the menu bar. This activates some check routines that execute a restricted integrity check (parameter input data and links). This menu item is disabled if a successful consistency check has been done before.

To start the simulation, please select **Simulation/Start** from the menu bar. The DOSIMIS-3window disappears and a window appears containing a blue progress bar. This window will disappear as soon as the simulation run is finished and the DOSIMIS-3window will be opened again. The disappearing of the DOSIMIS-3window during the simulation can be averted by



deselecting the check box “hiding” in the simulation parameter mask. Now the collected results can be analyzed.

63.4 Results

In the next step we will look at the animation.

Here the time flow of object movement is visible. For this purpose please select **Animation/Parameter** from the menu bar. The animation parameter window appears.

Please select the radio button “*time factor*”. The display speed is set by the *time factor*. Please set the value of this time factor to 30 (default is 60). Therefore 30 minutes of simulation time will be displayed during 1 minute real time. (Please note that accurate work of time factor can be affected by performance and utilization of your computer.) Then select **Animation/Start** from the menu bar - animation is executed. Boxes in different colors and numbers inside are displayed inside the modules. The displayed colors have the following meaning:

- green: status „waiting“ or „moving“
- red: status „blocked“
- blue: status „working“

Result of the first run is a deadlock - no more movements - after a short time. (Figure 3.15). This problem can be removed by changing parameters (see chapter 4). For this purpose the animation has to be stopped at first. Please select **Animation/Stop** from the menu bar.

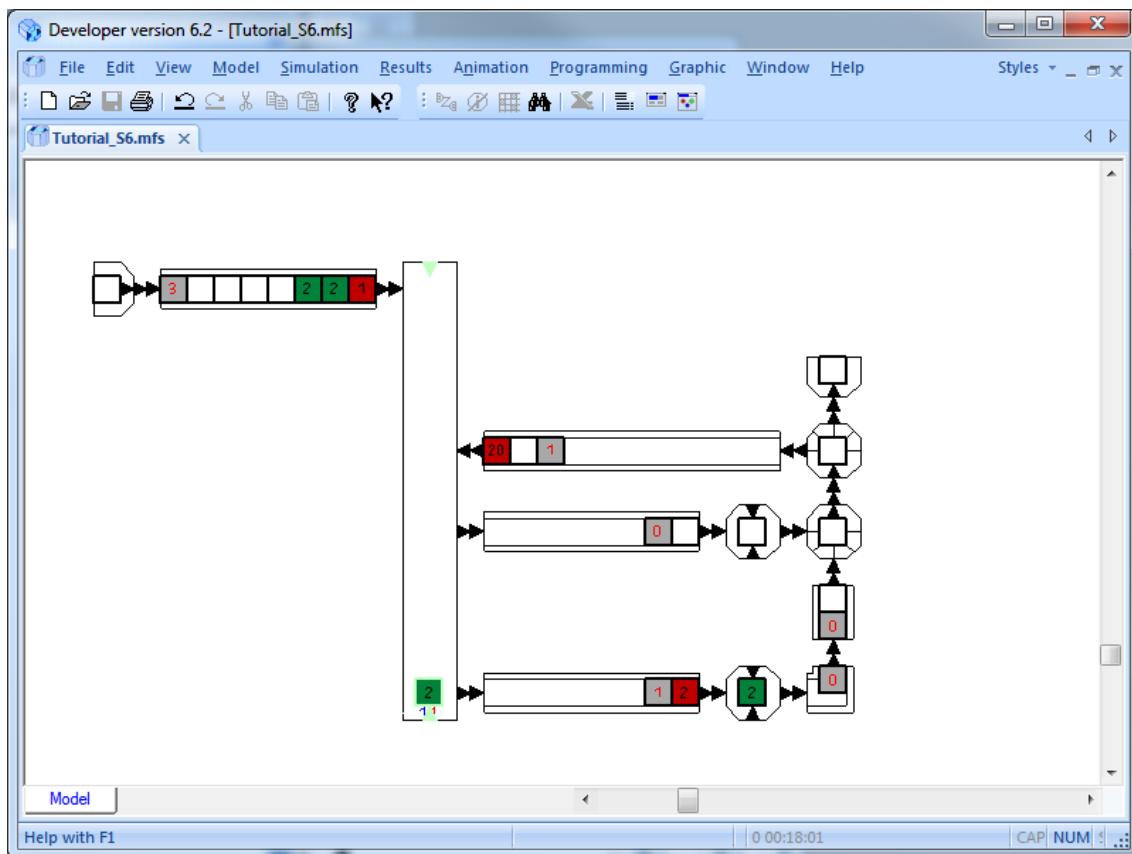


Figure 3.15: Process animation in the production system

Next we will attend to statistics.



A deadlock occurred during the first simulation run. Now the occupation of the source and of the buffer at receipt of goods is of interest. At first the modules of note have to be selected. Please click the source with the left mouse button, then press the “Ctrl” button on your keyboard and click the accumulation conveyor next to the source with the left mouse button. After this, please select **Results/Buffer analysis/Occupation diagram** from the menu bar.

The following picture will be displayed as a new window inside the DOSIMIS-3 main window.

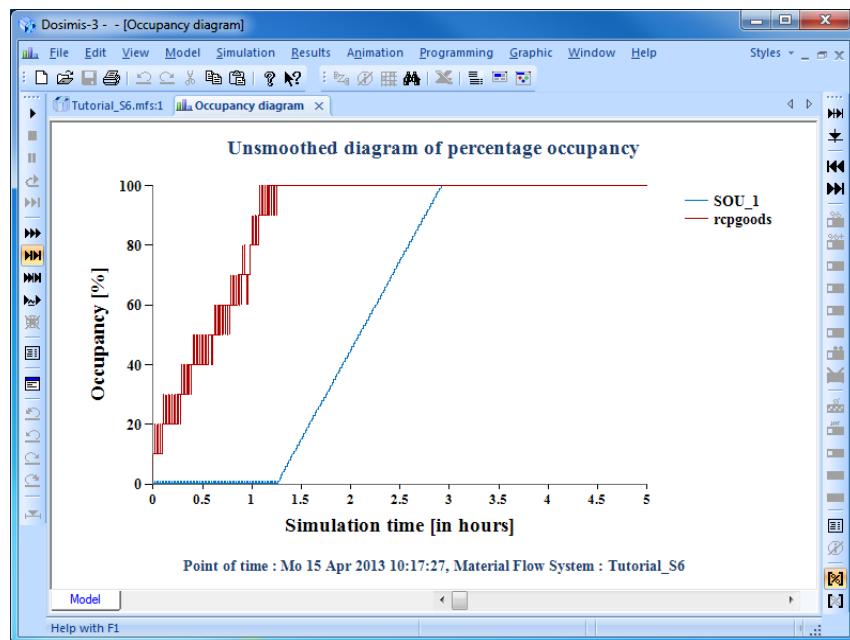


Figure 3.16: Occupation diagram

The capacity (in percent) of the buffer at the source and the source are shown over time. There are 10 % steps within the occupation of the incoming goods buffer for one palette inside the buffer.

Depending on random order a few deviations may occur in the pictures. To display the utilization of the work stations you must perform similar steps. Click the first work station then click the second work station while pressing the “Ctrl” button to select both work stations and select **Results/Module histogram** from the menu bar.



The following diagram appears as a new window inside your DOSIMIS-3 main window:

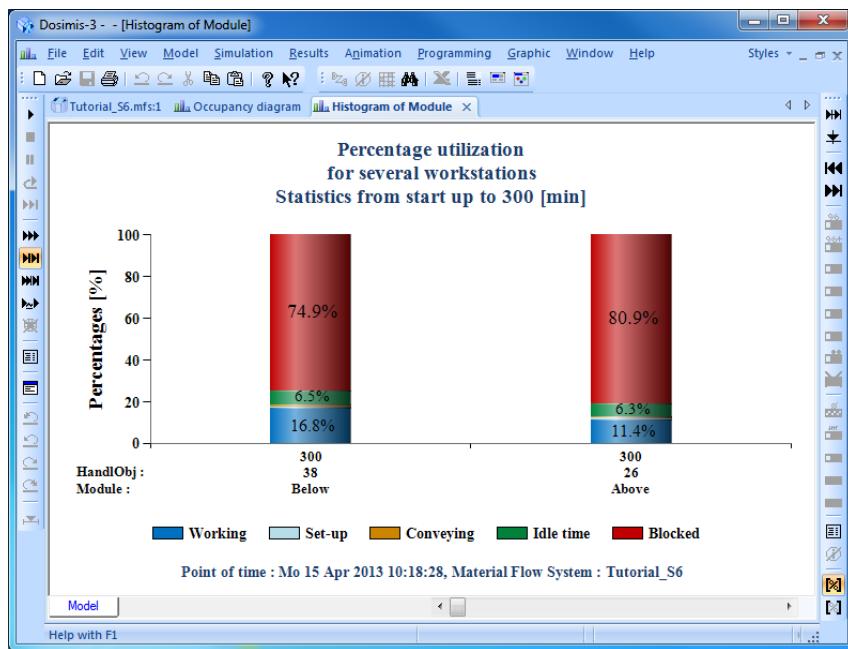


Figure 3.17: Module histogram

Utilization of work stations is displayed in the “*module histogram*”. To perform our task we will have to focus on following items: “working”, “set-up”, “idle time” and “conveying”.

63.5 Problems

When using a simulation system, some “surprising moments” may occur. These unexpected exceptional cases can be divided in two categories:

- program bugs
- handling errors

Program bugs:

- Program crash of simulation software
Precaution: Save your model as often as possible!
If a model was destroyed by a program crash you can retrieve the backup of the last saved model. The backup files are named **modelname.dbk** and **modelname.mbk** where “modelname” is the name you assigned to your model (default “modelname” is DOSIMIS-3-1). To back up the last saved model, please rename the files **modelname.dbk** to **modelname.dar** and **modelname.mbk** to **modelname.mfs**.

Handling errors:

There are some operating sequences that sometimes cause confusion. The following cases have occurred in practice:

- Wrong selection for statistic data
Module histograms can only be displayed for work stations and shuttles. If any other module was selected to display module histogram an error message is displayed in a new window. To quit this message click on the “x” symbol in the upper right corner of this message window (but do not click on the “x” symbol of the DOSIMIS-3 window!!!).
- Sometimes a user forgets to disable “linking mode” after connecting modules. The active “linking mode” is visible by a displayed node below the cursor and links are displayed in green color. A lot of usual operations are not possible in “linking mode”.





64 Experiments using the Practice Example

64.1 Initial Situation

The small production system was planned and shall be checked again by aid of simulation.

Open questions concern the dimensioning of buffer sizes and the strategies to control the equipment.

The layout, in particular the arrangement of incoming goods as well as the technical data of production (working times and quota of rework) are not changeable within the scope of planning. Starting point for further analyses is the completely entered model as described in chapter 3.

Following steps of optimization result from our first simulation run. (Important: result diagrams must not match this presentation exactly - depending on the sequence of inputs slight deviations may occur.)



64.2 Steps of Optimization

64.2.1 Step 1 - Deadlock

Start:

Animation parameter (starting at 0, time factor 30). A deadlock occurs after a short time - no more movements are visible.

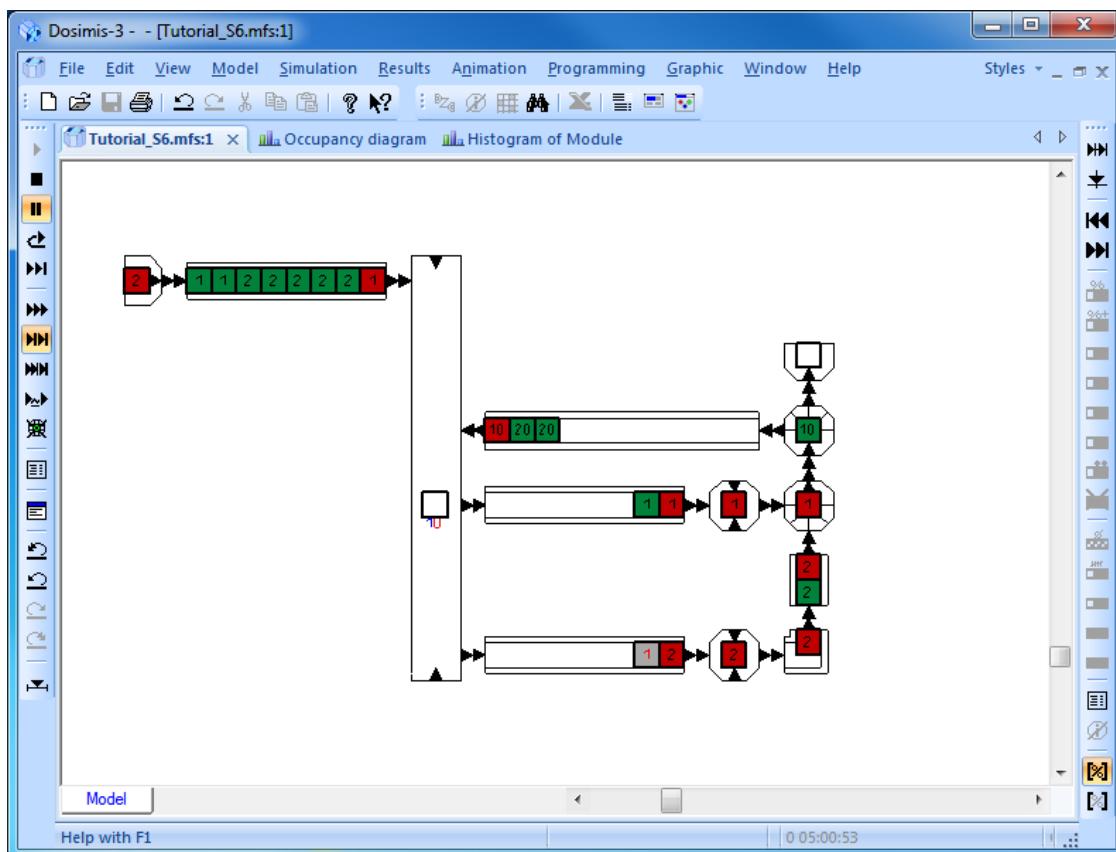


Figure 4.1: Deadlock

Cause:

The distributor contains an object for the destination feed back conveyor (rework), but the feedback conveyor is full. This object blocks the material flow and causes reverse blockages.

In the case of more precise consideration we can identify two material flow circuits. A small one, that is composed of the shuttle, the ACC before upper work station, the workstation itself, the combining station, the distributor and the feedback conveyor.

The big circle consists of the elements of the material flow containing the workstation below. The shuttle does not pick up any object in case of a jam. The control unit of the shuttle and the storage/retrieval machine does not allow picking up objects, if the object cannot be delivered. Even if this control statement did not exist and the shuttle would pick up and carry the next object to its destination, the object could not be unloaded.

A first proposal to solve the problem could be to increase the buffer size of the feedback conveyor. This leads, however, to deadlocks again and again - the appearance of the deadlock



is only retarded since critical situations occur less frequently - they are not avoided in this way.

Conclusion:

The actual cause of the deadlock is the wrong priority in shuttle control. The feedback conveyor has to dispose of the elements with higher priority.

64.2.2 Step 2 - Presorting

Measure:

Please open the parameter input mask of the shuttle and change the entrance priority.

Right-of-way strategy	
Entrance	Priority
1	2
2	1

Priority “1” is the highest; therefore priority “2” is of lower priority.

Please start a simulation again. Select **Simulation/Start** from the menu bar.

Start:

- a) Occupancy diagram of the source and the ACC after the source: The diagram shows that buffers become full very soon. *The source has a capacity of 100 objects (therefore one object is equivalent to one percent).* After 5 h about 80 - 90 objects have retailed in incoming goods (inside the source).

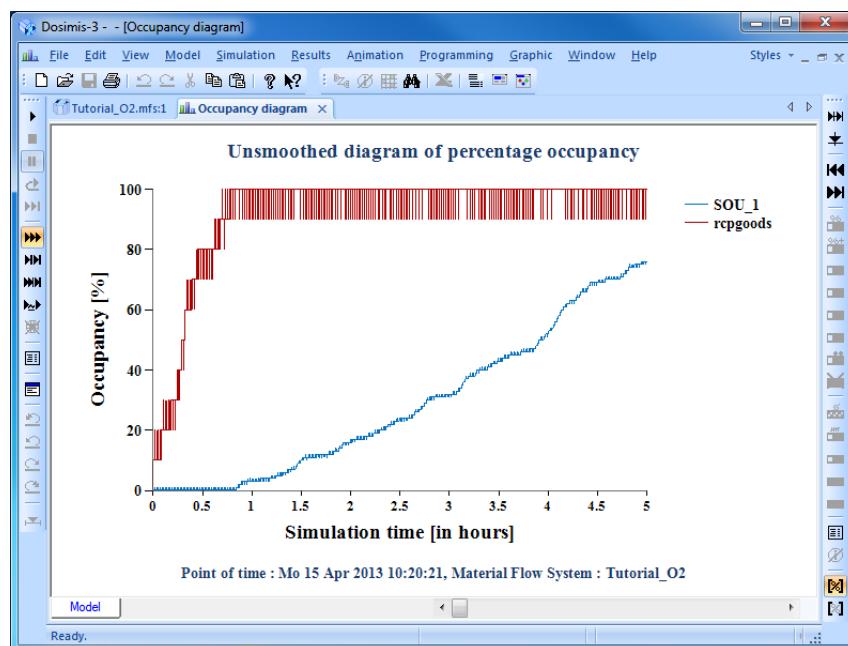


Figure 4.2: Occupancy diagram of the source and its following ACC



- b) Module histogram of both workstations: As you can see both workstations are idle (green) for about 20 % of time.

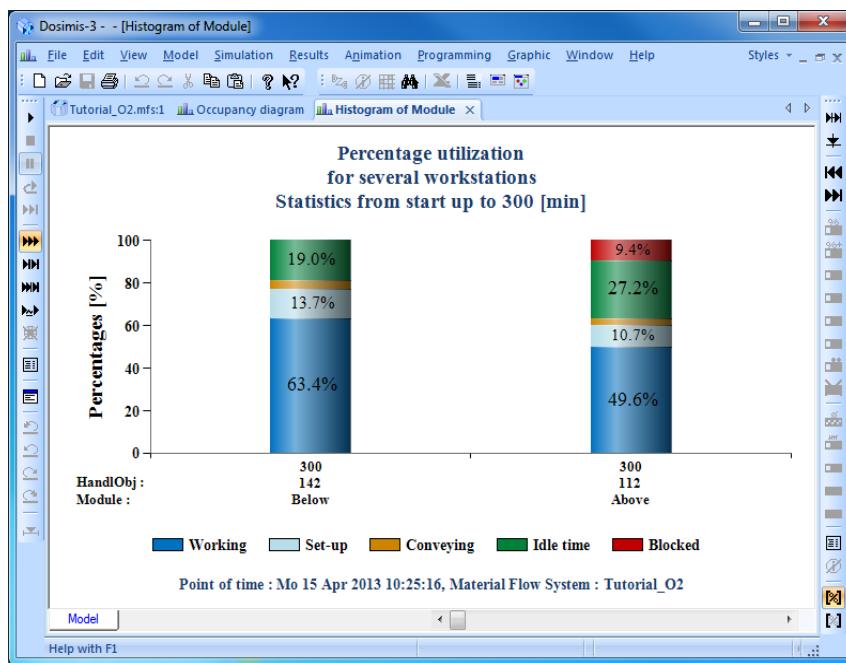


Figure 4.3: Module histogram of both work stations

- c) Module histogram of the shuttle: There is idle time too - even though the value is a little bit smaller.

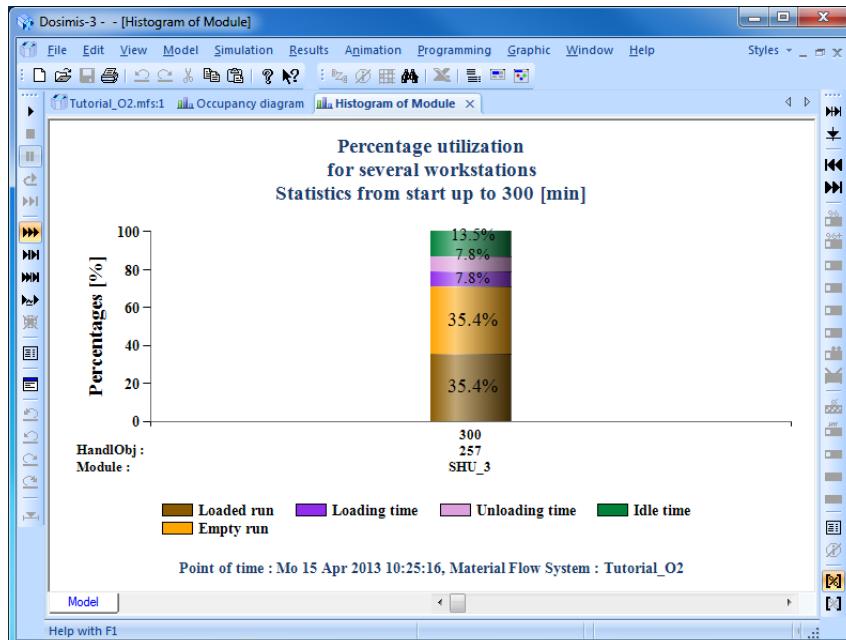


Figure 4.4: Module histogram of the shuttle

Conclusion:

Work stations and shuttle are waiting (idle) for some time but there is a jam of parts at the incoming goods (source).

**Cause:**

Products of type “1” and “2” appear in irregular sequence. They are balanced in the long run, but in smaller time intervals, the product mix can unfavourably swing to one or the other side.

Size of the buffer between shuttle and workstation is too small for these variations. One of the buffers before the workstations is filled up- if the same type of product now waits at the end of the buffer after the source, the buffer before the other workstation will become empty and at least that workstation will become idle

Different measures could basically be considered:

- a) Sorted delivery of goods at the source.

Components of type 1 and 2 will be delivered in batch size 1 or 2 (that means alternating).

Disadvantage:

The preliminary system has to sort the components - problems are shifted outside. In addition the random onset of rework will confuse every normal sorting.

- b) Flexible distribution of products 1 and 2 onto both work stations - according to volume of work.

Disadvantage:

Big set-up efforts to change from product 1 to 2 and vice versa as well as expensive investments for highly flexible machines

- c) Presorting of products from the source onto two side-by-side sorting conveyors at the receipt of goods.

Disadvantage:

Though there are tailbacks which affect alternating on both work stations, one sorting conveyor always will fill up and finally will block the supply of the neighboring conveyor.

- d) Increase size of the buffers between shuttle and work stations.



64.2.3 Step 3 - Decoupling (Increase of Buffer Size)

Measure:

Measure d) will be realized in this step. Please increase the buffer size of the accumulation conveyors between shuttle and work stations. A typical value for these capacities may be 4 or 5. This is the number of segments on one accumulation conveyor.

Then please save your model and start another simulation run.

Start:

Occupancy diagram of the source and its succeeding conveyor: As you can see the reverse blockage inside the source decreased only a little - the measure was ineffective.

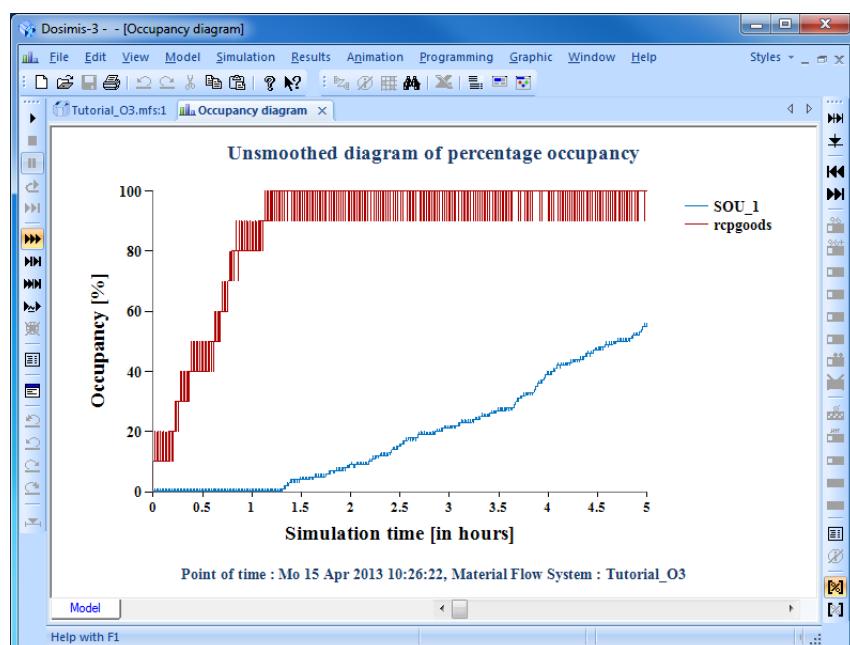


Figure 4.5: Occupancy diagram of the source and its succeeding conveyor

Cause:

No direct cause can be recognized, the causes of step 2 are still basically valid.

Conclusion:

Simulation technology affords the opportunity to analyze circumstances, initially independent of investment possibilities.

The actual measure in step 2 was to decouple the critical modules work station and shuttle. We wanted to analyze if problems could be solved by decoupling.

Measure:

Please increase the capacity of the accumulation conveyors in front of the work stations again and use the unrealistic value of 10 this time.

Then start another simulation run.



Start:

Occupancy diagram of the source and its succeeding conveyor: The reverse blockage decreased only a little - especially if you consider that buffer capacity of the whole system was increased significantly.

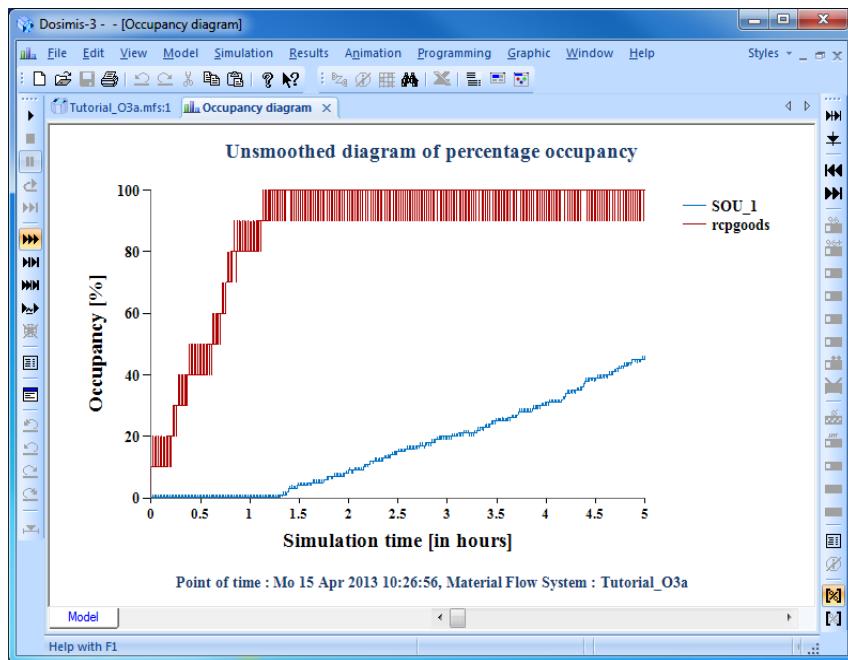


Figure 4.6: Occupancy diagram of the source and its succeeding conveyor

Occupancy diagram of ACCs before work stations: The buffer below is never filled up and the buffer above is well used.

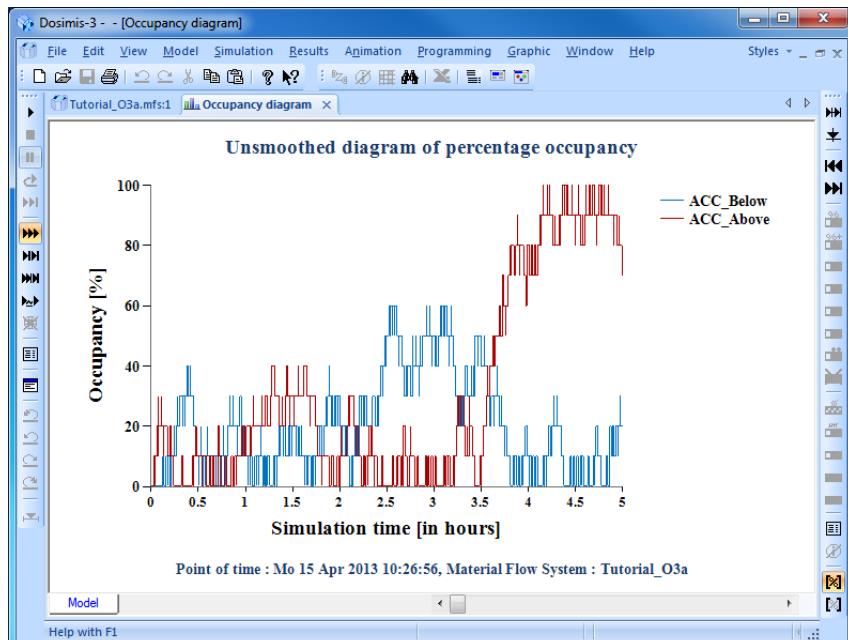


Figure 4.7: Occupancy diagram of ACCs before work stations



Conclusion:

The shuttle must be the bottleneck, because the buffer before the shuttle is filled up all the time and the buffers behind the shuttle are not filled up all the time. A look at the module histogram of the shuttle shows that the shuttle has nearly no more idle time.

64.2.4 Step 4 - Increase Shuttle Speed

Measure:

The shuttle has to become faster. A technically and economically reasonable value amounts to 2 m/s maximum speed. Please enter 2 m/s as the new value of “speed fast” in the parameter mask of the shuttle.

Afterwards, please start another simulation run.

Start:

Occupancy diagram of the source and its succeeding conveyor. Reverse blockage is now clearly smaller - however still present.

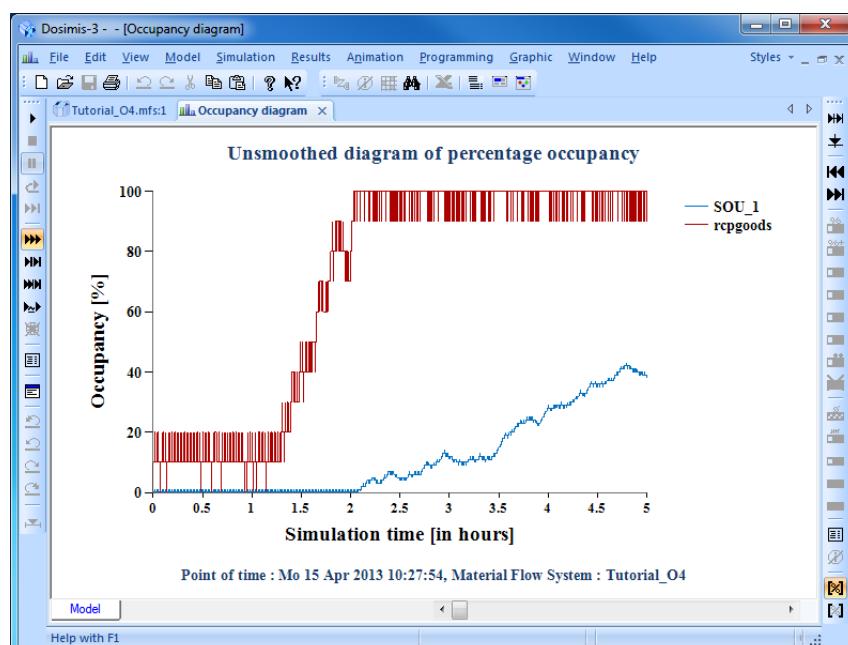


Figure 4.8: Occupancy diagram of the source and its succeeding conveyor



Module histogram of both work stations: The upper work station proves a relatively high blockade portion (about 20 %, displayed in red color).

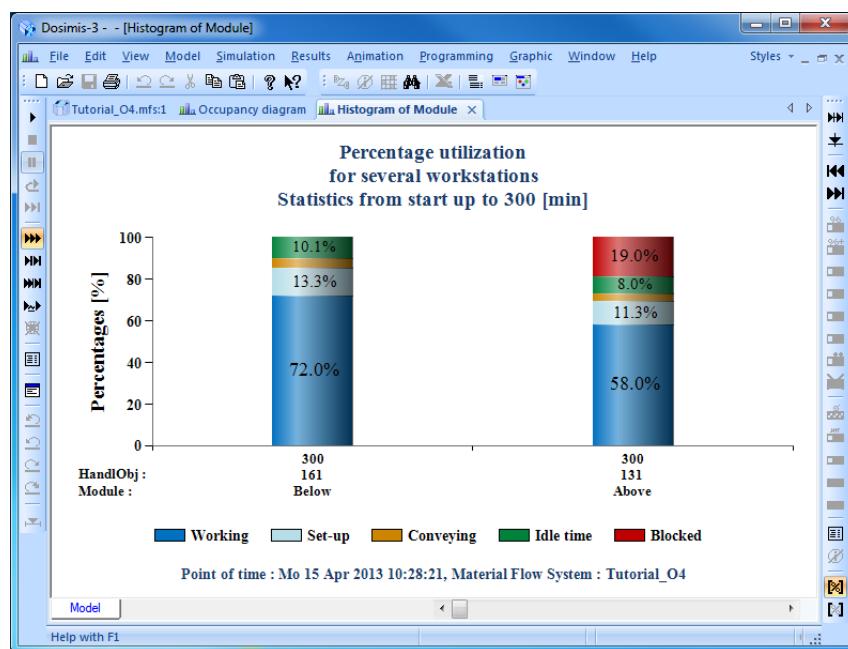


Figure 4.9: Module histogram of both work stations

Cause:

Due to varying process times of the sink, sometimes longer queues form in front of the sink. These queues repeatedly block the accumulation conveyor before workstation “Above” - workstation “Below” is not affected in the same way because there is a buffer of 5 places behind that work station.

64.2.5 Step 5 – Preferred Disposal

Measure:

Please change the “right-of-way” strategy of the combining station: The entrance connected directly to work station “Above” shall be favored - please assign priority “1” to this entrance and priority “2” to the second entrance (connected to the accumulation conveyor).

Then start a new simulation run.

Result:

A look at the occupancy diagram of the source and its succeeding accumulation conveyor shows that the last measure was not efficient.

Cause:

The jam at the combining station results from the dynamic behavior of the sink. Changing the priority of right-of-way changes nearly nothing ; the jam at the combining station remains. Workstation “Below” is not affected in the same way because its distance to the sink is larger compared to work station “Above”. Due to a jam at workstation “Above”, the accumulation conveyor before this workstation will be filled up - this ACC is no longer able to receive objects of type “1”. A filled-up buffer is no buffer anymore - it lost its operativeness.



64.2.6 Step 6 - Buffer before Sink

Measure:

Please insert an accumulation conveyor between distributor and sink. To decouple the sink buffer capacity should be set to 10 (number of segments). The best way to do this is to copy an existing module (e.g. ACC before work station “Above”). Therefore click the module “ACC_Above” and select **Edit/Copy** and then **Edit/Paste** from the menu bar. A new module appears above “ACC_Above”. Right click this module and select **Element/Replace element** from the context menu. Now first click the distributor then the sink and press the right mouse button to complete this action. If the new module is not at the right place, then click this module and keep the left mouse button pressed while moving it to its destination. To fix the module release the mouse button. As the next step, switch to linking mode and connect the module to distributor and sink. Finally deselect the linking mode and save your model.

Now your model should look like shown below.

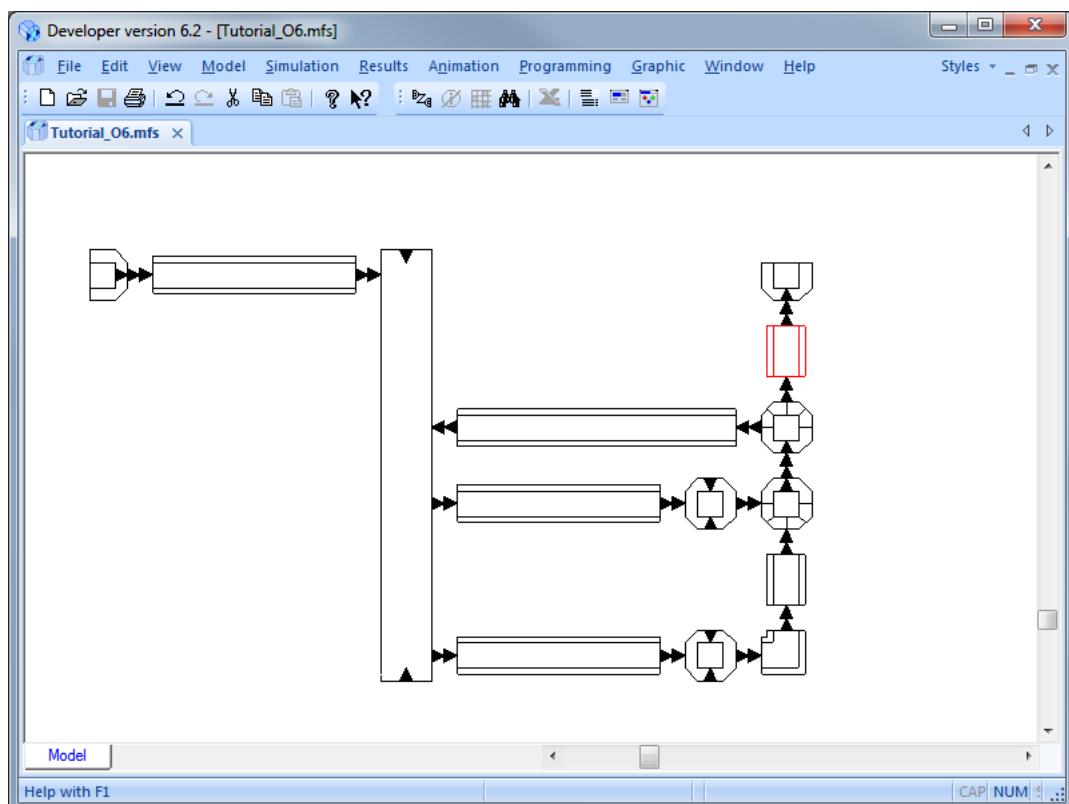


Figure 4.10: Layout with buffer before sink

Please start a new simulation run again.



Start:

Please look at the occupancy diagram of the source and its succeeding accumulation conveyor (see Figure 4.11):

The situation has partly improved - however, there is a reverse blockage on the accumulation conveyor. The stability of this system appears not to be safe, it could only be proved by a longer simulation time. Please enter 10 h as new simulation time in the simulation parameter mask and start a new simulation run.

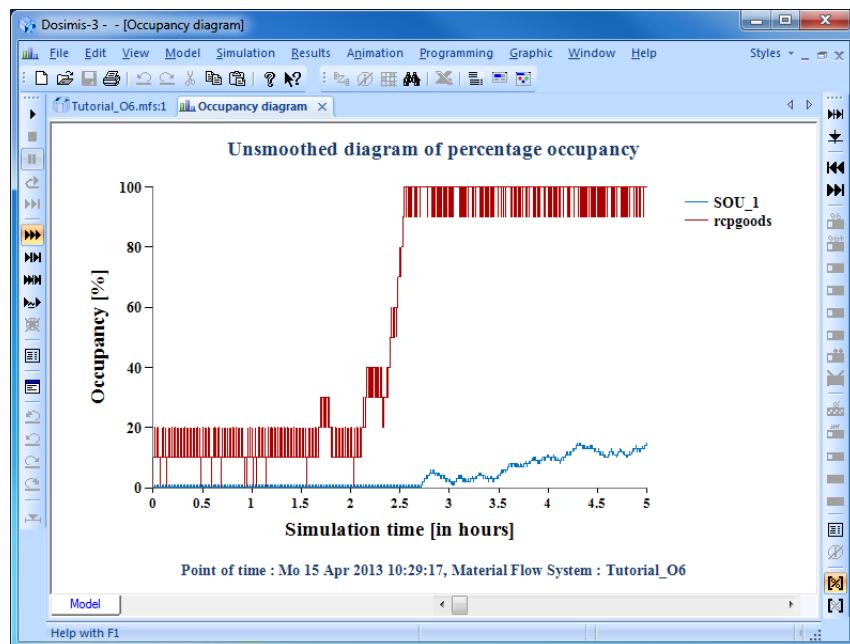


Figure 4.11: Occupancy diagram of the source and its succeeding conveyor (5h)

Regarding the occupancy diagrams of the source and its succeeding conveyor an unpleasant surprise occurs - the system is not steady and gets filled up (Figure 4.12).

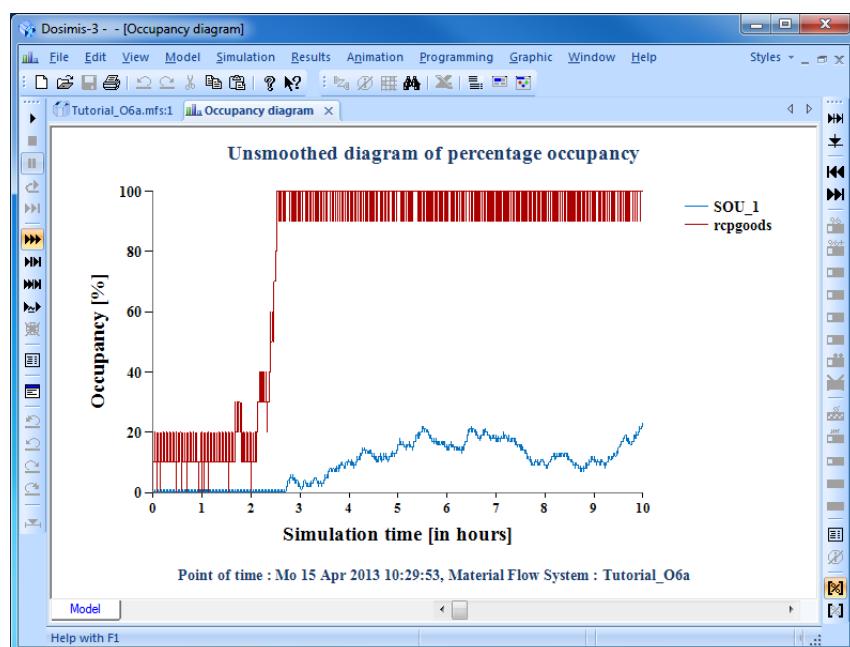


Figure 4.12: Occupancy diagram of the source and its succeeding conveyor (10 h)



Module histogram of work stations: As you can see both work stations still have free reserves (green, idle time). These reserves result from alternate filling up and emptying of buffers. An empty buffer causes a lack of parts at a workstation. These workstations are operated at the limit of their capacity.

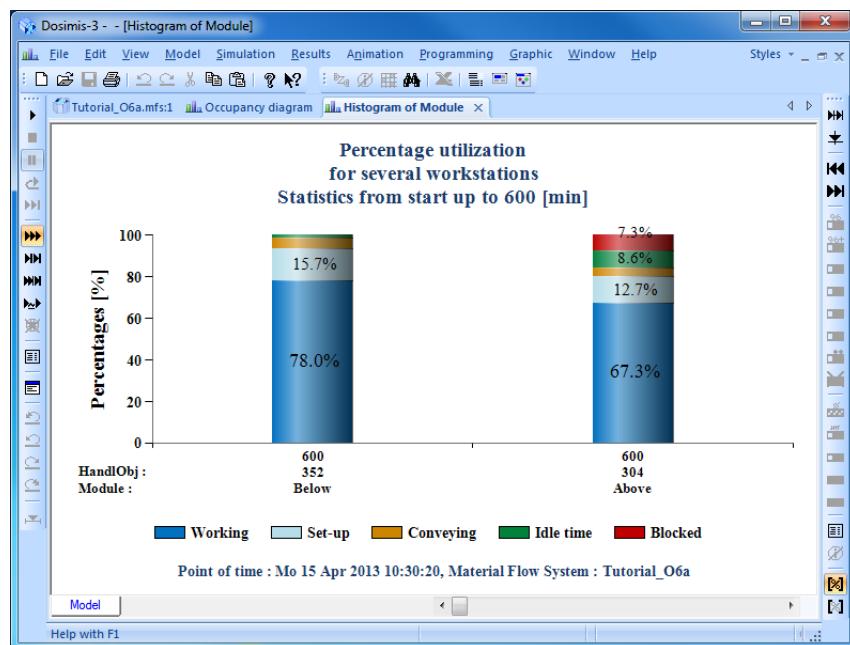


Figure 4.13: Module histogram of both work stations

Cause:

Because work stations reached the limit of their capability, reverse blockages are inevitable. This is also clearly visible in the diagrams of buffer occupation before work stations - they are filled up interchangeably again and again.

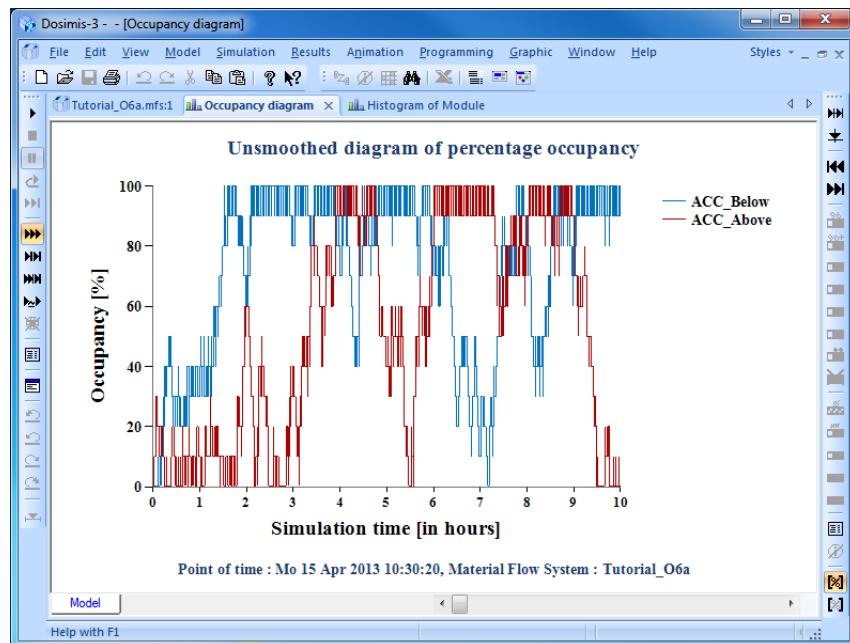


Figure 4.14: Occupancy diagram of buffer before work stations



Conclusion:

Since the system is exhausted and because work stations form the bottleneck, the only possibility is to use reserves.

64.2.7 Step 7 - Optimization of Set-up Time

Measure:

By assigning a high priority to the feedback conveyor as right-of-way strategy of the shuttle (to prevent the deadlock in step 1), a strategy of maximum set-up time indirectly results. It makes sense to reduce set-up time. One possibility to do this is to introduce a “lot size” at the feed back conveyor. In DOSIMIS-3 you can do this by inserting a “bulk conveyor (BUL)”.

The existing feedback conveyor with a capacity of 3 will be replaced by a bulk conveyor with a capacity of 8 and a preliminary accumulation conveyor with a capacity of 4.

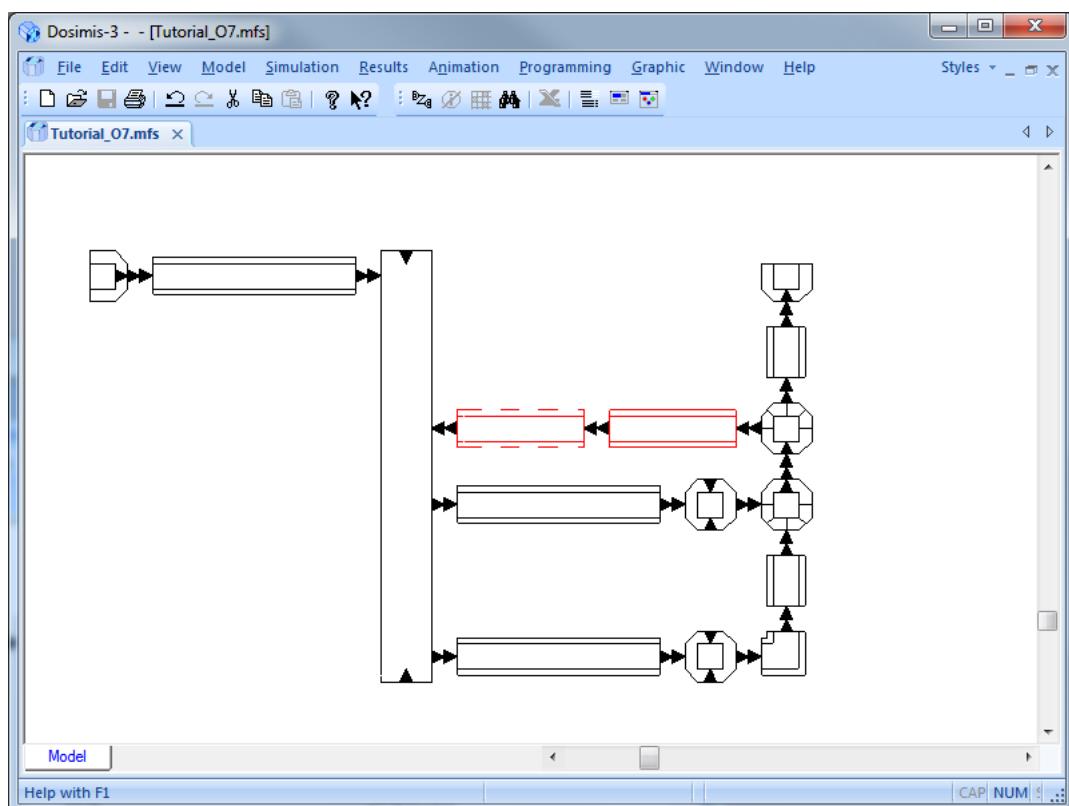


Figure 4.15: Layout with bulk conveyor and preliminary ACC

The bulk conveyor has following parameters::

length: 8 m
conveying speed: 0.2 m/sec
number of places: 8

Length indicates the total length, conveying speed defines unloading time.

A small buffer (with a capacity of 4) still has to be inserted between distributor and bulk conveyor. During unloading the bulk conveyor does not accept any object hat enters - so a deadlock situation might occur, which becomes more improbable because of the small ACC.



After this, please save your model and start a new simulation run.

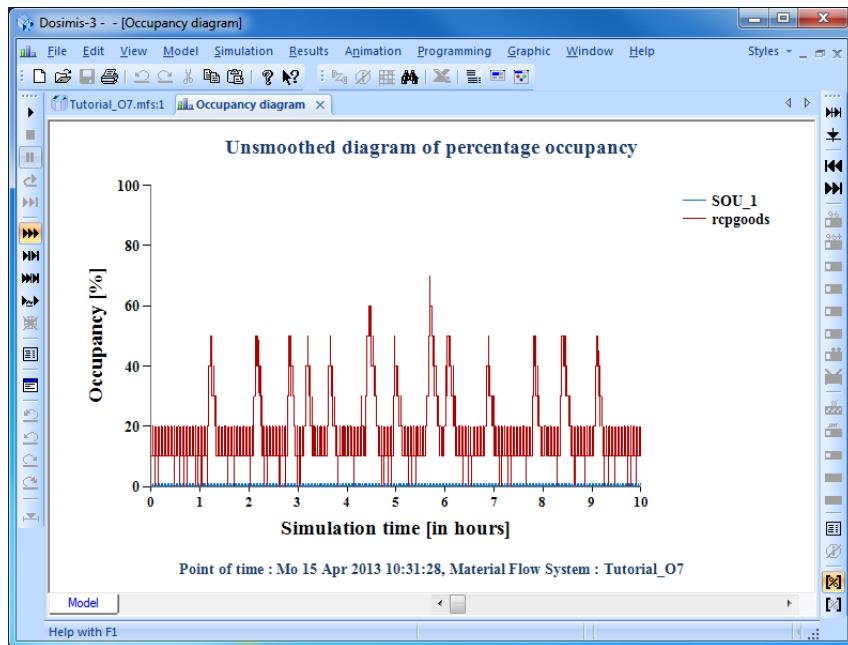


Figure 4.16: Occupancy diagram of the source and its succeeding conveyor

There is no more reverse blockage inside of the source - the system copes with the requested work load. Admittedly buffer capacity of the whole system is oversized. However the buffer before the sink is filled up often.

Cause:

The sink is still problematic. Due to its extreme distribution of departure time (exponential distribution) the sink repeatedly causes high loads inside the system. That is already visible in the Occupancy diagram of the buffer before the sink

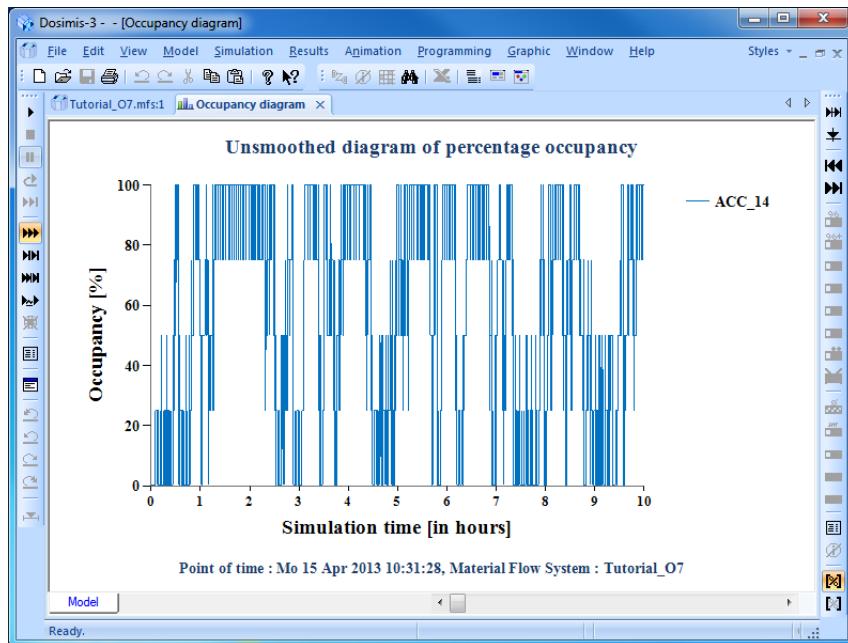


Figure 4.17: Occupancy diagram of the buffer before sink



64.2.8 Step 8 - Mitigation of the Sink

Cause:

By organizational measures in the plant (e.g. avoidance of cigarette breaks, stand-by workers in case of failures or overload, overcoming problem cases, etc.) it should be possible to attain an even workload of the sink.

Therefore we will analyze the behavior of the sink if “departure time” is normally distributed. Mean of 55 seconds is unchanged. The deviation is be 5 seconds.

Please start another further simulation run.

Start:

Occupancy diagram of the source and its succeeding conveyor: The source is able to dispense all palettes, sometimes little reverse blockages form on the accumulation conveyor due to deflation of the bulk conveyor.

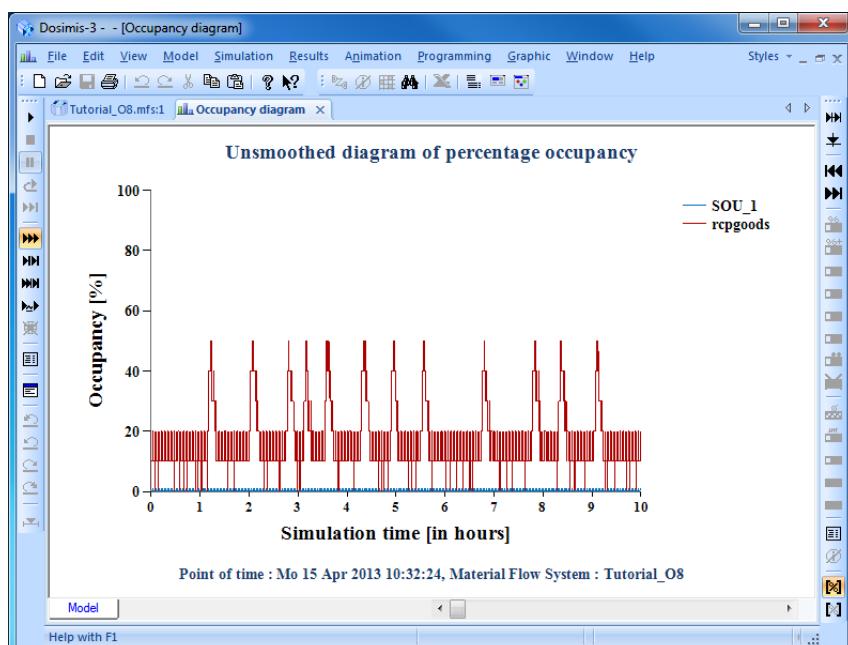


Figure 4.18: Occupancy diagram of the source and its succeeding conveyor



Module histogram of both work stations: as you can see, set-up times have decreased substantially and there are even “idle times” (about 15%) at the work stations again.

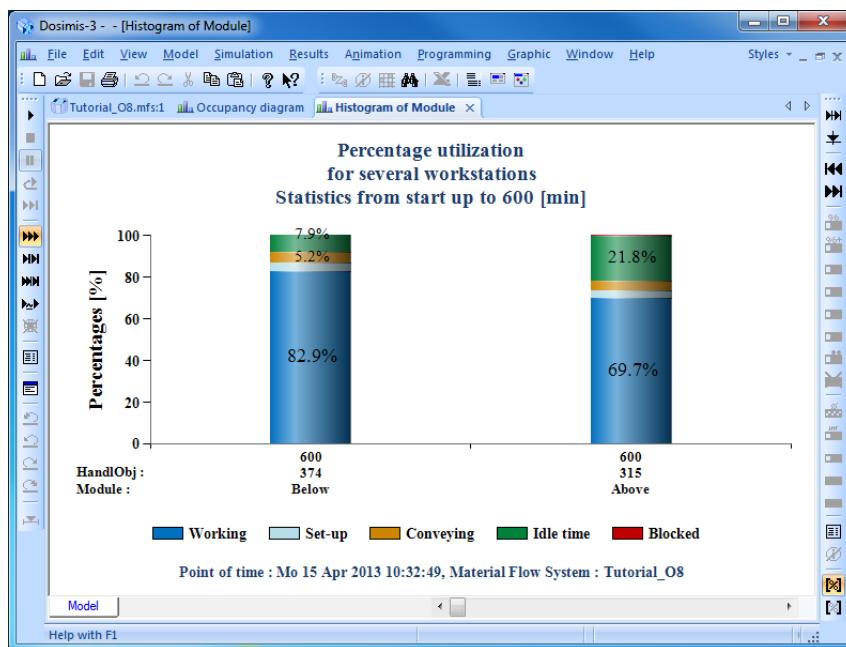


Figure 4.19: Module histogram of both work stations

Occupancy diagram of the buffer before sink: this buffer is barely filled to 50% anymore. More than three pallets are relatively rarely contained inside that buffer.

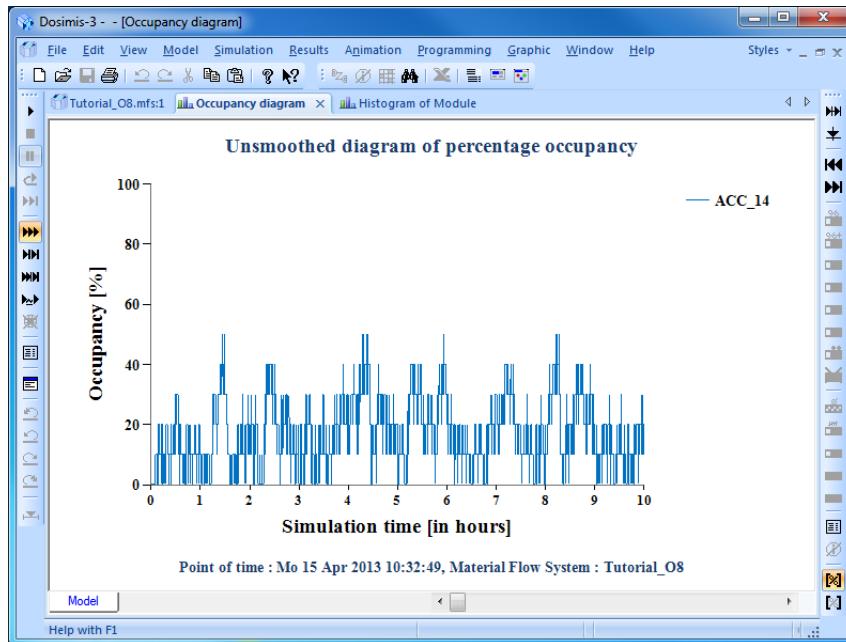


Figure 4.20: Occupancy diagram of the buffer before sink

Cause:

Due to changing the sinks dynamics (normally distributed) the buffer is hardly needed anymore.



64.2.9 Step 9 - Decrease of Buffer Size before the Sink

Measure:

The capacity of this buffer can be reduced to 2. Even if little reverse blockages appear at workstation “Above”, this should have little effect on the system - workstations have reserves.

Please start another simulation run.

Start:

Occupancy diagrams of the source and its succeeding conveyor as well as module histograms of both work stations show that reducing of buffer size before the sink has no noteworthy effect on the system as expected.

Please select **Results/Result parameter** from the menu bar, enter the values as shown in Figure 4.21 and click the “Accept” button. Then select **Results/Turnaround-time statistics** from the menu bar. Two bars are visible, each representing one product. The minimal, average and maximal turnaround time is displayed for product 1 and 2 -from source to sink.

As you can see, the maximum time for products to remain in the system is about 2 h - the average time of about 15 - 20 minutes appears to be amazingly high. The minimal turnaround time is about 5 minutes and roughly corresponds to the value estimated by seminar participants.

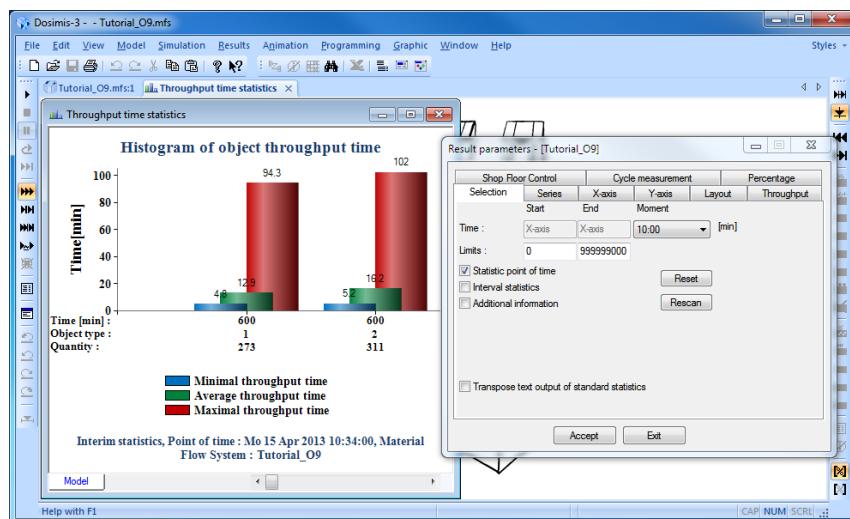


Figure 4.21: Diagram of turnaround time

64.2.10 Step 10 - Decrease of Buffer Size before Work Stations

Further Optimizations:

The reasons for high buffer occupations were explained already in Step 9. However, if this changing of buffer size has no effects on the throughput - what about further reductions? There are essentially several possibilities:

- a) Further decreasing the size of buffer before the sink. The work station could stand quite smaller blockades - it still has reserves. However the appearing blockades actually result in breakeven performance that will fix the planned interval of the source at 1 minute.



- b) Decreasing of bulk conveyor capacity. This is equivalent to the decrease of lot size, which means that quota of set-up time at the work stations will increase again. With that the breakeven performance of the system will be fixed as well.
- c) Decreasing the size of buffers before the work stations. As you can see at the Occupancy diagrams buffers are filled up occasionally - but there still are significant reserves in the incoming goods buffer.
- d) Minimizing the size of the buffer succeeding the source. The corresponding occupancy diagram shows that 5 places could be saved at once.

The resulting measure depends on the goals of the enterprise. On principle all measures affect in the same way - they reduce the range and flexibility of deviations of source frequency. Measures a), c) and d) affect the capital investments; measure b) furthermore affects the turnaround-time.

Measure:

Decreasing the size of buffer before the work stations from 10 to 8 places per conveyor - this measure reduces, first of all, the space requirements in the production area.

64.2.11 Step 11 - Factory Tuning

The examination of the statistics resulting from the measures in step 10 shows that the system generates the required performance.

Further Optimizations:

Sometimes planers have to answer the question: What has to be done to increase system performance - if necessary by changing organizational conditions. For example the question could be asked whether the production performance could be increased by 20 %, if rework is outsourced. This question shall be answered by a simulation run.

Measure:

Please increase the frequency of the source (decrease output time) and the sink (decrease departure time) by 20 %, that means:

Source, output time: 48 sec

Sink, leaving time: 44 sec

The distribution strategy of the distributor has to be changed - all objects must leave to sink.

Exit 1: Objects 1, 2, 10, 20

Exit 2: First delete the second entry then enter "99" in the first line (the first line has to contain a number greater than "0", so we enter "99" as dummy, therefore all objects will leave at exit 1).

Then start a simulation run.



Conclusion

As the Occupancy diagrams of the source and its succeeding conveyor show, the problem has been conquered.

A look at the “Turnaround-time statistics” displays a surprise: The average value of turnaround times is about 8 minutes now and its maximum is about 20 minutes (admittedly without regarding on rework!).

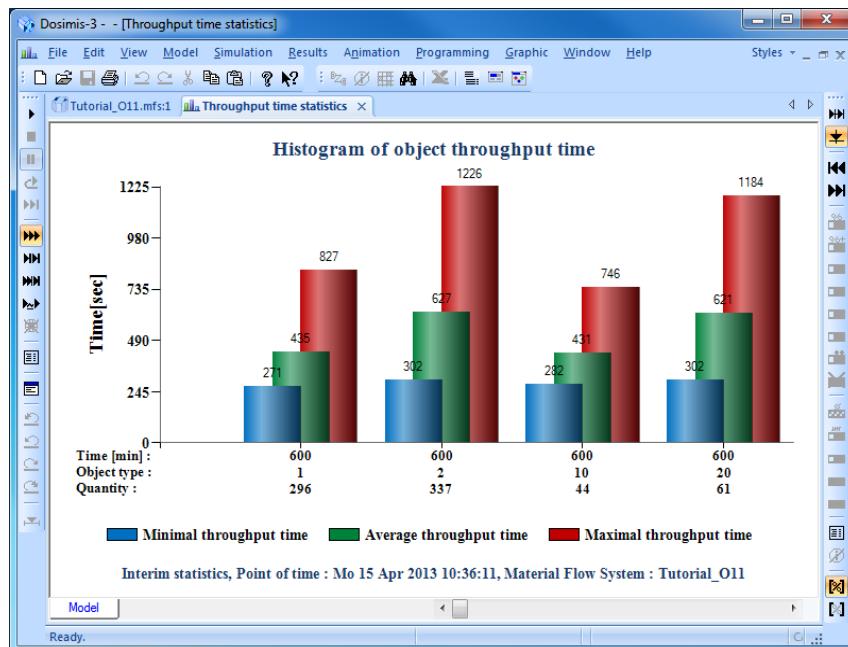


Figure 4.22: Diagram of turnaround times

The “small” rule of lot size multiplies turnaround time by about 3 (if rework is not regarded) - small measure, large effect.

Many production systems are much bigger than the one of this example. Often 500 or up to 3000 modules are used to reproduce a plant - using a lot of control rules (priorities, sortings, synchronizations) whose effects can barely be understood.

64.3 Results of this Simulation Study

In chapter 2.2.1 questions were asked that should be answered by simulation. Following answers result:

- Throughput of 60 parts per hour can be achieved, presumed that the measures taken can actually be accomplished in reality. The originally analyzed system did not achieve the work load approximately.
- The original system has a lot of weak points: shuttle (control strategy and performance), buffer sizes, quota of set-up times at work stations, and large deviation of sink performance.
- In the final model (step 10) workstations have a utilization of up to 90 %.
- The shuttle has a reserve of about 20 % resulting from the realized modifications.
- Behavior of the sink has substantial effects on the performance of the hole system - organizational measures are indispensable.

How the system behaves if failures occur has to be analyzed in further studies. Due to the bare design of work stations it can be assumed that the system reacts very sensitively to failures.





65 Graphical Comments

When using DOSIMIS-3 to create a simulation model no time-consuming layout true to scale will be generated, since these functionalities are reproduced by an appropriate parameterization.

The avoidance of an accurate layout illustration allows an extremely fast modeling as well as the screen-optimal display of the system to be simulated. Assistance tools, such as zoom, copying of modules or module groups and the input of partial models taken from libraries, are available.

Layout can be complemented individually by texts resp. comments and by the aid of integrated paint functions, in order to arrange the model more descriptively.

65.1 Insertion of Graphical Elements

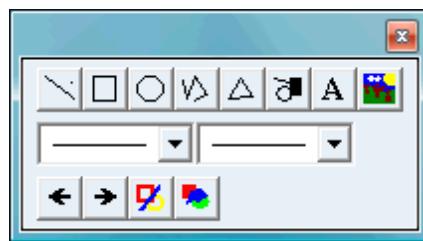


Figure 5.1: Graphic-palette

Please select **View/Tools/Graphic-palette** from the menu bar to activate the graphic-palette toolbar. You can create graphical elements in your model by using the buttons of the Graphic-palette. To insert a graphical element in your model, first click on the button representing the desired element on the “Graphic-palette”, then click at the point where you want to place the first edge of the element and click a second time on the point of the second edge.

65.2 Change the Size of a Graphical Element

To change the size of a graphical element, please select the element by mouse click and use the pulling points on its frame.

The grid is an invisible network of lines, which helps you to align the graphical elements. As a default the graphical element is drawn on the nearest intersection of the grid when placing. The default distance between two grid lines is 12. If necessary, you can change both the vertical and the horizontal distance between grid lines.

Please select **Edit/Snap parameters** from the menu bar to display or to edit grid parameters.



65.3 Add a Rectangle

To draw a rectangle, please click the rectangle button of the symbol bar called Graphic palette

Symbol



and place the element in your model. To change background or line color, please double-click the element to open the element dialog box.

By double-clicking on a rectangle following dialog-box appears.

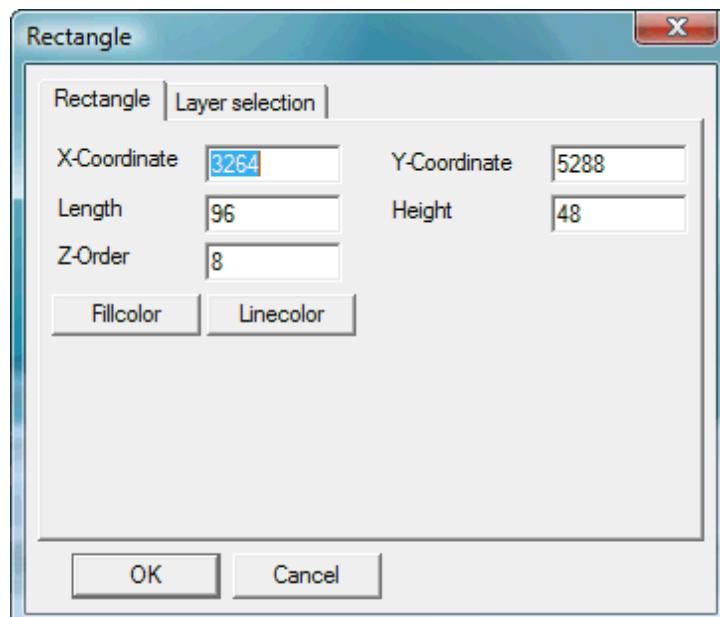


Figure 5.2 Dialog-box of a rectangle

With the help of the button “Fill color” you can set the background color of the element. By using the button “Line color” you can edit the color of the frame line of the element

If graphical elements are added in a model window, they will be stacked automatically on a unique level. The pile sequence is recognizable if elements overlap or do not appear at all. You are able to move a single element or a group of elements inside the stack, e.g. you can move objects inside the stack one level up or down by selecting the element or group of elements, then click the element with the right mouse button and select “Foreground” or “Background” from the context menu. Another way to change the position of an element inside the stack is to change the value of the “Z-Order” in the elements dialog-box. Each Z-order number is assigned to one graphical element inside the model. Z-Order number “1” is assigned to highest level in the foreground.

In order to achieve special effects, you can arrange elements so that they overlap and produce shadow effects.

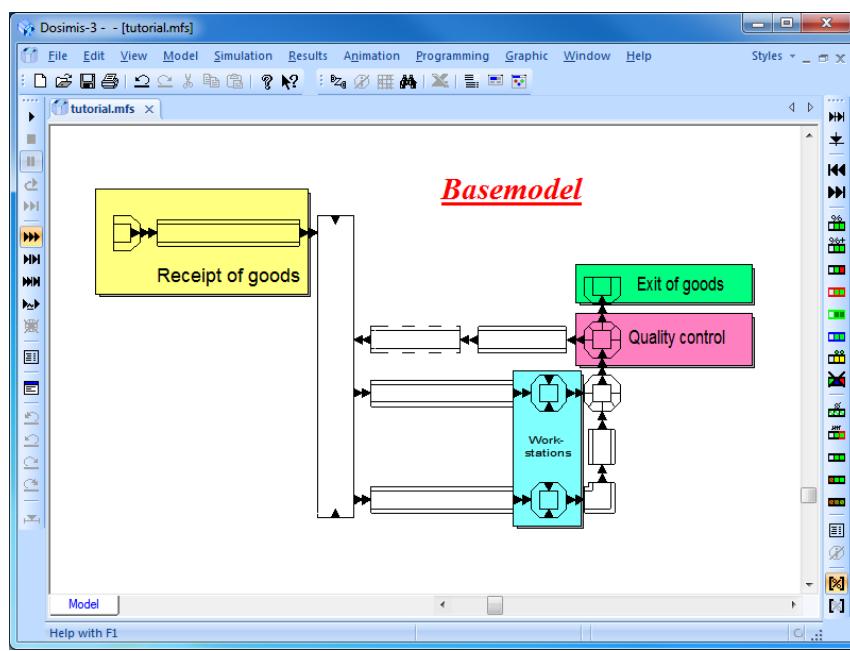


Figure 5.3 Completed layout including graphical comments



66 Summary: Data of the Study

66.1 Model Parameter

Source:	Objects are generated randomly:	Type 1 and 2 with same frequency (50 : 50)
	Distribution of output time:	normally distributed: mean 60 sec, standard deviation 5 sec
Accumulation Conveyor:	Conveying speed:	0.2 m/sec
	Length of segment:	1 m
	Capacity:	- after the source: 10 parts - before workstations: 2 parts - after workstation "Below": 1 part (edge conveyor - forward control) - after edge conveyor: 4 parts - feed back conveyor: 3 parts
Shuttle:	Loading path:	1.1 m
	Unloading path:	0.1 m
	Loading/unloading speed.:	0.2 m/sec
	Slowly driven path:	0.5 m
	Speed fast:	1.0 m/sec
	Speed slow:	0.1 m/sec
	Right-of-way strategy:	Priority to receipt of goods
	Distribution strategy:	destination with (object type)
	- Object types:	1,10 work station above 2,20 work station below
	Position parameter:	- Entrance 1: 0 m (receipt of goods) - Entrance 2: 15 m (feedback) - Exit 1: 20 m (to workstation "Above") - Exit 2: 25 m (to workstation "Below")
Workstation:	Length:	1 m
	Speed:	0.2 m/sec
	Working time:	normally distributed mean 80 sec, deviation 5 sec
	Quota of rework:	15 %
	Set-up time:	60 sec (with each change of type)
Combining station:	Conveying path:	1 m
	Speed:	0.2 m/sec
	Right-of-way strategy:	FIFO
Distributor:	Conveying path:	1 m
	Speed:	0.2 m/sec
	Distribution strategy:	destination with (object type)
	- Object types:	1,2 to the sink, 10,20 to the feedback conveyor
Sink:	Distribution of departure time:	exponential distributed, mean 55 sec

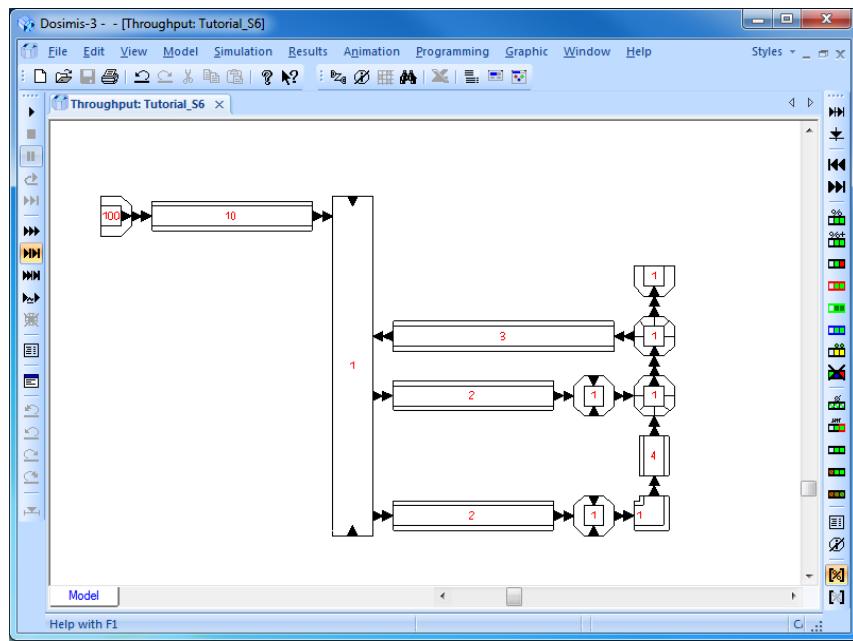


Figure 6.1: Module capacities

In order to enable or to disable the display of module capacity, please select **Model/Info.../Capacity of elements** from the menu bar.

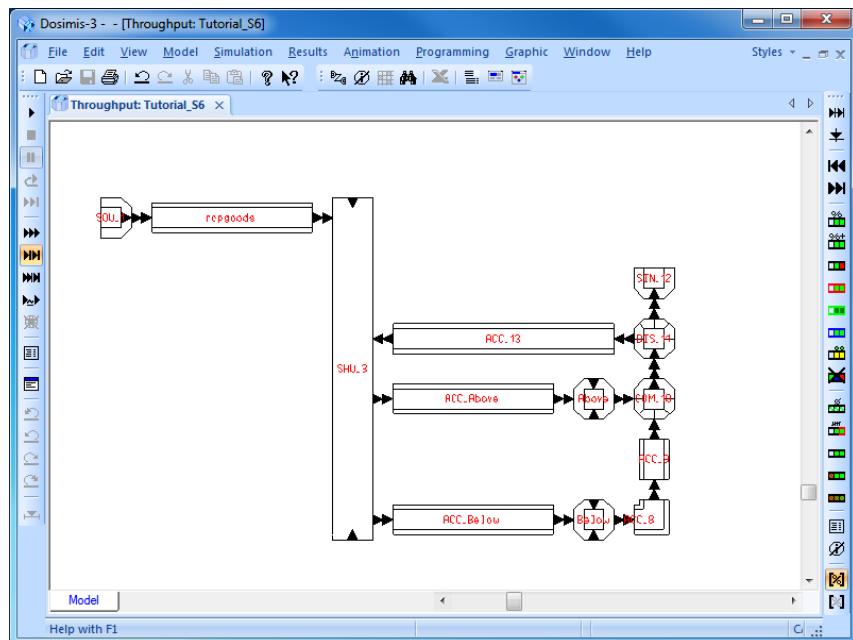


Figure 6.2: Module names

In order to enable or to disable the display of module names, please select **Model/Info.../Names** from the menu bar.

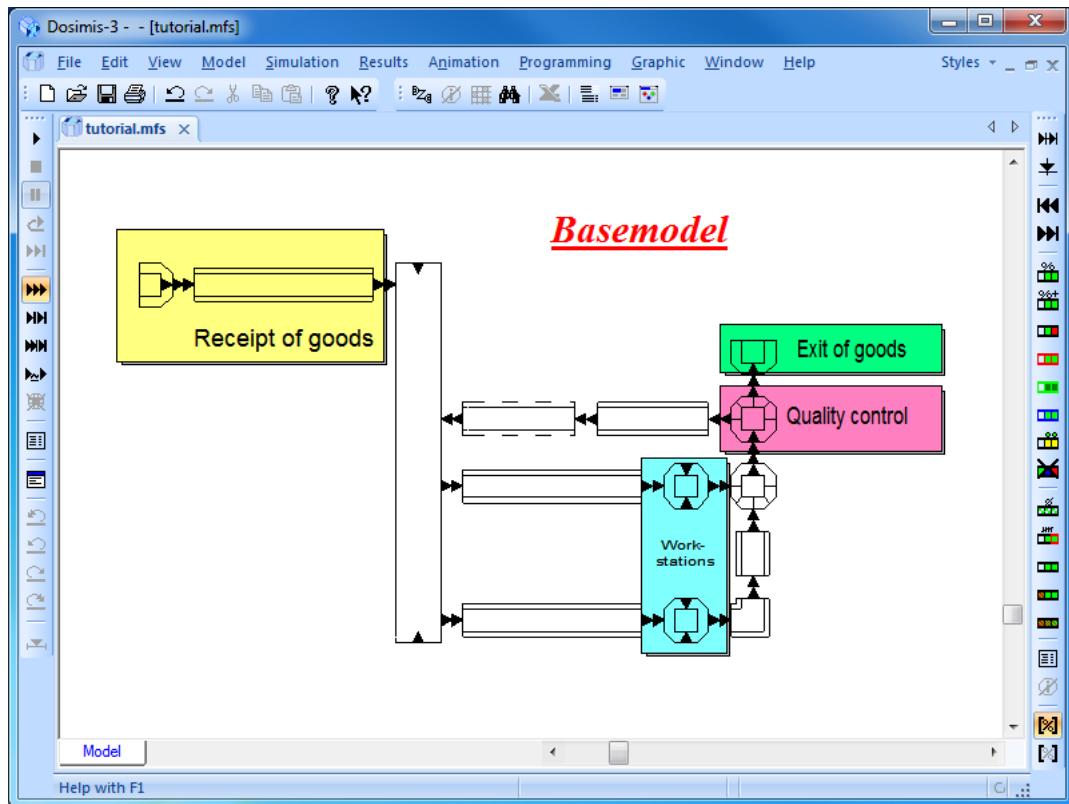


Figure 6.3: Final layout after optimization

66.2 Summary of all Simulation Runs

<i>Step</i>	<i>Characteristics</i>	<i>Measure</i>
1	Deadlock	Changing the entrance priority of the shuttle. High priority to feedback conveyor.
2	Presorting	Increasing the size of buffers before workstations from 2 parts to 4 parts.
3	Decoupling	Further increasing of buffer sizes before workstations up to 10 parts. Decoupling between shuttle and workstations.
4	Increase of shuttle speed	Increase maximum speed of the shuttle to 2 m/sec.
5	Disposal preferred	The distributor after workstation "Above" prefers to dispose the material flow out of this work station (Priority 1 to the entrance connected to this work station).
6	Buffer before the sink	Inserting a buffer before the sink (with a capacity of 10).
6a	Prolongation of simulation time	Simulation time doubled to 600 min.
7	Minimizing of set-up time	Inserting of a bulk conveyor to form lots in the feedback wing (including a small preliminary accumulation conveyor).
8	Mitigation of the sink	Changing the type of departure time distribution from "exponentially distributed" to "normally distributed" keeping the same mean and setting deviation to 5 sec.
9	Decrease of buffer size	Decreasing the buffer size to 2 parts.



	before the sink	
10 a)	Decrease of buffer size before work stations	Decreasing the size of both buffers to 8 parts.
10 b)	Further Measures	Further decrease of buffer or lot size.
11	Factory tuning	Decrease of cycle times of about 20 % (Source 48 sec., Sink 44 sec.). Rework is sent to the sink.



67 Introduction to the Tutorial (Part 2)

67.1 Structure of the Tutorial

By the first part of the Tutorial the user shall be enabled to create the planned simulation models, to edit parameters and to modify the model structure by himself. Part 2 of the Tutorial is a short overview of further functionalities as well as special module groups of DOSIMIS-3.

Part 2 of the Tutorial is divided to following chapters:

1. Structure of the Tutorial
2. Faults and Breaks
3. Work Areas, Workforce

This Tutorial is to facilitate the understanding in further functionalities of DOSIMIS-3 for you. However, please do not expect all functionalities of individual addressed topics to be explained in completeness. The Tutorial does not replace the user's manual. This would go beyond the scope of this Tutorial. User's manual/Online help and the Tutorial can help you to assimilate enough knowledge about DOSIMIS-3 to be able to use it without any training.

67.2 Icons

In order to facilitate orientation in this Tutorial for you, we arranged the text into sections of special functionality and marked these by appropriate symbols or icons. The following icons are used:



Examples or steps that may help you to find your way when using DOSIMIS-3.



Please pay attention to this important notes, which are marked by this icon.



Attention: this icon indicates a warning. The facts described in this section lead easily to errors, problems and deadlocks.





68 Failures and Pauses

68.1 Task

In part 1 of the Tutorial the model of a small production system was analyzed. In continuation of this example now the modeling of failures and pauses is supposed to be explained.

During operation it was determined that faults occurred at both work stations frequently. This faults reduced the utilization to 95 %. Furthermore the records proved that the average downtime was at 5 minutes. In a further analysis it is to be examined whether the plant is able to achieve the requested throughput in spite of this performance loss.

68.2 Theory

The following chapter shows to you how to reproduce failures resp. pauses. The parameterization of failures and pauses is completely identical. The distinction is only used for separate statistical recording of component standstill times.

Failures and pauses can be parameterized for all modules and module groups. These may be operational interrupts as e.g. break, shift changeover, group discussions, set-up and cleaning times or unpredictable faults. Additionally rejects resp. rework as well as logistic failures can have big influence on the system. All these can be reproduced as failures too.

The “time between failure (MTBF)” and the “time to repair (MTTR)” of an interrupt are determined in each case according to a selectable kind of distribution. The “time between failure” describes a time interval between the end of one fault and the beginning of the next fault. The “time to repair” designates the duration of a fault.

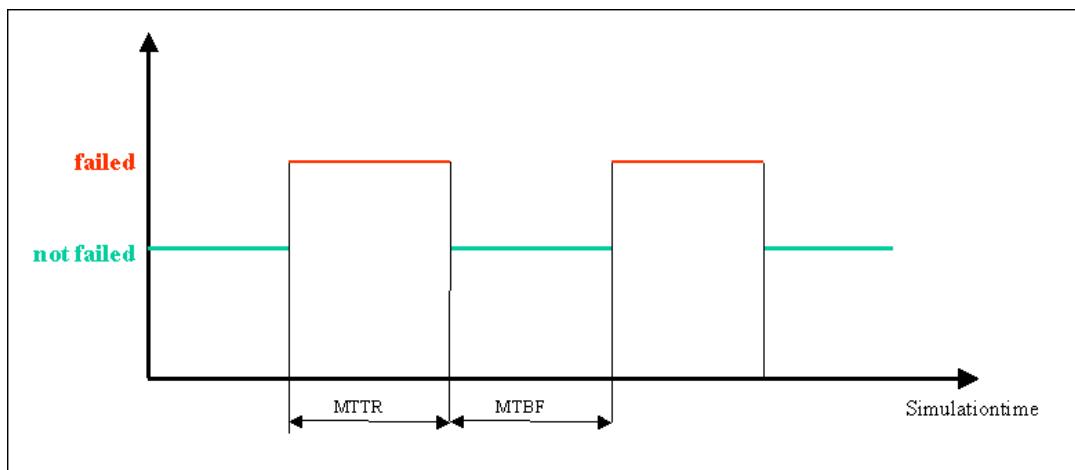


Figure 68.1 “Time between failure” and “Time to repair”

Thus the availability results:



$$\boxed{Availability = \frac{MTBF}{MTBF + MTTR}}$$

Figure 68.2 Availability

The “time between failure” is calculated from the “mean time to repair” and the “availability”:

$$\boxed{MTBF = \frac{Availability * MTTR}{1 - Availability}}$$

Figure 68.3 Time between failure

Regarding an availability of 95 % and a “time to repair” of 5 minutes a “time between failures” of 95 minutes results.

68.3 Including of Failures in the Simulation Model

The availability of work station “above” shall be 95 %. The work station shall fail in irregular time intervals. The fault is supposed to be reproduced randomly (stochastic) and not in fixed time intervals. The “mean time to repair (MTTR)” is “exponentially” distributed with an mean value of 5 minutes.

The “mean time between failures (MTBF)” will be normally distributed with a deviation of 10 % and a mean value calculated as shown above, depending on availability and MTTR.



At first you will be shown how to define a failure resp. pause. Please reproduce a “failure” as described below:

- Open the model „tutorial2.mfs“.
- Save the model with a new name as „tutorial2S.mfs“
- Select “View” → “Controls Palette” from the menu bar or press the “F2” button while pressing the “Ctrl” button on your keyboard.



- Select the break symbol (BRK)  from the controls palette and place the module in the work area of your model by click on the left mouse button.
- Please select “Model” → “Linking active” from the menu bar or press the “F9” button on your keyboard to activate the linking mode.
- First click on the break module with the left mouse button to select the module. The selected break module is displayed in blue color now. Then click on the module or group of modules to be failed. Modules that are connected to the selected break module will be displayed in red color in linking mode.
- At last deselect linking mode and save your model.

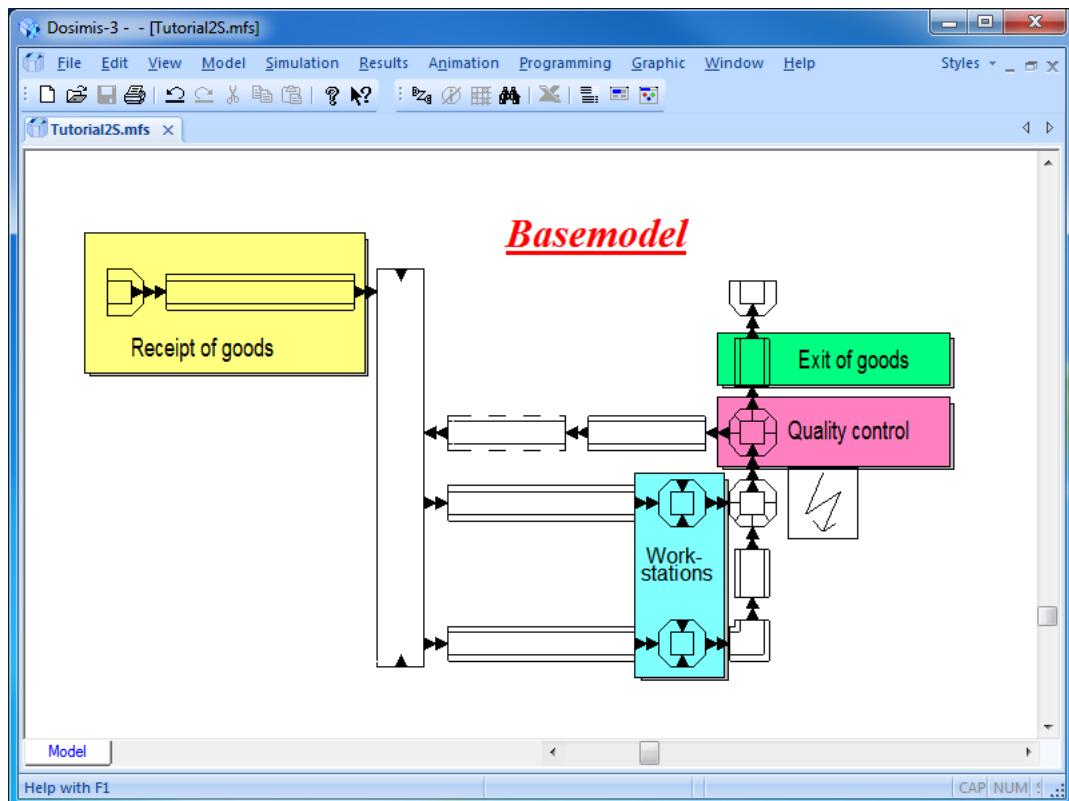


Figure 68.4 Linking work station to break module



68.4 Editing of Failure Parameters

Now the parameter of the failure can be entered. Double-click on the break module to open the parameter input mask.

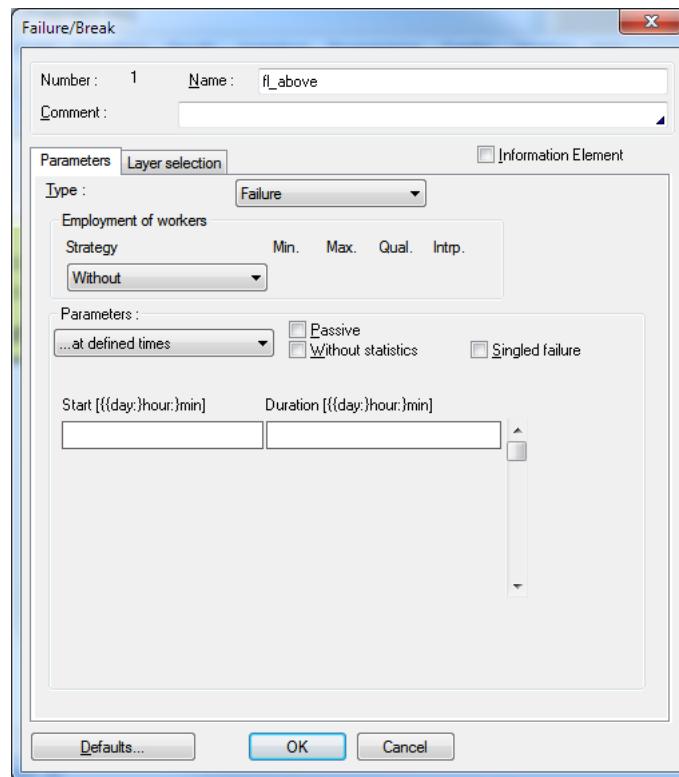


Figure 68.5 Parameter input mask of Failures /Pauses



To define the kind of break (**failure/pause**) select the according “type” from the combo box. No “employment of workers” is planned.



- Rename the break as “fr_above” (default: BRK_1).
- There are several kinds of failure that you can select from the “failure” combo box. Select “...at random” from the “failure” combo box.
- Now select “normally distributed” from the “MTBF” combo box and enter 5700 sec as “mean” and 270 sec as “deviation”.
- Then select “expo. distributed” from the “MTTR” combo box and enter 300 sec as “mean”.

Finally the parameter input mask of the failure module looks like following:

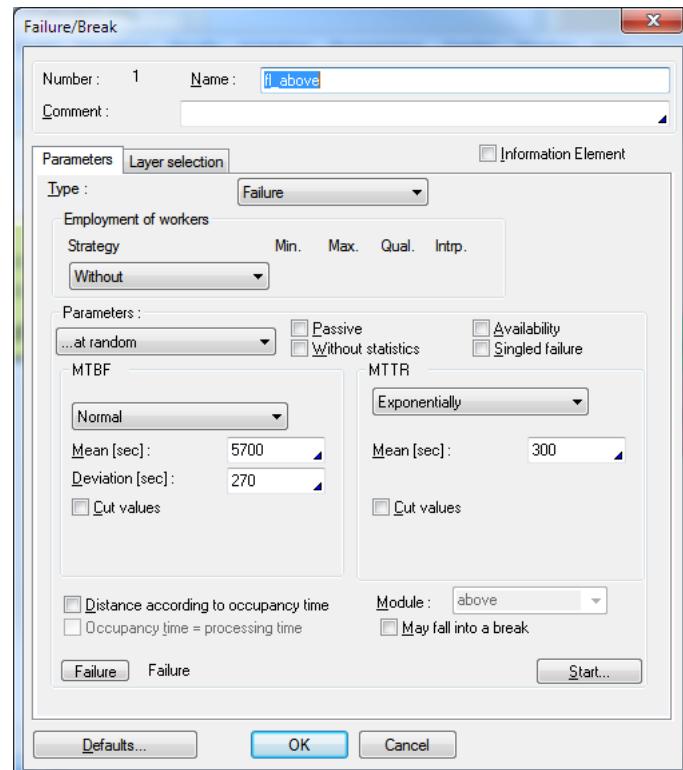


Figure 68.6 Completely filled out parameter input mask of the failure



Checking the failure parameters by using the “failure” button on the parameter input mask (available only for failure “...periodically” or “...at random”).



When using the “Availability” function DOSIMIS-3 will calculate the mean value of the “MTBF” according to the entered availability:

- Enter the “MTTR” parameters as described before.
- For “MTBF” enter the value of “1” as “mean” and as “deviation”. This is necessary to avoid any consistency check error.
- Select the check box called “Availability” and enter the desired value (e.g.: 95) into the “Availability[%]” input area.
- Now deselect the “Availability” check box. As a result the calculated “mean time between failure” is displayed inside the input area called “mean[sec]”.

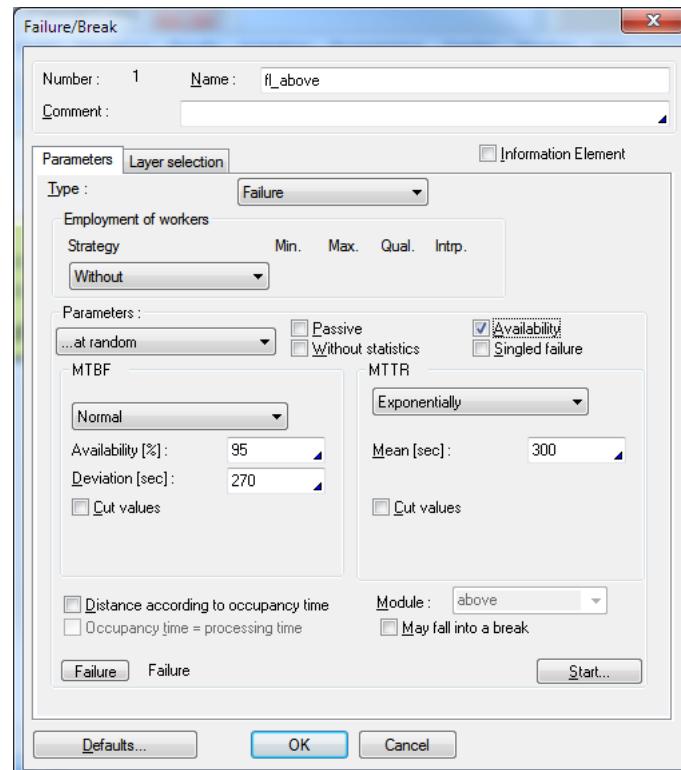


Figure 68.7 Determination of MTBF by using the availability check box



68.5 Analysis of Failures

After the simulation run is completed, you can start the animation and watch the movement of objects inside the model. Please select “View” → “Tools” → “Animation-toolbar” from the menu bar to activate the animation-toolbar in your DOSIMIS-3 window. Then click on the button “time factor” on the animation-toolbar and start the animation by click on the “start” button. The first failure occurs after about 1,75 hours.



While a module is failed, it is displayed in red color.

While a module is paused, it is displayed in blue color.

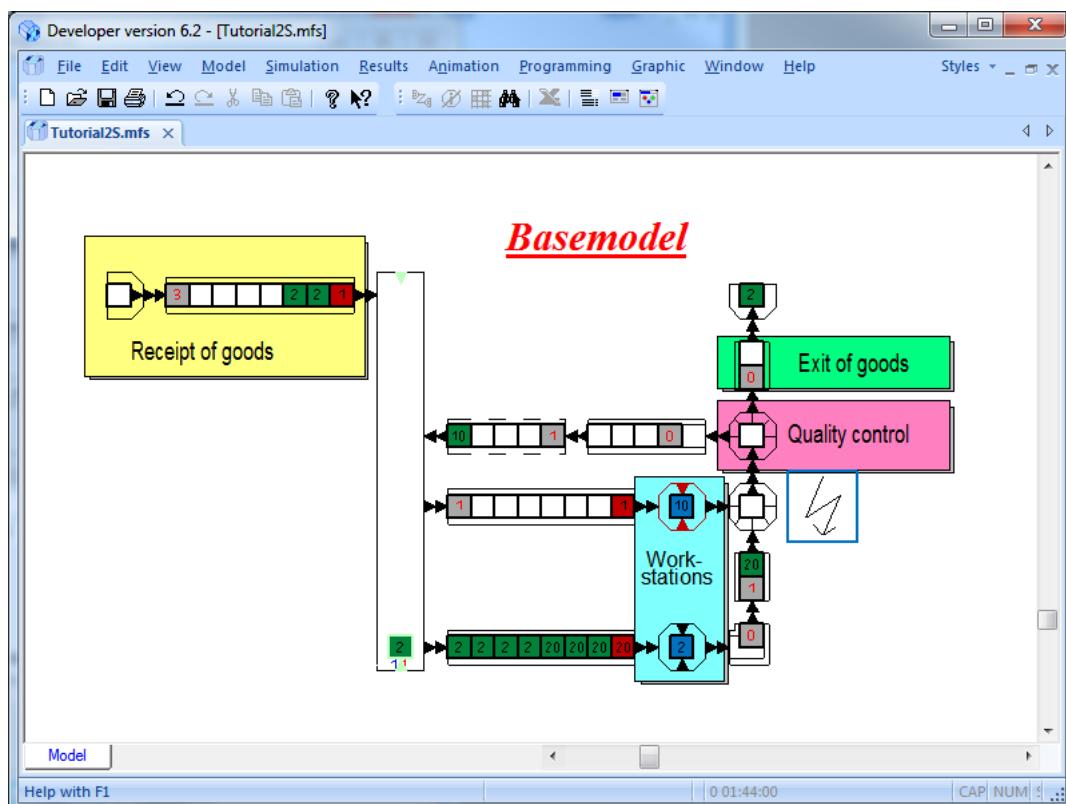


Figure 68.8 Animation



If a failure occurs all processes of the assigned modules are stopped. It is possible to consider the state or failure of one module for strategy decisions of other modules.

If a failure appears after a strategically decision has been done (e.g. distribution strategy), this decision will not be changed afterwards, that means that the object has to wait until the failure disappears.

In DOSIMIS-3 it is possible to assign one module to several break modules, especially if the user wants to separate between failures and pauses or if different kinds of failure shall be considered. These may overlap what is according to the theory of random distributions.

Failures are unpredictable interrupts, on the other hand pauses are planned. The frequency of both kinds of interrupts can be determined by the separate data collection. If a module is failed and paused at the same time, then the time of



overlap is imputed to failure time.

Please first select work station “above” by mouse click and then select “Results” → “State diagram” from the menu bar to open the state diagram window. As you can see a failure appears about 100 minutes after start. There are no further activities until to the end of this failure. The failure has to be finished before a new object is allowed to enter the module and will be processed.

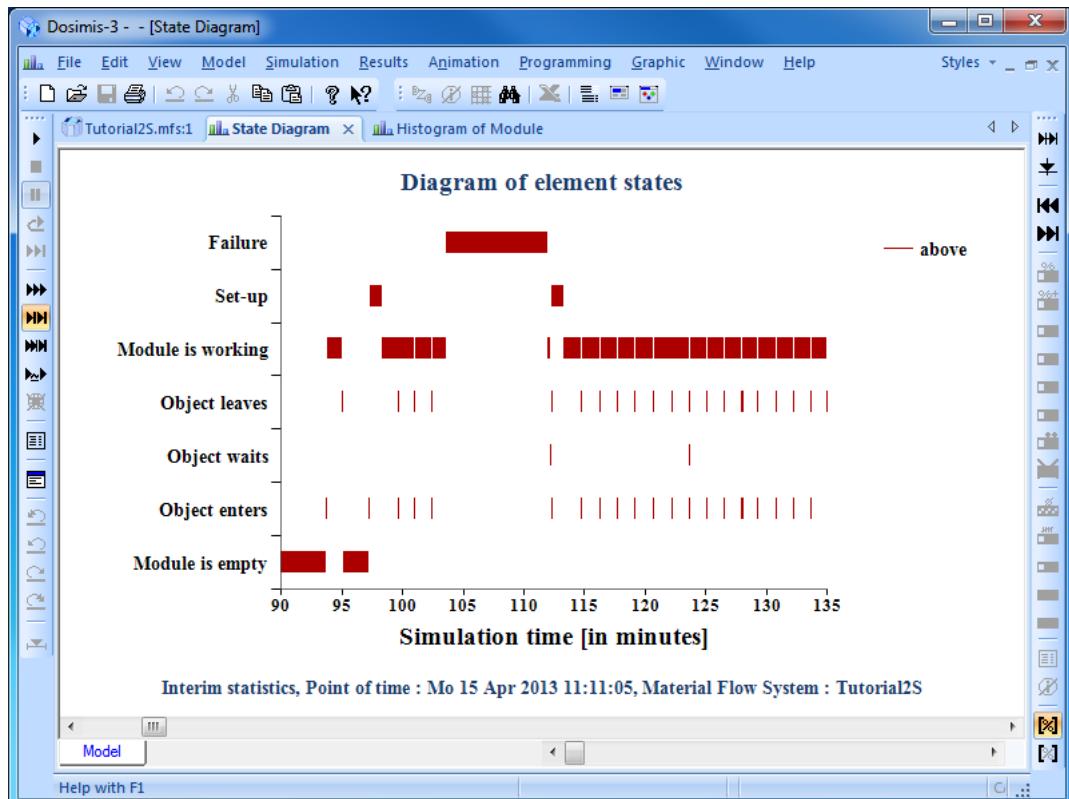


Figure 68.9 State diagram



Now the failure quota, that results from the assigned failure appeared during the statistics period, is displayed in the module histogram too.

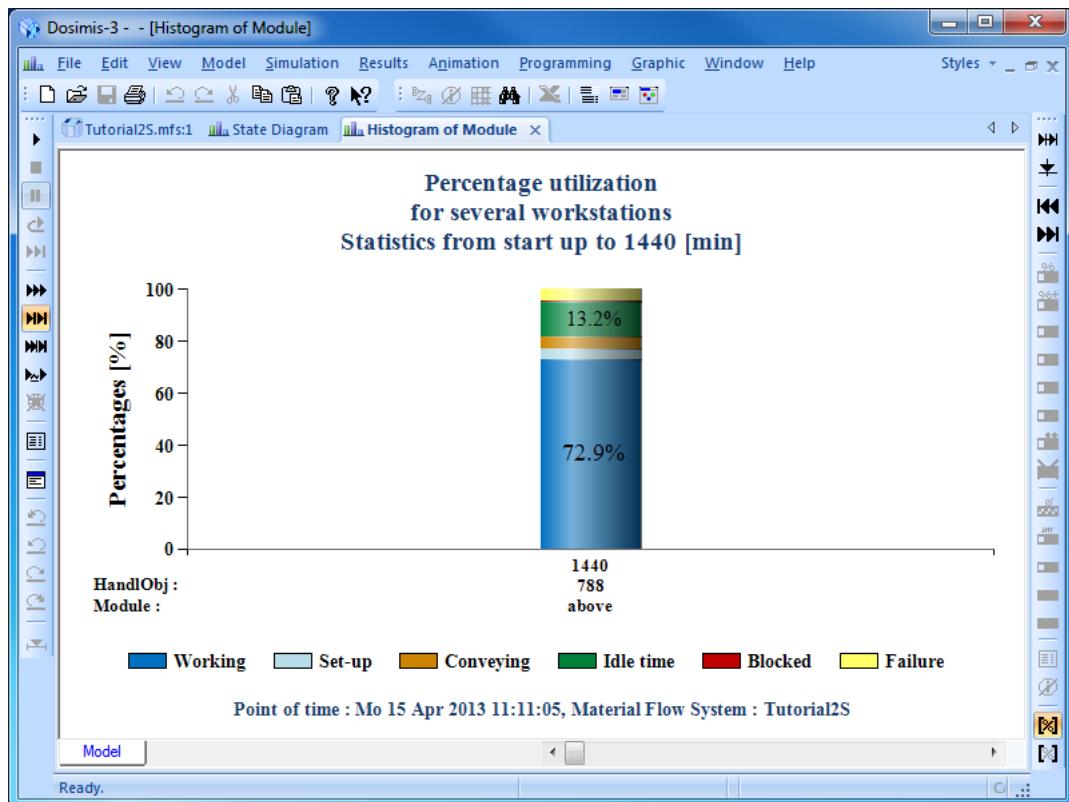


Figure 68.10 Module histogram



68.6 Statistics File

In order to open the statistics file called “*tutorial2s.slg*” please select “Results” → “Statistic data” from the menu bar. The “Final Statistics” window opens. Use the tree view on the left side of this window to browse for the statistic data you want to see. It is not possible to edit or to copy any data of this file.

The final statistics of material flow system *tutorial2s* gives a global overview on simulation results. Module performance values contain general information like throughput or utilization. A failure related statistic of all modules is to be found as “classification of utilization” of the module type. This displays a quota of failure of about 4.3 % for work station “above”.

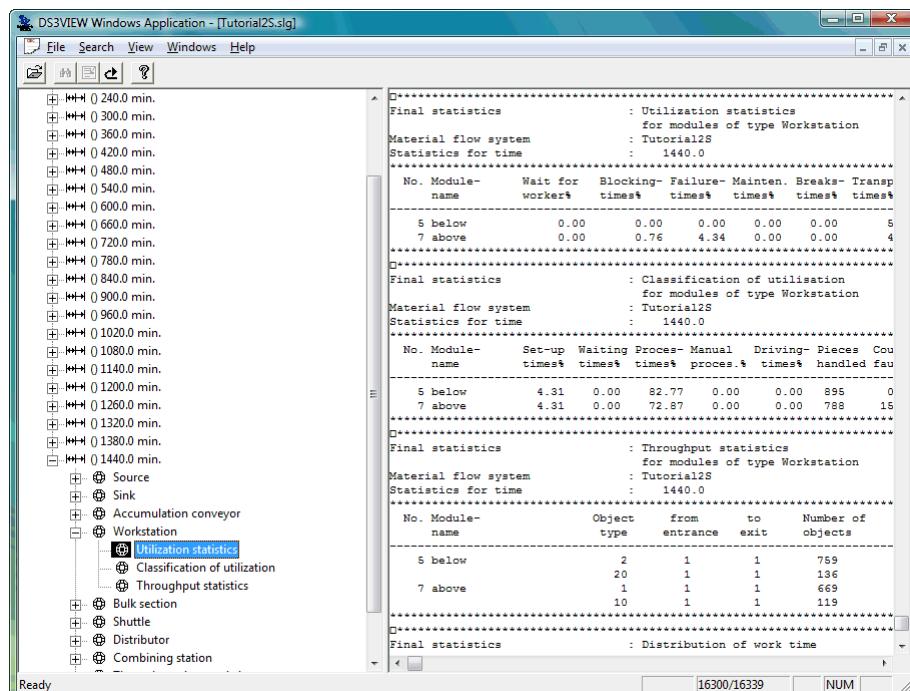


Figure 68.11 Final Statistics (classification of utilization of workstations)

At the bottom of the statistics list you will find the evaluation of duration and interval of failures and pauses. So for the break module “fr_above” the following results are displayed.

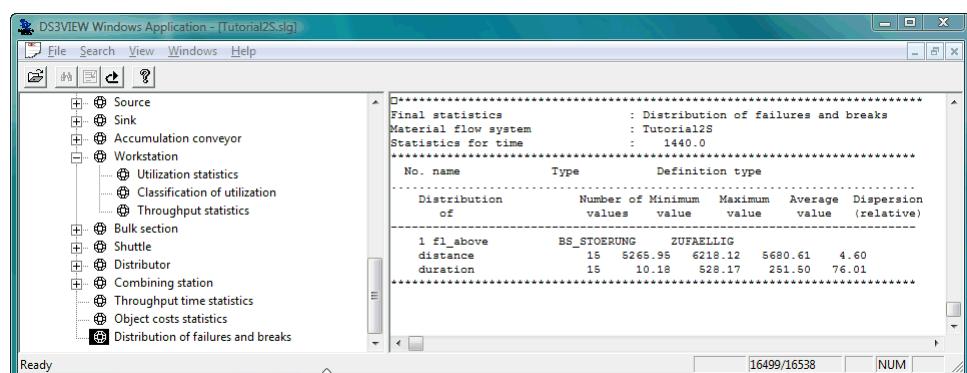


Figure 68.12 Final Statistics (Failures)



The analysis of the defined failure shows however a serious problem. Despite the failure quota was set to 5% the simulation provides a failure quota of 8.53%. The reason for this is to be found in the number of failures that appeared during simulation time. Fourteen faults appeared during this period. The number of events is to small to approximate the chosen distribution of the MTBF and to receive a representative mean time. "Good natured" distributions (such as fixed time, uniformly distributed, normally distributed) will achieve the mean value very soon., since in best case the mean may be hit after chosen randomly twice. In case of the exponential distribution it is necessary to chose randomly several times to achieve the desired mean value. In our example the maximum chosen value is 1613 seconds. After this value has been chosen, e.g. it is necessary to choose furthermore at least five times to reach the mean value of 300 seconds if the mean value of these five choices is about 38 seconds. The number of at least 50 events is to be regarded as a benchmark for this kind of distributions. Therefore the simulation time in our example has to be increased. The "simulation time" now will be set to 7 weeks and the "statistic interval" will be set to one day. Additionally the "Pre-run" time is set to one day too.

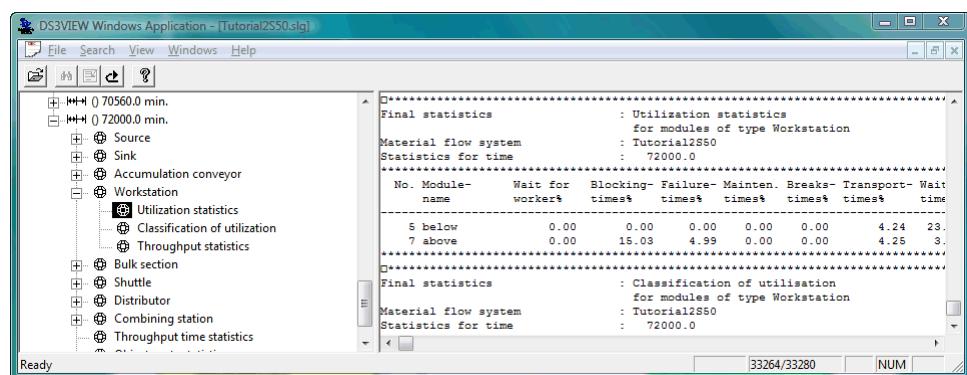


Figure 68.13 Final Statistics (classification of utilization of workstations)

The analysis of failure now results a quota of 4.99% failure. For this 705 events have been considered

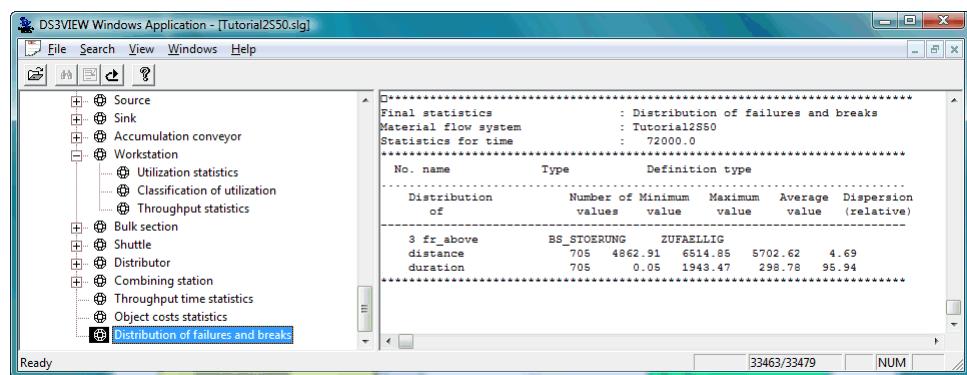


Figure 68.14 Final Statistics (Failures)



68.7 Task

Now also the work station “below” shall be failed

- Please select the break module called “fr_above”.
- Then use the copy function to copy this break module and move it to the favored position.
- Double click on the new break module.
- Enter “fr_below” into the input area called “Name”, and then click on the “OK” button of this module.
- Press the „F9“ button on your keyboard and deselect the module called „fr_above“ by click with the left mouse button on this module.
- Now select module „fr_below“ by click with the left mouse button on this module.
- Press the „F9“ button on your keyboard again to finish connecting mode.



Please select “Model“ → “Info” → “Connections” from the menu bar or **click the right mouse button while pressing the “shift” button on your keyboard** to display all links done in the opened model. To display links of the new break module named “fr_below” only, please click with the left mouse button on that module

To hide all links, select again “Model“ → “Info” → “Connections” from the menu bar or **click the “Finish View” button** from the „Modeling“ bar.

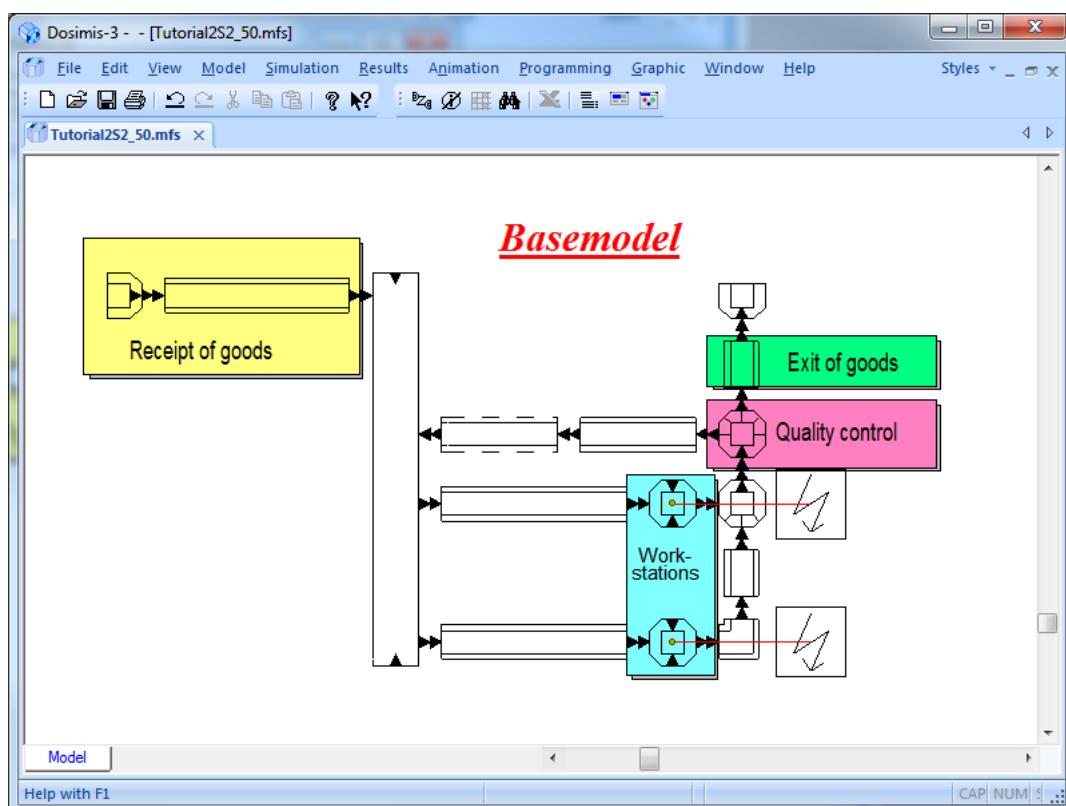


Figure 68.15 Links



68.8 Shift Model

For the analysis of large periods often it is reasonable to include the shift model of the real production line as a parameter in the simulation model. So it is easier to determine the time when an effect appears, because it is not more necessary to add shift breaks and pauses to simulation time.

In this example the plant works in a tow-shift operation mode for five days a week. Furthermore all pauses of work time shall be considered in the simulation model, too. During pauses the conveying system and the sink continue to work, but work stations, source and quality inspection are paused. Outside of the shift times all components are to be turned off.

The used Shift model looks like following:

Pause: 9:00-9:15; 12:00-12:30; 16:00-16:15 and 19:00-19:30

Daily work time: 6:00-22:00; on Friday only 6:00-16:30

For a better understanding a reference time corresponding to simulation time „0“ should be specified. In this example the reference time should be Monday 0:00 o'clock. This facilitates the assignment of times in the model.

Three breaks/pauses will be defined in addition. The first break module is to reproduce all daily pauses. A periodical pause will be defined that repeats every day. Start time and duration of all pauses have to be entered into the appropriate input boxes in the failure area of the break module parameter mask. Because start of period is 0:00 o'clock all the start times of pauses match the real times, no offset has to be added. Please regard to enter the duration and not the end time of the pause. Only work stations, the source and the quality inspection station are to be connected to that break module.

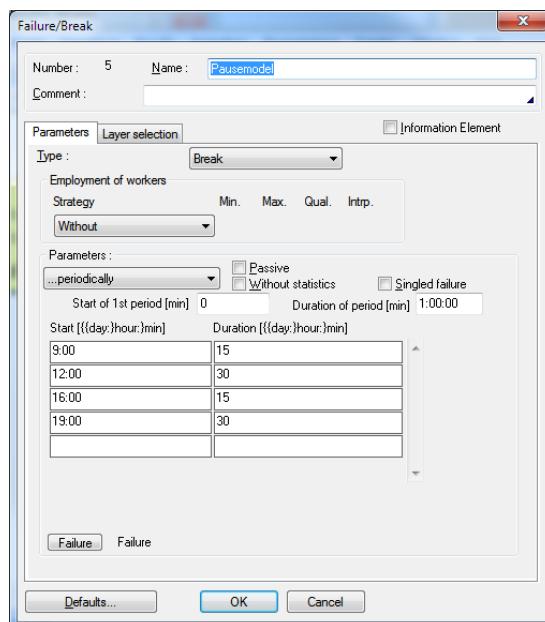


Figure 68.16 Shift Model of Pauses

The second pause is periodical, too. This pause reproduces the daily shift model. So only the first 6 hours from 0:00 to 6:00 o'clock and the 2 hours from 22:00 to 24:00 o'clock have to be



defined as pauses that stop the plant. This break module has to be connected to all modules of the model.

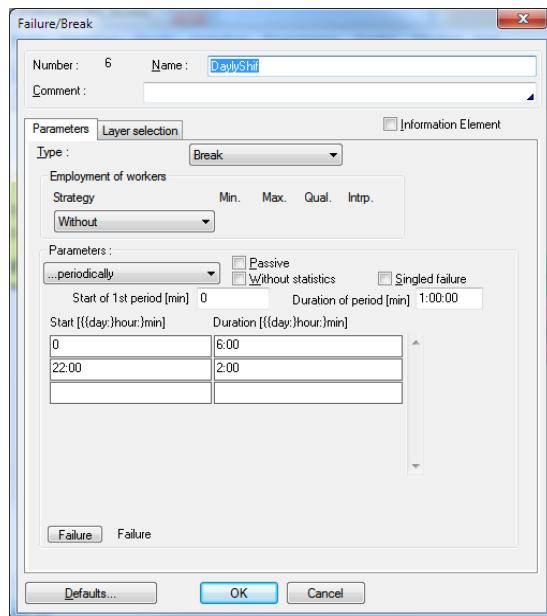


Figure 68.17 Shift Model of daily work time

The third break module reproduces a periodical pause with a period time of 7 days. In this module the pause starts on the 4th day at 16:30 o'clock and lasts to the end of the period (duration = 2 days, 7 hours and 30 minutes). Monday is day number "0", so Friday is day number "4". This break module has to be connected to all modules of the model.

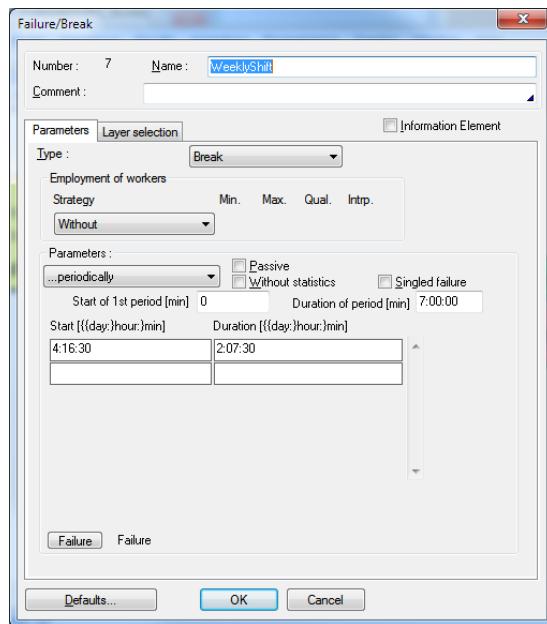


Figure 68.18 Shift Model of weekly work time

After a new simulation run, the behavior of the simulated plant is to be analyzed. To use minutes as scale for the x-axis in statistic data appears not to be reasonable, because simulation time is about 50 days now. So it might be wise to use bigger units instead. For this,



please select “Results” → “Result Parameter” from the menu bar, click on the tab “X-axis” and select the check box named “Time[DD:HH:MM]”. Now click on the tab “Selection” and make sure that the check box named “Intervalstatistics” is selected. Then click on the “Accept” button and close the “Result Parameter” window by click on its “Exit” button. In diagrams the time unit of the x-axis now is changed to “Day:Hour:Minute” (DD:HH:MM).

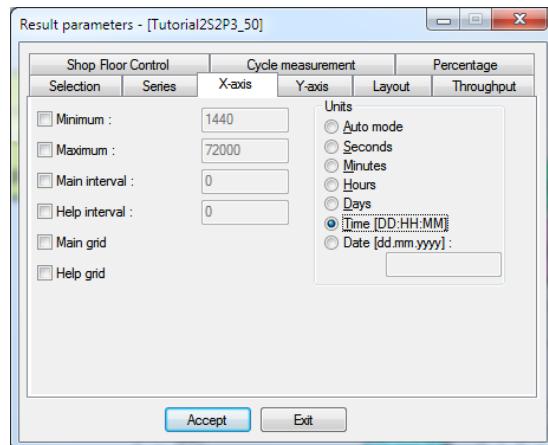


Figure 68.19 Result Parameter

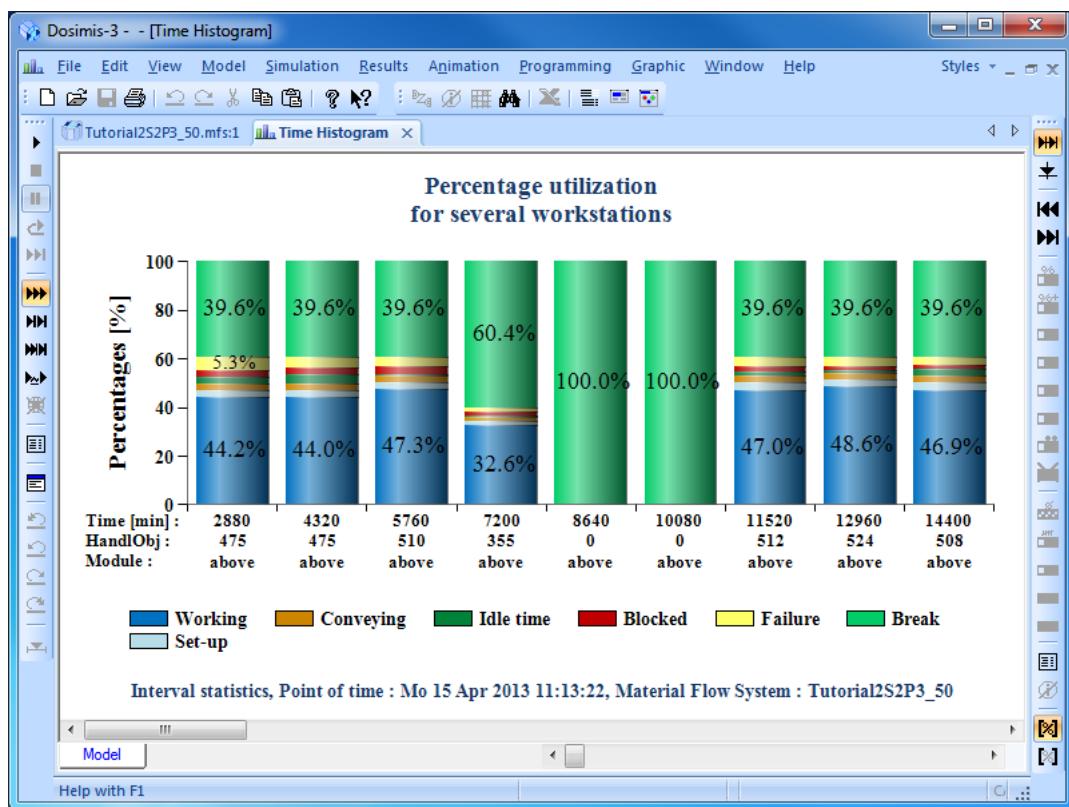


Figure 68.20 Statistic with Shift Model used

To display the diagram shown in Figure above, please select “Results” → “Time Histogram” from the menu bar. Please use the “scroll bar” at the bottom of that window to browse along the time axis. It might be fretful that a lot of pause times appear in that diagram during the considered period. So a big portion of time is displayed as pauses. To eliminate



pauses out of statistic calculations, in DOSIMIS-3 it is possible to filter statistic data of the regarded time period.

68.9 Filter Failures and Pauses out of Statistic Data

In a DOSIMIS-3 model you are able to filter times of failure or pause out of statistic data. Double click on the break module to be filtered.

Select the check box named “**Without statistic**”.

Click on the “OK” button to close that break module and start simulation again.

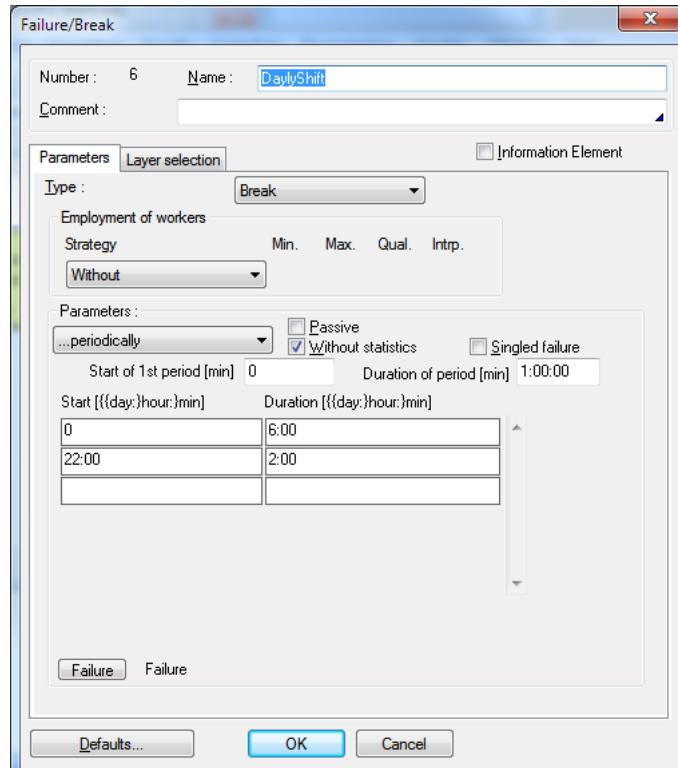


Figure 68.21 Activate Filter for statistic data in a break module



In that example the shift model of daily and weekly work time is to be filtered. Now the portion of failure or pause is displayed as a thin bar (named “without stat.”) on the left side of the main bar in the “Module Histogram”.



Take a look at the “Time Histogram” of the work stations again after simulation is completed. The two blocks without statistics are remarkable. These blocks represent weekends of the simulation period. Weekends are filtered out of statistic data. The day before, the portion without statistics is higher than to the remaining 4 days, since on Friday work time ends at 16:30 o'clock.

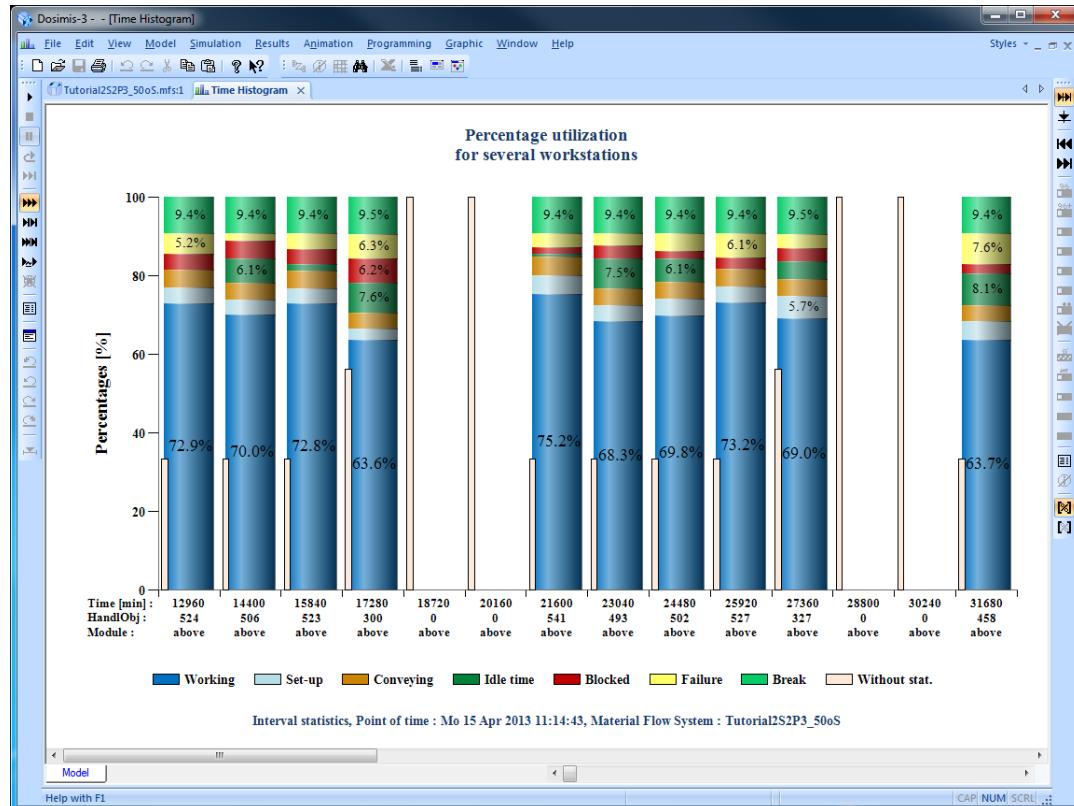


Figure 68.22 Module Histogram with statistic filter activated



The following diagram is to clarify again what it means to filter break times out of statistic data. 33,33% break time per day (8 of 24 hours) and 50% work portion per day (12 of 24 hours) connote, since the plant runs only for 16 hours a day, that in reality the work portion results to 75% of the running time (12 of 16 hours).

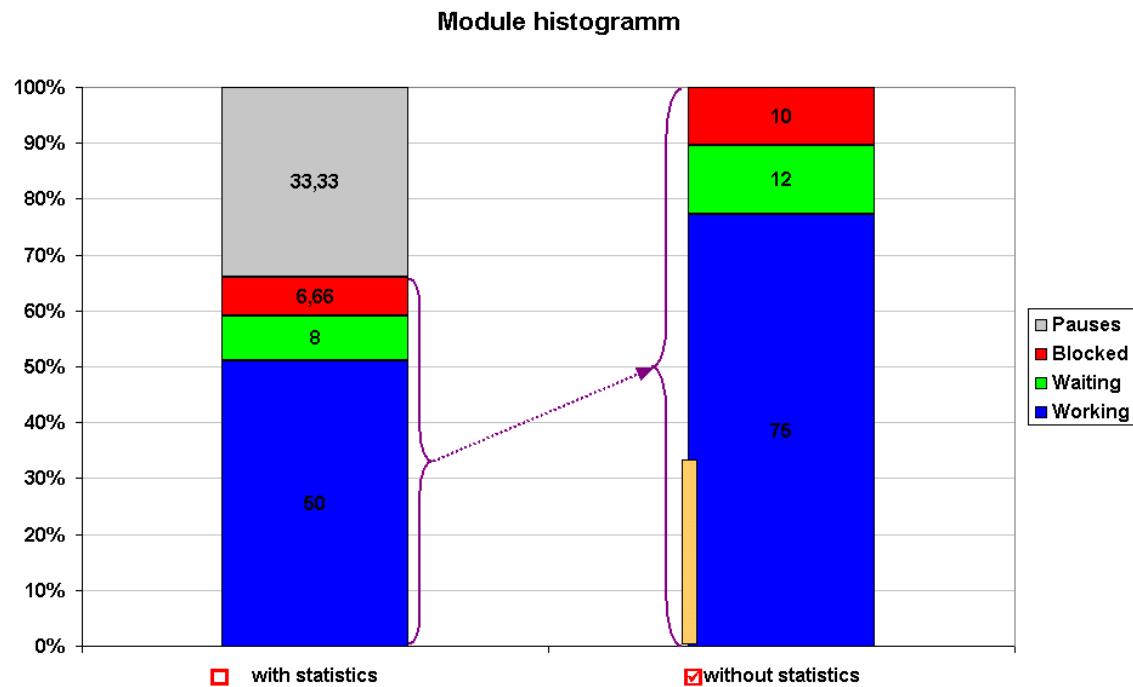


Figure 68.23 Conversion “with → without” statistic of break modules



68.10 Disable Failures

You can disable either a single failure (break module) or all failures together global, without deleting any break module. This may help you validating the model during modeling phase.

68.10.1 Disable a single Break Module

Double click on the break module.

Select the check box named “Passive”.

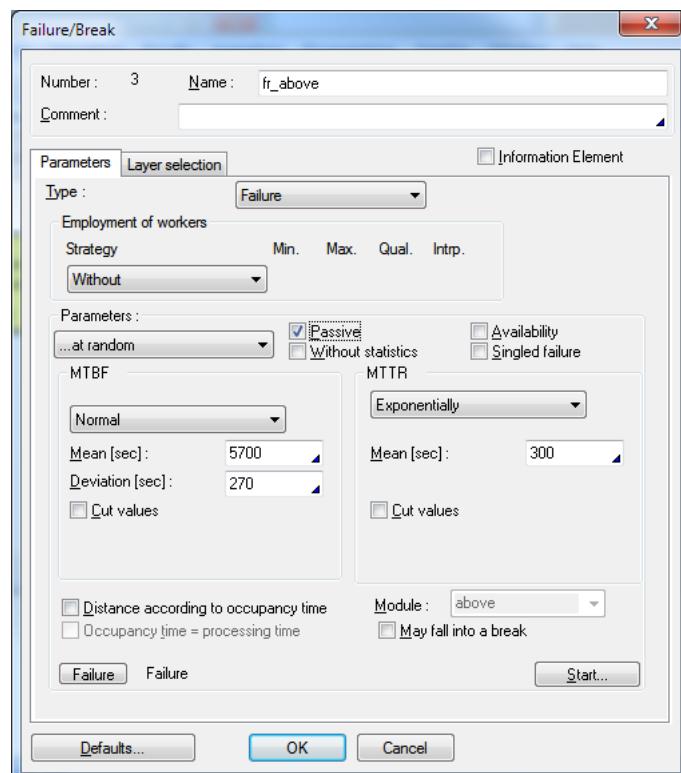


Figure 68.24 Disable failure “fr_above”



68.10.2 Disable all Break Modules global

Select “Simulation” → “Parameter” from the menu bar.

Select the check box named “**Disable failures**” and close the “Simulation parameter” window by click on its “OK” button.

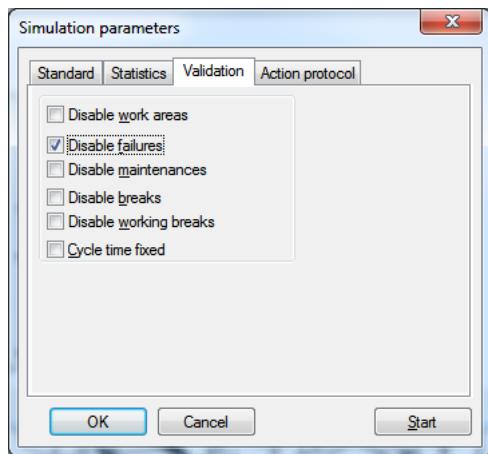


Figure 68.25: Disable all Failures global



69 Work Area

69.1 Theory

In order to achieve important optimization goals and results by using DOSIMIS-3, worker employment concepts, group work and multi-machine operation are applied with consideration of quality characteristics, task assignments and shift models. Additionally “work areas” allow a detailed analysis of a kanban controlled job shop production resp. consignment or assembly systems.

Applying work areas the user is able to define tasks that are accomplished in the real system by workers.. This may concern manual treatment of objects in work stations, maintenance work, fault clearance, set-up activity and other general tasks. Recently worker employment can be analyzed exactly with help of work areas.

The special module named “**work area**” is used to reproduce worker in a simulation model. Even different skills and different numbers of workers can be regarded. If a module requests workers, they will be provided to the module by a “work area” if possible. Further the processing sequence can be given as a function of priorities. Also the distance time, that a worker needs to walk from one workstation to another, can be considered for all possible mating of work stations. The default of work breaks can be defined similarly to failures/pauses.



In order to reproduce workforce in a model, following premises have to be fulfilled:

- A module of type “work area” has to be placed inside.
- Every module that may request workers from this “work area” has to be linked (connected) to it.
- A module can be assigned to several “work areas”.
- A “work area” can be assigned to several modules.

69.2 Task

As continuation of the work on the optimized Tutorial model of the first part, a “work area” including one “worker” is to be reproduced. The worker is supposed to work on objects at work station “above” and to set-up work station “below”.

The distance time between regular work place (this is the place, where the worker waits, if no work is present) and the work places amounts to 30s in each case and the distance time between work places is about 10s. After object treatment at work station “above” the worker should wait for 6s, to afford a further object to enter the work station. Additionally a pause model is to be defined for the worker.

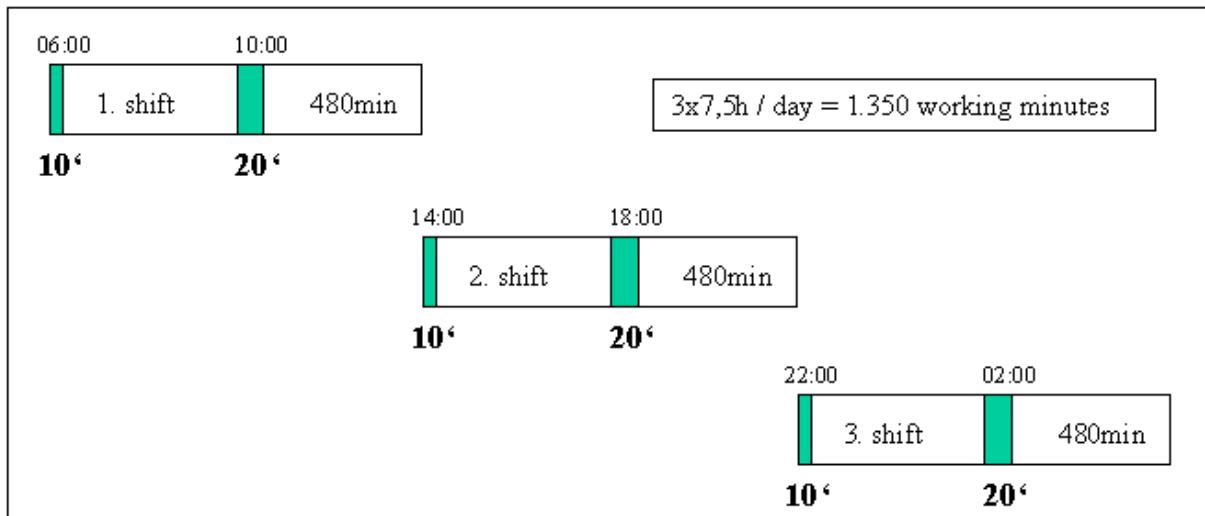


Figure 69.1: Pause model



Please create a “work area” according to following instructions.

- Please open the model **Tutorial2.mfs**. Note! This is not the final model of the previous chapter, but the basic model.
- Save this model using the new name “Tutorial2A.mfs“.
- Select “View“ → “Controls Palette” from the menu bar or press the “F2” button while pressing the “Ctrl” key on your keyboard..



- Select the “Work area” symbol on the “Controls Palette” and place it inside the opened model window by click on the left mouse button.
- Select “Model“ → “Linking active” from the menu bar or press the “F9” button on your keyboard to enable “linking mode”.
- At first the work area has to be selected by click with the left mouse button. The symbol color of the selected module turns to blue.
- Now select the work stations. These will turn their display color to red, what means that these stations are connected to the blue displayed work area.
- Disable “linking mode” by pressing the “F9” button again.



- Select the “Working place” symbol on the “Controls Palette”.
- Place the “Working place” (circle) close to work station “above” by click on the left mouse button.
- Select “Model“ → “Linking active” from the menu bar or press the “F9” button on your keyboard to enable “linking mode” again.
- Select the “Working place” by click on the left mouse button. It will be displayed in blue color then.
- Now click on work station “above”, then click on the “work area” while pressing the “SHIFT” button on your keyboard
- In the animation of the “Working place” only personnel from that work areas is animated, that are connected (linked) with the “Working place”.
- Disable “linking mode” by pressing the “F9” button again.



- Please insert another “Working place” close to work station “below” and connect it to this work station and to the work area as described above.

Often it is favorable to place all components first and to define the connections afterwards. The proceeding is left to the user.

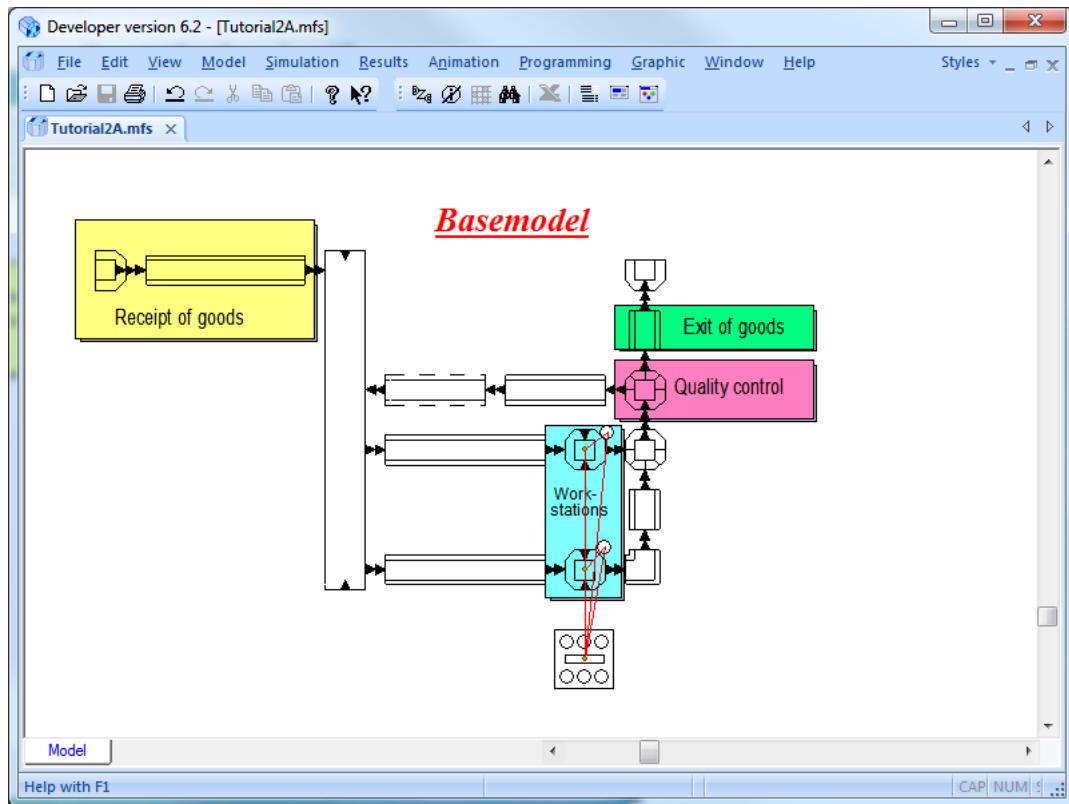


Figure 69.2: Links between modules and work area



- If a module was connected to a work area by mistake, so proceed please as follows to remove this link: Enable (if necessary) “Linking mode” by pressing the “F9” button, select the work area by click on the left mouse button (=> blue) and click on the appropriate module. Its color will change from red to black.
- Furthermore it is possible to assign as many worker positions (Working places) to a module as desired.

69.3 Parameter Setting of Work Area

Setting of parameters of workers consists of the setting of parameters of the modules at those the workers are to work and the setting of parameters of the work area. Settings of the work stations have to be changed in such a way that they request workers. So e.g. for object type 1 it has to be defined, that a worker is necessary for object treatment at work station “above”. This definition is done at the **select box “Strategy”** which is to be found in the area of **“Work procedures”** sub area **“Distribution of working time”, “Employment of workers”** on the **module parameter mask**. Work station “below” only requests a worker for set-up activities, so this definition is done in the “Set-up time” area of the module parameter mask.

69.3.1 Work Stations

Please double click on work station “above”.



- Use the **select box** named “**Strategy**”, which is to be found in the area “**Distribution of working time**” “**Employment of workers**” of the opened parameter mask, to select “**Maximal no. of workers**”. The “**Strategy**” specifies the relationship between the inscribed working time (processing time) and the number of workers. In case of “**Maximal number of workers**” this means, that the inscribed “**working time**” will be spent exactly, if the number of workers available at the work station is equal to the inscribed “**Maximal number of workers**”. If less worker are available, then the actual working time at the work station prolongs according to the ratio (**Maximal number of workers**):(**Actual number of workers**).
- There are four input boxes on the right side of the select box described above. Please enter the value “1” into the first three boxes and press the “TAB” key on your keyboard after each entry. The first and the second box define the minimal and the maximal amount of available workers. In the third box the requested level of qualification is to be assigned. Enter a “0” in the “**Intrp**” box (Interrupt). This means that work at this work station is not allowed to be interrupted. The entry in this field corresponds to the limit of the priority of a task. The activity of workers accomplished at this work station can be interrupted only if another work station needs that worker for a task, whose priority is higher than the “**Interrupt**” limit. Here lower numerical value corresponds to a higher priority („0“ = highest priority). Example: “**Interrupt at Station A = 3**” => only tasks with a priority value less or equal “2” (higher priority) may interrupt processing at Station A.
- Now repeat this two steps for work station “below”. Please regard that the worker at work station “below” is not requested for object treatment but for set-up activities. This entries are done in the “**Set-up times**” area.

Parameter mask of work station “above”

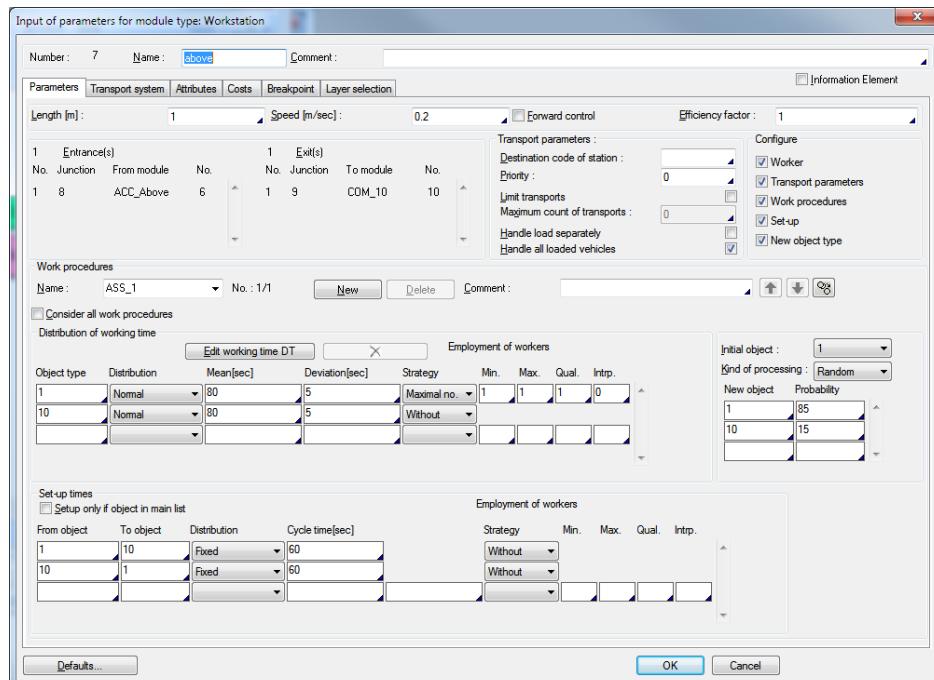


Figure 69.3: Parameter of work station “above”



Set-up parameter of workstation “below”.

Set-up times				Employment of workers				
From object	To object	Distribution	Cycle time[sec]	Strategy	Min.	Max.	Qual.	Intrp.
2	20	Fixed	60	Maximal no.	1	1	1	0
20	2	Fixed	60	Maximal no.	1	1	1	0

Figure 69.4: Set-up parameter of work station “below”

69.3.2 Work Area

Now the parameter settings at the work area can be done.. Please proceed as follows:

- Open the parameter mask by double click on the symbol of the work area module.
- Define “**List of workers**”: Click on the input box named “**Qualification**” and enter the value of “1” (“1” is the highest qualification level). Then press the “**TAB**” key on your keyboard and enter the “**Quantity**” of “1” (quantity, number or workers).
- “**Worker assignment**” and “**Task assignment**”: Default settings in both arrays can be accepted.
- “**Defined Tasks**”: The first input area contains the number of the task (“**No.**”, enter “1”). Then select “**process object**” from the select box named “**Kind**” and “**above**” from the select box “**Element**”. The priority (“**Prio.**”) specifies the preference of that task (here “2”). Please enter the following assignments in the second line: **No.** = “2”, **Kind** = “**set-up**”, **Element** = “**below**” and **Prio.** = “1”. Thus set-up task at work station “below” has higher priority and will be processed with preference.
- **Change of work place:** It is possible to select a strategy for changing the work station from the select box for each defined task. For the first task, at work station “above”, a delay of 6 seconds is to define. The worker will wait for this delay time for the arrival of a further object at that work station. So select “**when delay > n sec.**” from the **select box** and enter “6” in the input box named “**n:**”.

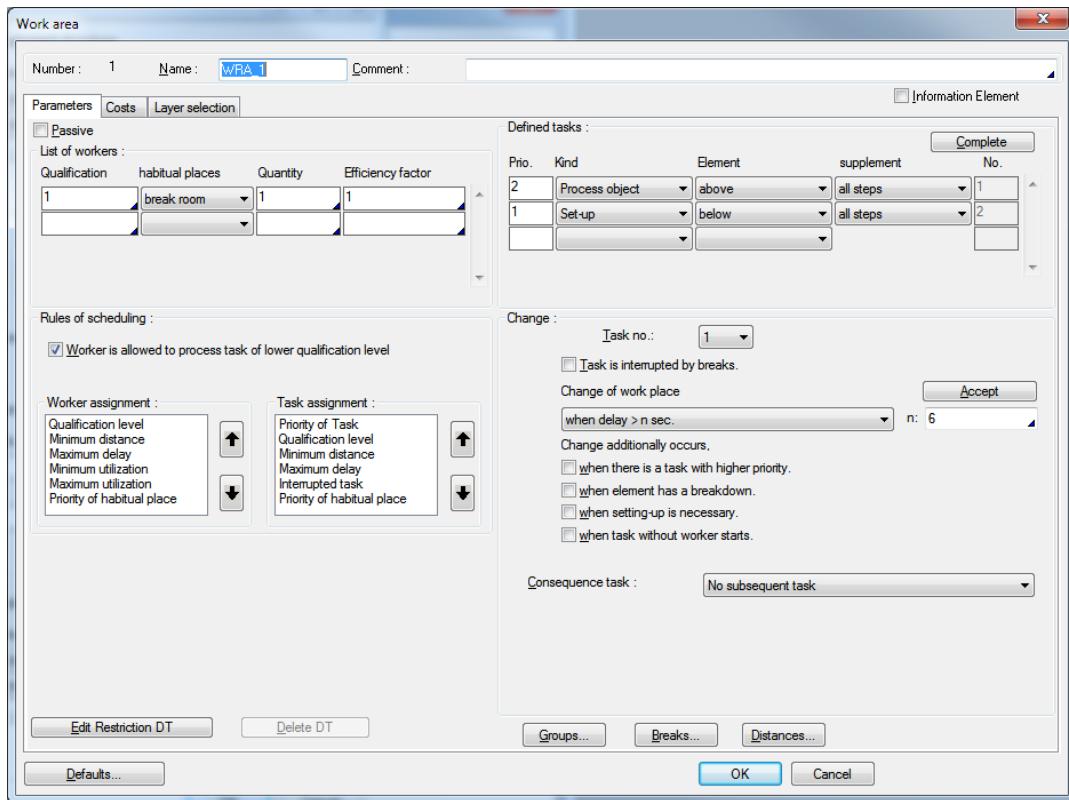


Figure 69.5: Parameter of the work area

- **Breaks:** Click on the button “**Breaks**” of the module parameter mask. Those in the following illustrated parameter mask appears. Click on the button “**New**”. Select the option “...periodically” from the select box inside the “Failure” area and enter “0” in the field named “**Start of 1st period**”. Enter “8:00” in the input box named “**Duration of period**” for a period of 8 hours. Now enter **two pauses according to the figure illustrated below** in the array below. Please regard to enter “**Begin**” and “**Duration**” of each pause during one period. This pauses are to be filtered out of statistic data, too. So please select the **check box “Without statistic”**. Click on the “**OK**” button after all of this is done.

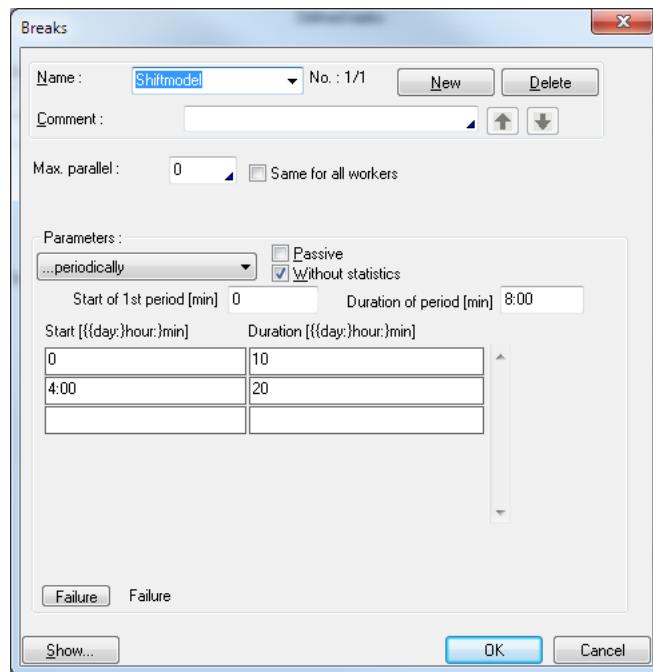


Figure 69.6: Pauses

- **Distances:** Please click on the button named “**Distances**”. The Work Area parameter mask disappears, the model appears again and the parameter mask named “**Path list**” is displayed in the foreground. Modules that are referred by the “Time” to set in the “Path list” are displayed in blue color in the model. If only one module is marked, then the referred path is the way between that module and the Work Area. Make sure that the accordant modules are selected. Please enter all “**Time**” values according to the figure below. When done, click on the “**OK**” button. The Work Area parameter mask will be displayed again.

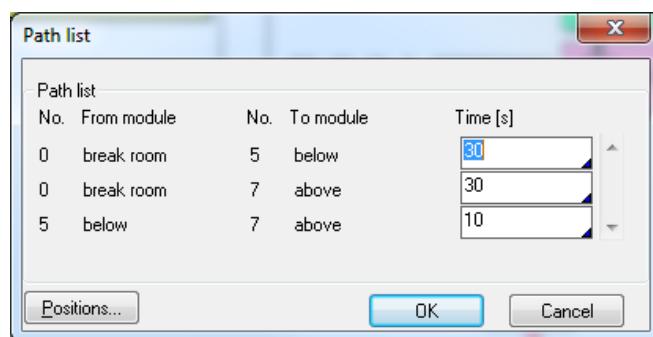


Figure 69.7: Path list

- Finish parameter setting by click on the “**OK**” button or press the “**Enter**” key on your keyboard.



Depending on the object type entering the work station, a worker has to be requested or not. If no worker is available at this point of time, the task is marked as “**undone**” and the module state turns to “**waiting**” (the object is displayed in yellow color). In DOSIMIS-3 there are several module types that support manual tasks: e.g. “**Work Station**”, “**Assembly**” and “**Disassembly**”. This kind of tasks are defined by the generic term “Object Treatment”.



Object treatment depends on the type of object, that means that a worker does not have to be available all the time for object treatment at a work station. Additionally worker employment strategies can be defined to control the change of tasks during object treatment.



The activities of workers are displayed in the animation. If this additional animation is not desired, please select “Simulation” → “Parameter” from the menu bar and **deselect** the check box named “Worker trace”.

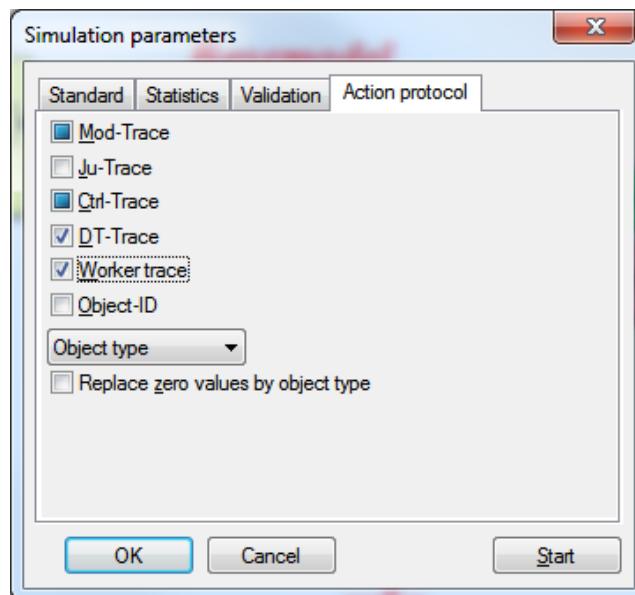


Figure 69.8: Disable of “Worker trace”.

Changing of this simulation parameter only effects on animation. There is no effect on the procedure of worker arrangement.

69.4 Analysis of Work Area

After simulation run has finished you can start animation mode and watch the model. Please select “**Single step**” by click on the appropriate button of the “**animation bar**”. Worker #1 is displayed in red color inside the work area. That means he is at his regular work place and has a break (the worker is paused). The break lasts 10 minutes. The first object enters work station “above” after about 2 minutes of simulation time. As the object has entered the work station completely its color turns to yellow and displays the state of “waiting for worker”. After 10 minutes of simulation time the worker leaves his work area and moves to work station “above”. Thus the worker position (Working place displayed as cycle) of work station “above” changes its filling color to green. As soon as the worker arrives at the work station and starts to work, the “working place” is displayed in blue color. **Note that contemporaneous events are animated successively.** That is why the object in work station “above” will be displayed in blur color (working) in the next step. However both events take place at the same time, what you can watch at the “digital display” of simulation time. After the object has left work station “above”, its “working place” appears in yellow color. This is because of the fact that the worker still waits maximally 6 seconds for the next object to enter the work station completely.

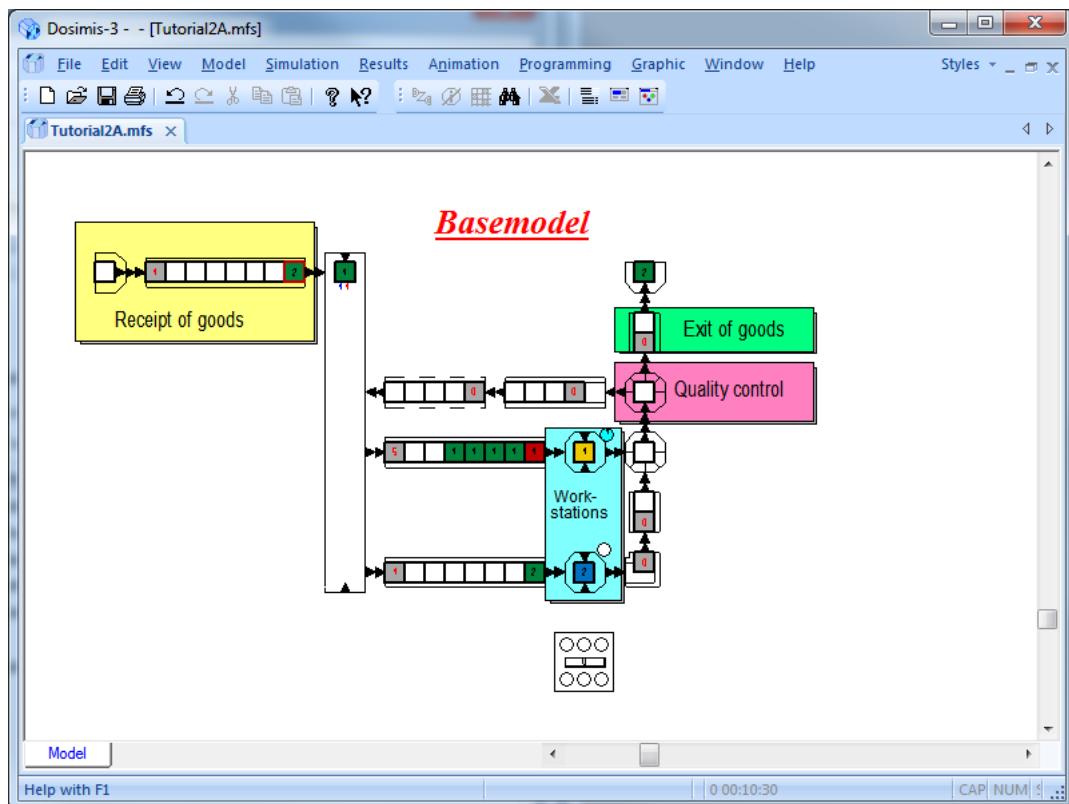


Figure 69.9: Object treatment at work station “above”

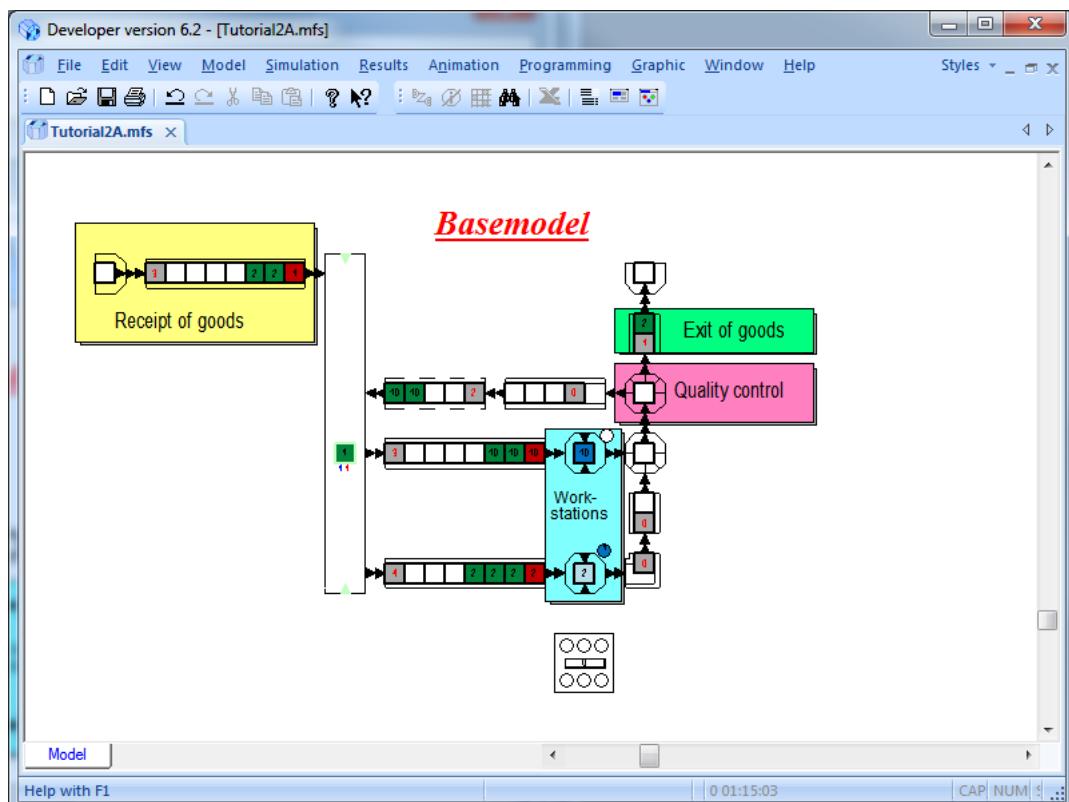


Figure 69.10: Set-up activity at work station “below”



The color of an object which is shown as a rectangle displays the status of this object.. Thus applies particularly with worker employment:

- Yellow* = waiting for worker
- Magenta* = waiting for Set-up worker

69.5 Statistic

After simulation run has finished it is possible to display the results of the work area in form of bar charts.

- Please select the work area by mouse click on its module.
- Then select “Results“ → “Result Parameter” from the menu bar.
- Mark the **check box** named “Statistic Point in Time” which is to be found on the “Selection” tab. The final statistic at of the point in time of 600 minutes is to be displayed.
- Click on the “Exit” button to accept your changes and to close the “Result Parameter” mask.
- Now select “Results“ → “Work Area Statistics” from the menu bar.

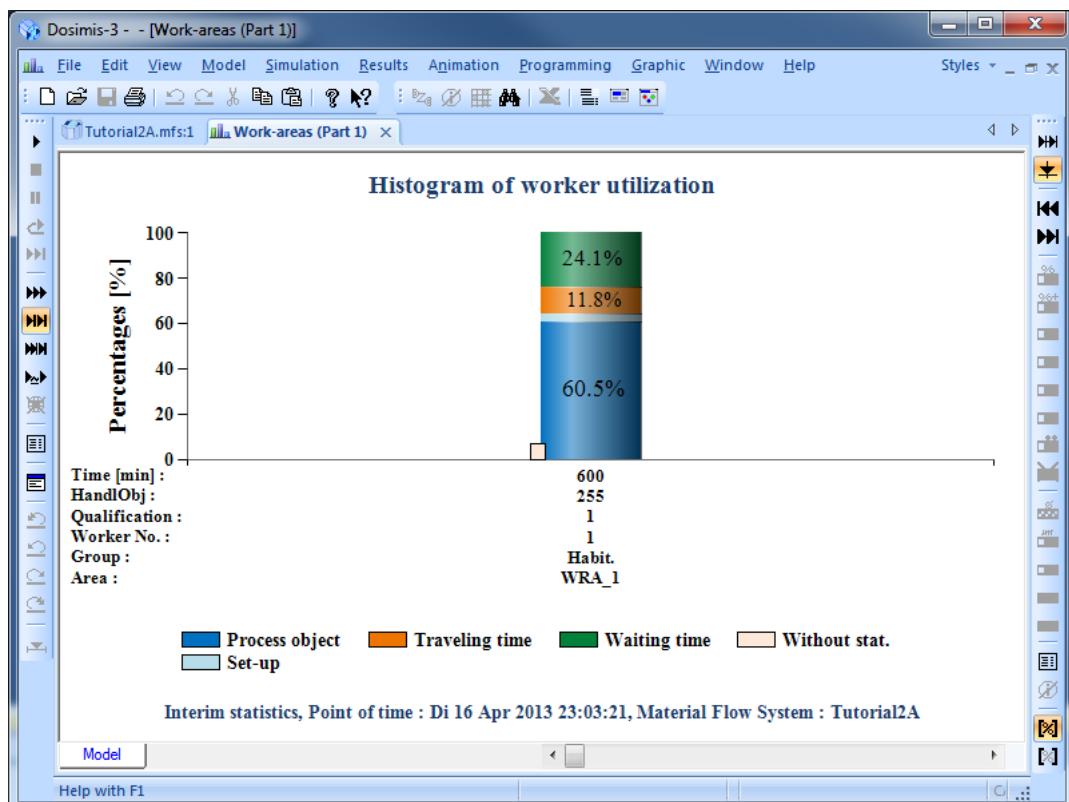


Figure 69.11: Final statistic of work area

In this case worker utilization shows a quite high reserve (Waiting Time). This seems to be sufficient. However a look on the statistics of the work stations leads to another interpretation.

- Please select both work stations for this.
- Then select “Results“ → “Time Histogram” from the menu bar.

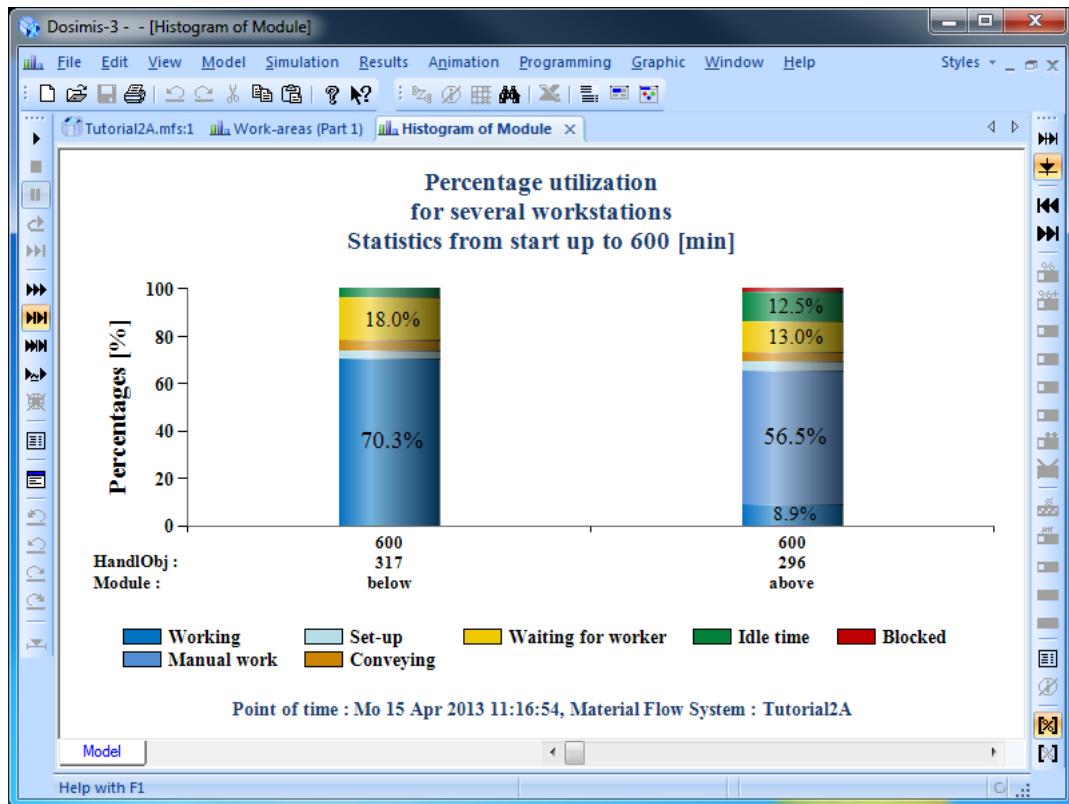


Figure 69.12: Final statistic of both work stations

There are high portions of “**waiting for worker**” on both work stations. This can be verified in the final statistics, too. Similar values appear in the statistics of all other modules. These results show that worker and machines thwart each other by “waiting for each other” and so system performance is to be reduced. Thus throughput at the sink is approx. 16% lower in this simulation run than in a simulation run without worker.

Waiting times of the worker result from:

- the time a worker has to wait until the next object has entered work station “above” (5s travel time, 6s maximal waiting time).
- the inactivity of the worker during rework at work station “above”.
- and waiting for objects.

Waiting times of the machines for the worker result from:

- pause times of the worker,
- the occupation of the worker by the other work station in each case and
- in case of work station “below” due to the fact, that the worker will not be released from work station “above” until there is a change of object type (from 1 to 10) or the buffer before is empty.

Please select “**Results**“ → “**Statistic Data**” from the menu bar to display the final statistics.

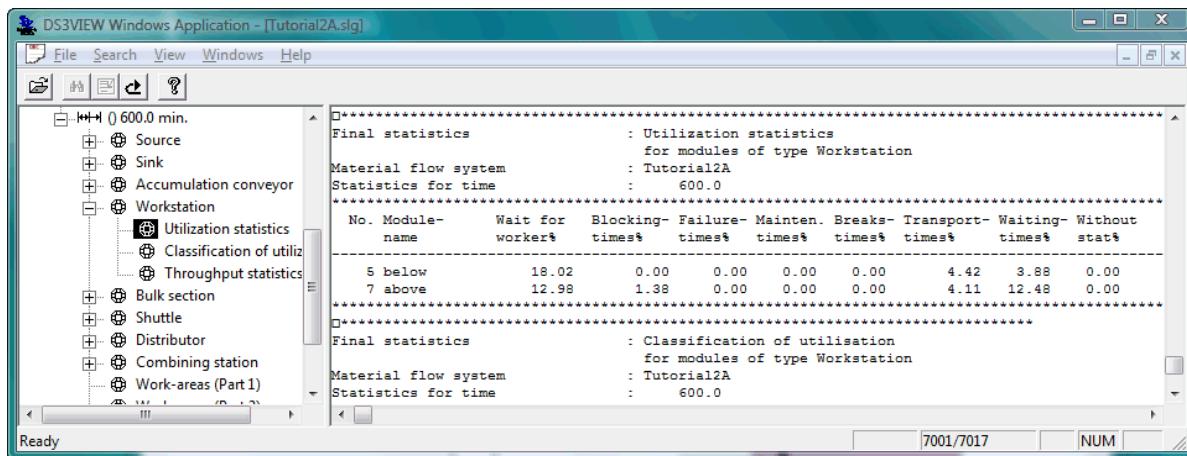


Figure 69.13: Final statistic (utilization of work stations)

The statistics of “work areas” is to be found below statistics of modules.

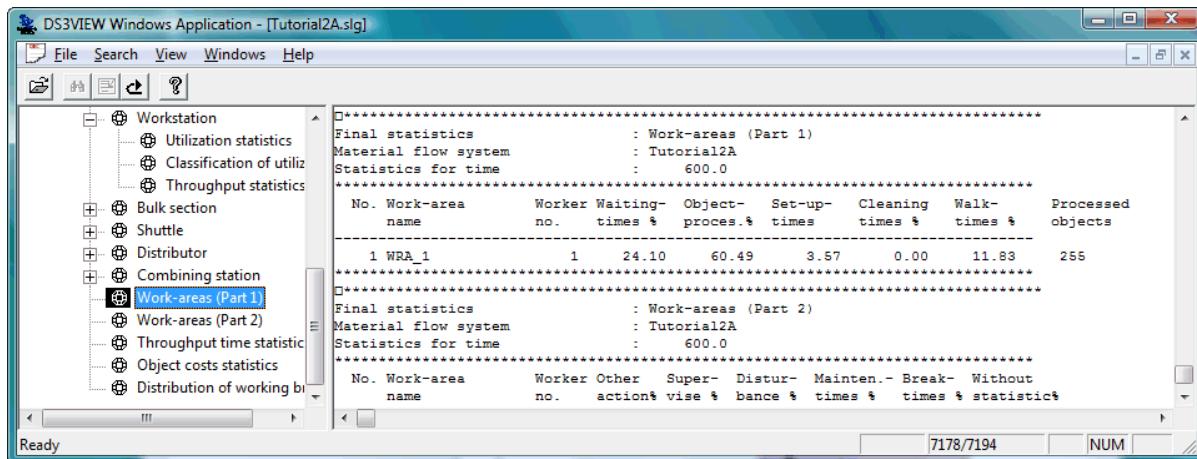


Figure 69.14: Final statistic (work area)

69.6 Several Worker per Task

With extensive activities often it is necessary that several workers have to work at the same time on one task. This is supported by the worker employment strategy of DOSIMIS-3, because it is possible to assign a minimal and maximal number of workers to each task. The work time that is set always refers to the registered maximal number of workers. Work start when the minimal amount of workers are available. If the values of minimal and maximal number of workers are different, then the process time will increase inversely proportional to the number of active workers. So, if the maximal number of workers needed is three and only two workers are available, process time will be prolonged by 50% (“maximal number of workers”/“actually number of workers” * work time = $3/2 * \text{work time}$).

Now for each object treatment one worker is needed in the example model. For set-up activities however two workers are necessary for each work station.

Manning level necessary for one task can be entered in the input boxes of “Employment of workers” Strategy in the parameter mask of the work stations.



Set-up times				Employment of workers				
From object	To object	Distribution	Cycle time[sec]	Strategy	Min.	Max.	Qual.	Intrp.
2	20	Fixed	60	Maximal no.	2	2	1	0
20	2	Fixed	60	Maximal no.	2	2	1	0

Figure 69.15: Several workers per task

Please extend worker employment strategy of both work stations in such a way, that each object treatment will request one worker and each set-up activity will request two workers. If all workers work according to the same shift model, the number of workers could be increased to two in the existing work area. But if the shift models (pause times) of the two workers are different, a second work area has to be included into the model. After the existing work area has been copied and the “Working places” have been placed, the new model looks like following:

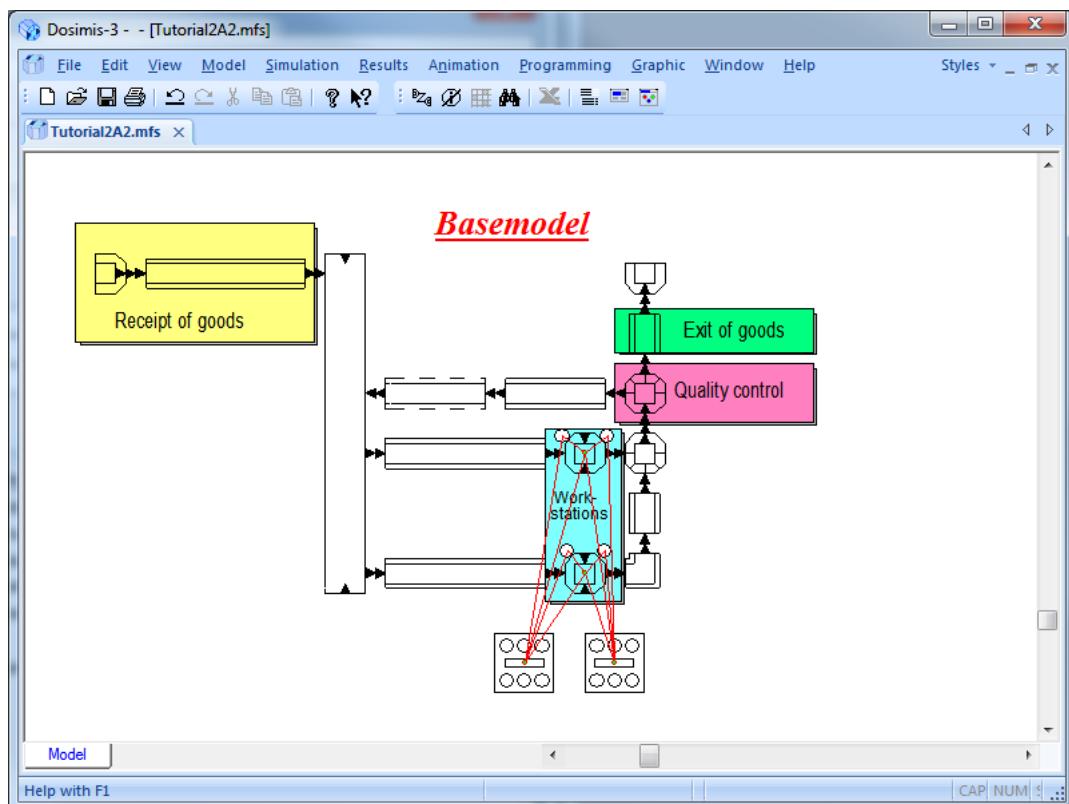


Figure 69.16: Relations of elements in the model

To display the connections between the elements please select “Model“ → “Info...“ → “Connections” from the menu bar. The two work stations are connected to the two work areas in each case. And one “Working place” at each work station is assigned to one “work area”.

Select “Simulation“ → “Check Output” to see the results of “Consistency Check”. Some error messages are displayed. E.g. no worker is assigned for object treatment at work station “below”. Only if these errors are removed, the model can be simulated.



```

consistency check: Tutorial2A2

For working at object (2)
instead of minimal (1) are only (0) qualified (qualification 1) worker defined

Workstation:object list (2) contains error

For working at object (20)
instead of minimal (1) are only (0) qualified (qualification 1) worker defined

Workstation:object list (20) contains error

Error : Distribution of working time
For working at object (2)
instead of minimal (2) are only (1) qualified (qualification 1) worker defined

Workstation:object list (2) contains error

For Setting-up at object (2) to (20)
instead of minimal (2) are only (1) qualified (qualification 1) worker defined

For working at object (20)
instead of minimal (2) are only (1) qualified (qualification 1) worker defined

Workstation:object list (20) contains error

For Setting-up at object (20) to (2)
instead of minimal (2) are only (1) qualified (qualification 1) worker defined

Error : Set-up times
Error in work procedures: 1/ASS_1
Error in Workstation: 5/below
< >
Ready

```

1/139 1/30 NUM

Figure 69.17: Check Output of Consistency Check

For that purpose please proceed as follows:

Supplement in the parameter mask of work area 1 the task:

No	Kind	Element	Supplement	Prio
3	set-up	above	all steps	1

Change and supplement in the parameter mask of work area 2 the tasks:

No	Kind	Element	Supplement	Prio
1	process object	below	all steps	2
3	set-up	above	all steps	1

Thus each worker accomplishes the treatment of objects at his work station, if however set-up activities are requested, both workers will cooperate for this purpose. This is because of the fact that all workers are disposed internally independently from an affiliation to a “work area”. The criterion is that a worker **may** work on the demanded activity.

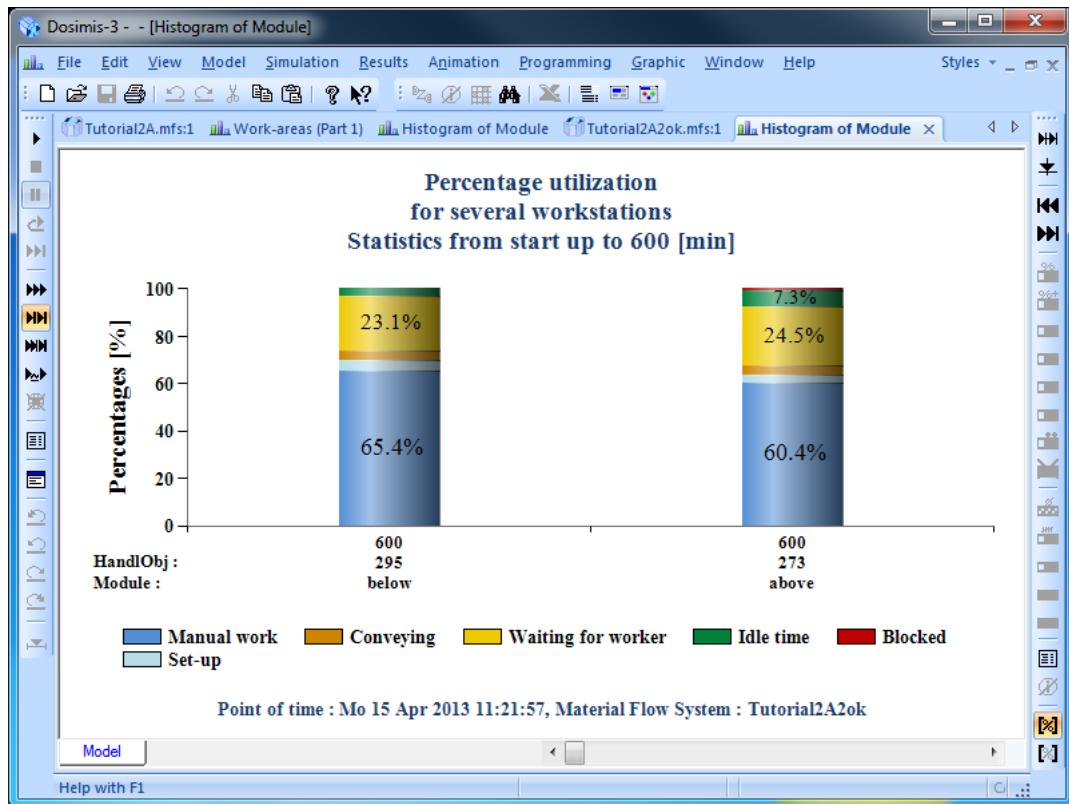


Figure 69.18: Final statistics of both work areas

The “Time Histogram” of both work stations shows a large portion of “**waiting for worker**”. An analysis shows that the work stations wait for workers quite frequently for set-up activities, since workers terminate their treatment of objects first, before a change of working place is possible.

69.7 Interrupt of Tasks

Set-up activities have already highest priority. Since however always two workers are needed, it is always necessary to wait for the worker from the in each case other work station. Now, in case of a set-up request from the other work station, the worker shall interrupt object treatment at his work station and shall change to the other station at once. For this purpose the value of 10 is to be set into the “**Interrupt (Intrp)**” field of the work station parameter masks as shown in the figure below. This means that tasks with a priority higher than 10 (priority 9 is higher than priority 10) may interrupt this task. This parameter change is to be done for both work stations.

Note: The remained work time of the interrupted work is provided again for disposition purposes. However the released worker will not serve it, since a task with higher priority exists. Because only such tasks lead to an interruption.



Distribution of working time					Employment of workers				
Object type	Distribution	Mean[sec]	Deviation[sec]	Strategy	Min.	Max.	Qual.	Intrp.	
1	Normal	80	5	Maximal no.	1	1	1	10	
10	Normal	80	5	Maximal no.	1	1	1	10	

Figure 69.19: Interrupt of task

A look at the result diagram (Time Histogram) shows the desired effect. The quota of “waiting for worker” could be reduced significantly. A further reduction is no longer possible due to pause times of the workers.

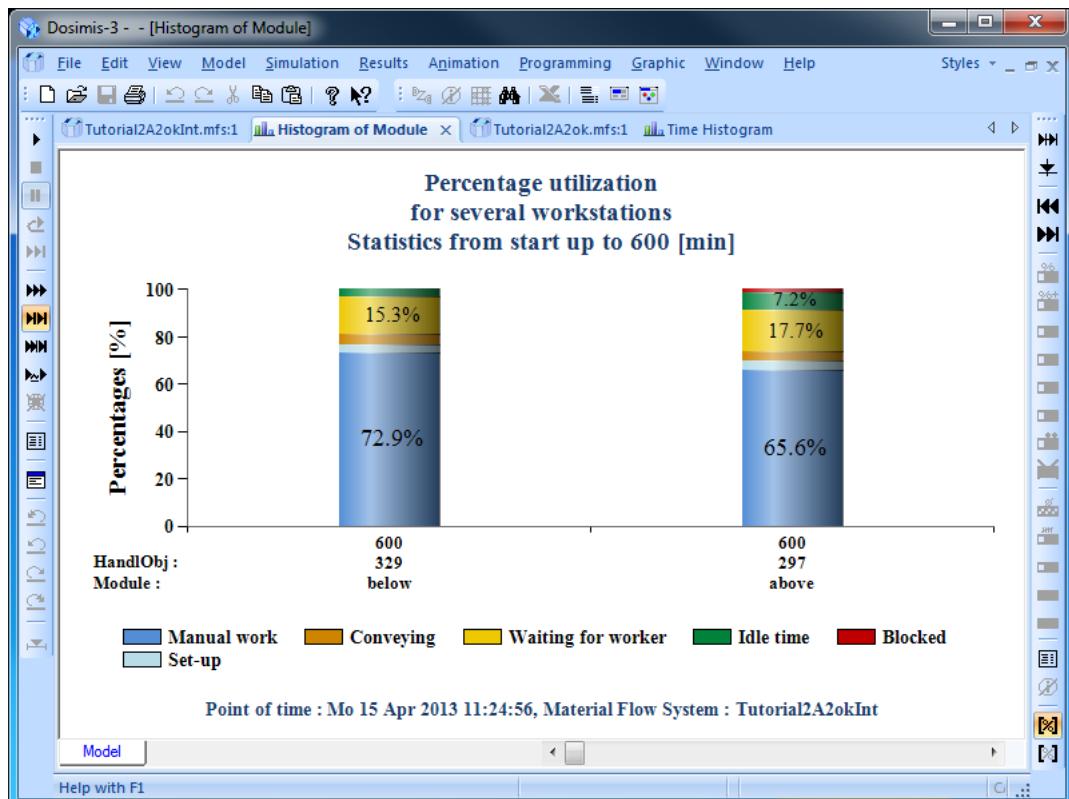


Figure 69.20: Final statistics of work stations



69.8 Passivate Work Areas

Scenario: What, if there were enough workers for each activity always, that means, if there were no “waiting for worker” times? All manual activities are accomplished automatically in this case, so, as if the maximum number of workers is available, the pauses of workers do not overlap and no distance times accrue.

Passivate global

- Please select “**Simulation**” → “**Parameter**” from the menu bar.
- Select the check box named “**Disable work areas**”.

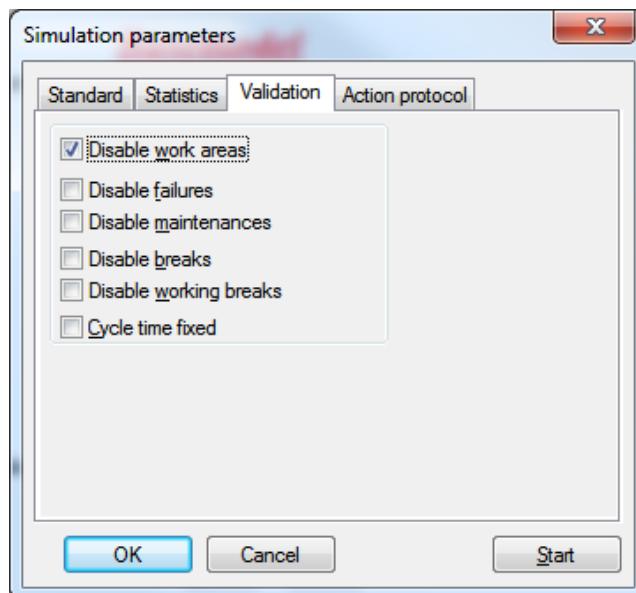


Figure 69.21: Passivate all work areas global





70 Index

- 3—
- 3D toolbar 26-5
 - 3D-Animation 26-10
 - animation speed 26-12
 - cameras 26-13
 - keyboard control of animation 26-11
 - object type assignment 26-11
 - parameter 26-11
 - 3D-Objects
 - move 26-5
 - rotate 26-5
 - scaling 26-5
 - static graphics 26-5
 - 3D-Visualization
 - 2D- und 3D-models 26-8
 - 3D toolbar 26-5
 - camera 26-12
 - graphics exchange 26-7
 - Installation 26-1
 - introduction 26-1
 - navigation 26-3
 - open view 26-1
 - operations in the 3D-view 26-4
 - parameter 26-9
 - selection of 3D-objects 26-4
 - system requirement 26-1
- A—
- Abort statistic 11-11
 - Accumulation conveyor 4-10, 62-3
 - initialization 4-10
 - parameter 63-9
 - Accumulation conveyor2 4-12
 - initialization 4-13
 - object length 4-14
 - sensor 4-13
 - Action 16-3
 - deactivating 16-8
 - Activating decision table 16-12
 - Actual contents 11-12
 - Adding
 - line 16-4
 - Additional statistic
 - assembly station 11-14
 - disassembly station 11-14
 - loading station 11-14
 - module cost 11-18
 - object cost 11-19
 - paired shuttle 11-15
 - select list 2-57
 - shuttle 11-15
 - throughput time 11-17
 - turning table 11-15
 - unloading station 11-14
 - worker cost 11-19
 - workstation 11-14
 - Adjusting graphic 12-16
 - Algorithm
 - optimization 10-16
 - Allocation proportion 11-36
 - compact 11-37
 - Alternating
 - distribution strategy 3-13
 - Alternating distribution 3-16
 - Alternative assembly list 4-35
 - Alternative stations
 - shop floor control 24-4
 - Analysis
 - failures 68-7
 - work area 69-8
 - Animation 12-1, 21-1, 63-20
 - backward 12-12
 - breakpoint 12-13
 - control 12-11
 - decision table 21-1
 - forward 12-11
 - hide 12-11
 - kind of animation 12-11
 - log window 12-11
 - parameter 12-2, 12-11
 - save state 12-12
 - states 12-6
 - toolbar 12-10
 - view objects 12-14
 - Animation text 17-1
 - operations 18-34
 - Application program interface
 - Functions 33-1
 - Arrange
 - background 25-3
 - foreground 25-3
 - one layer back 25-3
 - one layer front 25-3
 - reorder 25-3
 - Assembly
 - shop floor control 24-5
 - Assembly entrance 4-34
 - Assembly station 4-33
 - additional statistic 11-14
 - alternative assembly list 4-35
 - assembly entrance 4-34
 - functionality 4-36
 - kind of assembly 4-35
 - transport entrance 4-34
 - waiting for assembly 4-34
 - Assign layer 2-43
 - Assignment of worker 7-1
 - Attribute 3-40, 16-16
 - Attributes from file 4-8
 - Auto position
 - syntax editor 16-6
 - Automatic
 - path finding 3-18
 - routing 5-17
 - Automatic generating 27-3



Average contents 11-12
—B—
 Base point
 shuttle 4-63
 Basic elements 18-5
 decision tables 18-10
 mathematical function 18-9
 random function 18-8
 selectors 18-5
 text output 18-11
 transport system 18-12
 variables 18-6
 Basic position
 shuttle 4-61
 turning table 4-54
 Battery charge 5-27
 Battery charging possibility 5-19
 Battery charging station 5-19
 Battery control 5-18
 Battery parameter 5-18
 Battery run
 needed 5-27
 scheduling 5-27
 Bauschuld 3-9
 Bauschuld distribution 3-15
 Bin location 4-24
 Bitmap 25-13
 parameter 25-13
 Bitmap animation
 adjusting graphic 12-16
 animation run 12-18
 export 12-19
 introduction 12-15
 object number 12-17
 preparation 12-15
 printing 12-19
 start 12-18
 Blocking time 5-27, 11-13
 Break 8-1
 Break time 11-13
 Breakpoint 3-44, 12-13
 rule 16-10
 Breaks 69-6
 work area 69-6
 Bulk section 4-16
 Button 2-2
—C—
 Call
 decision table functions 35-1
 Call stack 16-20
 Camera 26-12
 Cameras
 3D-Animation 26-13
 Cancel
 input of a line 16-5
 Capacity monitoring 8-11
 result graphic
 minimum maximum 8-14
 occupancy 8-15
 result table
 Change
 rule 16-9
 Change of object type 4-30
 Charging station 5-18
 Check output 10-1
 Circle 25-9
 parameter 25-9
 Clean 2-24
 Cleaning 4-41
 Clipboard
 copy 2-37
 cut 2-37
 diagram 11-44
 graphic 2-23
 layout result 11-21
 paste 2-38
 Close 2-22
 Collective 3-45
 COM Server
 data types 28-1
 IDs3Application 28-3
 IDs3Control 28-4
 IDs3Document 28-4
 IDs3Element 28-8
 IDs3Evaluation 28-9
 IDs3Failure 28-9
 IDs3Junction 28-10
 IDs3Workare 28-11
 introduction 28-1
 methods 28-2
 VBA 28-12
 VBA Example 28-15, 28-16
 Visual C++ 28-17
 Visual C++ Example 28-20
 Combining station 4-46, 62-4
 parameter 63-16
 Comments 3-36
 Common words and terms of C/C++ 29-2
 Common words and terms of Visual-C++ 29-2
 Compact representation 11-40
 Compile 30-1
 Completion
 task list 7-7
 Condition 16-3
 Configuration 4-31, 29-4
 workstation 4-31
 Connecting to VBA
 COM Server 28-12
 Connecting to Visual C++ 28-17
 Connection set-up 27-1
 Connector 8-17
 Consistency 2-13, 8-32
 Consistency check 10-1, 33-9
 program interface for decision tables 35-2
 show errors See Check output
 Consistency check at transport system 5-27
 Context menu 2-12
 Context sensitive popup 16-7
 Context sensitive selection 16-7
 Continuous 12-19
 Continuous animation
 parameter 12-20



Control bar 2-1
 Control palette 2-45
 Conveying circuit 4-58
 null position 4-58
 positions 4-58
 Conveying section 4-15
 Coordinates 2-49
 Copy
 decision table 16-14
 elements 2-38
 group of elements 2-39
 rule 16-9
 Copy to clipboard 2-37
 Cost 3-41
 diagram 11-26
 storage 4-26
 Cost measurement 11-30
 diagram (single) 11-31
 Create
 decision table 16-13
 rule 16-9
 Crossing 4-49
 circuit time 4-52
 lift time 4-51
 Cut 2-37
 Cycle analysis 11-38
 Cycle time analysis
 compact representation 11-40
 density function 11-41
 single measuring 11-39
 cycled See fixed
 —D—
 Data input
 modules 34-7
 Data transfer 27-2
 Data types
 COM Server 28-1
 Deactivating
 action 16-8
 Deadlock 3-45, 5-20
 Deadlock control 5-20
 Debugging
 simulator 31-2
 user interface 31-2
 Visual-C++ (Version 2005) 31-7
 Visual-C++ (Version 2008) 31-4
 Visual-C++ (Version 2010) 31-4
 Visual-Studio debugger 31-9
 Decision
 distribution 3-11
 right-of-way 3-4
 Decision table
 activating module 16-19
 activating timer 16-19
 animation 21-1
 animation text 17-1
 attribute 16-16
 call stack 16-20
 copy 16-14
 create 16-13
 delete 16-14
 drag & drop 16-15
 float attribute 16-16
 further components 17-1
 global variables 16-16
 initial variables 16-16
 integer attribute 16-16
 introduction 14-1
 monitoring 17-4
 move 16-14
 net attributes *See* Decision table: global
 variables
 online simulation 22-1
 parameter 16-16
 progress indicator 17-7
 quicktable 17-2
 rename 16-14
 signal display 17-6
 string attribute 16-16
 timer 17-5
 Declaration
 program interface for decision tables 35-1
 Default rule 16-3
 Defaults 3-32
 Defined failure 8-4
 Delete 2-39
 decision table 16-14
 line 16-8
 references 16-19
 rule 16-9
 Deliver time 5-27
 Delivery destination 5-17
 Delivery strategy 4-23
 FIFO 4-23
 max. absolute occupancy 4-23
 max. relative occupancy 4-23
 priority of areas 4-23
 random 4-23
 Density function
 cycle time analysis 11-41
 destination 4-5
 Destination code
 loading station 5-5
 unloading station 5-7
 Destination parameter 3-15
 Detailing
 storage 4-21
 Detailing 4-21
 Dialogs 33-14
 Dictionary 36-1
 Digital display 10-9, 12-1
 Directory structure
 files of the program interface for controls 31-13
 project directory 31-12
 workspace directory 31-13
 Disable
 failure 68-19
 Disassembly
 shop floor control 24-6
 Disassembly station 4-37
 additional statistic 11-14
 disassembly exit 4-37



functionality 4-39
 transport exit 4-37
Discharger 4-43
 discharging direction 4-43
 discharging exit 4-43
 main conveying direction 4-43
 main conveying exit 4-43
Distribution function 3-19
 erlang distributed 3-25
 exponential distributed 3-24
 fixed 3-19
 histogram distributed 3-26
 normal distributed 3-22
 triangular distribution 3-21
 uniformly distributed 3-20
Distribution of random time 11-19
Distribution strategy 3-11
 alternating 3-13
 automatic path finding 3-18
 bauschuld 3-15
 destination parameter 3-15
 maximum free capacity 3-13
 minimum absolute occupancy 3-12
 minimum relative occupancy 3-12
 percentage 3-14
 priority of double cycle 4-70
 priority of exits 3-14
 self-defined 3-17
 shortest route 3-17
Distributor 4-42, 62-4
 parameter 63-17
Docking station 5-20
Dongle 37-2
DOSIMIS-3
 introduction 1-1
Dosimis-3 data structures
 debugging the implementation 31-11
 usage 31-11
Draw
 bitmap 25-13
 circle 25-9
 line 25-7
 polygon 25-10
 polyline 25-10
 rectangle 25-8
 reference 25-11
 text 25-12
Driving time 5-17, 11-15
Ds3Library.ini 31-13
DT-Script
 keywords 18-1
Duplicate
 rule 16-9
Duration of tasks 7-2
DXF
 export 25-3
 import 25-4
—E—
Edit
 assign layer 2-43
 copy to clipboard 2-37
 cut 2-37
 delete 2-39
 execute 2-40
 freeze 2-43
 hide 2-43
 line 16-4
 mirror 2-41
 move 2-41
 next selection 2-41
 parameter 2-40
 paste 2-38
 previous selection 2-40
 redo 2-37
 rotate 2-41
 rotate (right) 2-42
 select all 2-40
 snap individual 2-42
 snap parameter 2-42
 snap relative 2-42
 thaw 2-43
 undo 2-36
 unhide 2-43
Editing of parameters
 Failure 68-4
Editor
 decision table 16-1
 program interface 30-1
 shop floor control 24-2
Efficiency factor 3-40
 storage 4-21
Element
 animation text 17-1
 capacity monitoring 8-11
 changing the position 2-35
 collective parameterization 3-45
 connector 8-17
 copy 2-38
 copy (group) 2-39
 display element no. 2-53
 evaluation 8-32
 failure/maintenance/break 8-1
 length of 0 2-20
 linking 2-16
 measuring element 8-24
 monitoring 8-31, 17-4
 orientation 2-13
 polygons 9-1
 positioning 2-13
 progress indicator 17-7
 quicktable 17-2
 search 2-51
 show parameters 2-53
 shuttle control 8-18
 signal display 17-6
 storage control 8-19
 text 9-1
 throughput time measurement 8-20
 timer 17-5
Empty run
 shuttle 11-16
 with order 5-27



without order 5-27
End work
 Failure 8-7
Ending data transfer 27-3
Energy monitoring 5-18
English - German Dictionary 36-1
Entering
 Failure 8-7
Erlang distributed 3-25
EtFunction
 decision table functions 35-1
Evaluation 8-32
Events 34-1
Exact location
 storage 4-21
Exact locations 4-24
Example 27-9
 abstract 33-20
 data structure 33-16
 dialog 33-17
 excel data transfer 27-9
 initialization 33-17
 processing events 33-18
 program interface for controls 33-4, 33-16
 simulator functions 33-17
 user interface functions 33-17
VBA COM Server 28-15
VBA Model Creation 28-16
Visual C++ COM Server 28-20
Exchange
 references 16-19
Execute 2-40
Exercise 62-1, 68-1
Exiting
 Failure 8-7
Experiments 64-1
Exponential distributed 3-24
Export 2-23, 27-3
 DXF-format 25-3
 DXG-format 25-3
—F—
Failure 8-1
 defined failure 8-4
 dependent on throughput 8-7
 disable 68-19
 end work 8-7
 entering 8-7
 exiting 8-7
 main direction 8-7
 overlay 8-9
 parameter 68-4
 periodical failure 8-6
 random failure 8-5
 reference failure 8-8
 secondary direction 8-7
 start work 8-7
 statistics 68-10
 statistics of elements 8-9
 theory 68-1
Failure time 5-27, 11-13
Failures
 analysis 68-7
Faults
 task 68-2
FIFO 3-5
File
 *.mfs-file 13-4
 *.tra 13-2
 .apl 13-1
 .aws 5-28, 13-1
 .chk 13-1
 .dar 13-1
 .dxg 13-1
 .err 13-1
 .log 5-28, 13-1
 .mfs 13-1
 .mtx 5-28, 13-1
 .mtx.bin 13-1
 .pty 13-1
 .slg 11-11, 13-1
 .tra 13-1
 close 2-22
 include 2-22
 insert 25-4
 open 2-22
 operations 18-34
 save 2-22
 types of files 13-1
File formats 24-12
Filter time to repair out of statistic data 68-16
Final decision table 16-13
Final statistic 11-11, 11-14
Finish
 input of a line 16-5
Float attribute 16-16
Forcing of doubles cycles 5-20
Format
 export DXF 25-3
 export DXG 25-3
 import DXF 25-4
 import DXG 25-4
Forward control 3-29
Freeze 2-43, 25-5
Fully automatic mode 5-12
Functionality
 assembly station 4-36
 disassembly station 4-39
 varying distribution function 3-28
—G—
Generating from file 4-5
Generating of objects 4-2, 4-3
Global decision table 16-12
Global strategy
 paired shuttle 4-67
Global variables 16-16
Graphic
 insert file 25-4
Graphic menu 25-1
 background 25-3
 delete 25-3
 delete unused groups 25-3
 delete unused symbols 25-3



export/output-DXF (DXG) 25-3
 foreground 25-3
 freeze 25-5
 group 25-2
 hide all graphic 25-5
 import/read-DXF (DXG) 25-4
 one layer back 25-3
 One layer front 25-3
 select 25-4
 ungroup 25-2
 view group 25-3
 view symbols 25-3
Graphic palette
 bitmap 25-13
Graphic palette 25-1, 25-5
 arrow head 25-6
 border color 25-6
 circle 25-9
 drawing objects 25-5
 fill color 25-6
 line 25-7
 line type 25-6
 polygon 25-10
 polyline 25-10
 rectangle 25-8
 reference 25-11
 text 25-12
 thickness 25-6
Graphic pallet
 manipulate objects 25-6
Graphical comments 65-1
Grid 2-14
Group
 define 25-2
 delete 25-3, 25-14
 delete unused 25-3
 name 25-14
 rename 25-3, 25-14
 ungroup 25-2
Group selection 25-14
Groups of workers 7-14
H—
 Helper for modules 34-5
Hide 2-43
 Hide all graphic 25-5
Histogram distributed 3-26
I—
 Icons 67-1
IDs3Application 28-3
IDs3Control 28-4
IDs3Document 28-4
IDs3Element 28-8
IDs3Evaluation 28-9
IDs3Failure 28-9
IDs3Junction 28-10
IDs3Workare 28-11
Import
 DXF- format 25-4
 DXG-format 25-4
Include 2-22
Infeed element 4-47
 conveying entrance 4-47
 infeed entrance 4-47
Initial decision table 16-13
Initial variables 16-16
Initialization 16-2
 accumulation conveyor 4-10
 accumulation conveyor2 4-13
 track 5-3
Initializing 3-42
Insert 25-4
Install printer 2-24
Integer attribute 16-16
Interface
 Glossary 29-1
Interface functions 34-3
Interface of modules
 common 31-1
Interim statistic 11-11
Interrupt of tasks 69-15
Interval statistic 11-11
Introduction 67-1
 3D-visualization 26-1
 bitmap animation 12-15
 COM Server 28-1
 decision table 14-1
 decision table editor 16-1
 DOSIMIS-3 1-1
 excel interface 27-1
 optimization 10-12
 petri net 6-1
 program interface for controls 29-1
 program interface for decision tables 35-1
 shop floor control 24-1
 transport system 5-1
 work area 7-1
Introduction: 12-19, 61-1
J—
Junction 2-16
Junction attribute
 operation 18-27
 value 18-26
Junction variable 16-16
K—
Keyword: 27-7
Keywords 18-1, 27-3, 29-1
Kind of assembly 4-35
L—
Language reference 18-1
 animation text 18-34
 decision tables 18-10
 DT-Script 18-1
 file 18-34
 junction operation 18-27
 junction values 18-26
 mathematical function 18-9
 module and transport systems 18-21
 module operations 18-22
 module state 18-18
 module values
 read only 18-16
 read/write 18-17



write only 18-17
 modules and model structure 18-22
 modules and objects 18-19
 modules and strategies 18-20
 object values
 read only 18-23
 read/write 18-24
 order 18-33
 quicktable 18-28
 quicktable operations 18-29
 random function 18-8
 selectors 18-5
 sensor 18-33
 storage 18-21
 text output 18-11
 timer operations 18-30
 timer values 18-30
 transport order 18-31
 transport system 18-12, 18-33
 transport vehicles 18-25
 variables 18-6
 work 18-32
 worker 18-31
 working plan 18-32
 working step 18-32
 Layer 3-43
 LIFO buffer 4-18
 Line 25-7
 adding 16-4
 cancel input 16-5
 delete 16-8
 edit 16-4
 finish input 16-5
 parameter 25-7
 select 16-4
 Link 63-5
 Linking
 controls 2-17
 elements 2-16
 List
 select 2-57
 List of parameters
 program interface for decision tables 35-2
 Lists 3-38
 Load See *Open*
 Load file See *Open*
 Loaded run
 with order 5-27
 without order 5-27
 Loading station 5-4
 additional statistic 11-14
 default loading time 5-5
 destination code 5-5
 stop position 5-5
 Loading times 11-16
 Local variables 16-16
 Log window 20-1
 Lots 24-7
 —M—
 Mail 2-23
 Main direction
 Failure 8-7
 Maintenance 8-1
 Maintenance time 11-13
 Manual processing 11-15
 Marking of input fields 3-33, 3-37
 Mask
 parameter pack 15-8
 Maximum absolute occupancy 3-7
 Maximum contents 11-12
 Maximum free capacity 3-13
 Maximum relative occupancy 3-7
 Measuring element 8-24
 result table 8-30
 Menu 29-2
 junction selection 15-2
 Methods
 COM Server 28-2
 MFS 2-3
 change name See *Save As...*
 open 2-22
 save 2-22
 Minimum absolute occupancy 3-12
 Minimum contents 11-12
 Minimum free capacity 3-8
 Minimum relative occupancy 3-12
 Mirror 2-41
 Model
 send 2-23
 Modeling 63-1
 Module
 breakpoint 3-44
 initializing 3-42
 layer selection 3-43
 Module attribute
 model 18-22
 objects 18-19
 operations 18-22
 state 18-18
 storage 18-21
 strategies 18-20
 transport systems 18-21
 values
 read only 18-16
 read/write 18-17
 write only 18-17
 Module bunch 3-30
 Module cost
 diagram 11-28
 Module histogram
 comparison of several modules 11-44
 Module type
 accumulation conveyor 4-10, 62-3
 accumulation conveyor2 4-12
 assembly station 4-33
 bulk section 4-16
 combining station 4-46, 62-4
 conveying circuit 4-58
 conveying section 4-15
 crossing 4-49
 disassembly station 4-37
 discharger 4-43



- distributor 4-42, 62-4
- infeed element 4-47
- LIFO buffer 4-18
- loading station 5-4
- multifunctional workstation 4-40
- multiple workstation 4-32
- paired shuttle 4-66
- pallet changer 4-56
- petri net event 6-5
- petri net state 6-4
- service station 5-11
- shuttle 4-60, 62-3
- sink 4-9, 62-4
- source 4-1, 62-3
- storage 4-19
- swiveling belt 4-45
- track 5-2
- turning table 4-53
- unloading station 5-7
- waiting position 5-11
- work station 4-28, 62-3
- workstation with transport systems 5-9
- Module variable 16-16
- Modules
 - link 63-5
- Modules palette 2-13, 2-45
- Modules with a length of 0 3-29
- Monitoring 8-31, 17-4
- Mouse 2-3
- Move 2-41
 - decision table 16-14
- Multi Threading 31-14
- Multifunctional workstation 4-40
 - cleaning 4-41
 - work procedure 4-40
- Multiple tasks 4-3
- Multiple workstation 4-32
- N—
- Net attributes see Global variables
- Network License 37-2
- Normal distributed 3-22
- Notes 3-36
- Null position
 - conveying circuit 4-58
- Number of task 4-2
- O—
- Object attribute
 - transport vehicles 18-25
 - values 18-24
 - values (read only) 18-23
- Object cost
 - diagram 11-27
- Object length 4-14
- Object number 12-17
- Object property 26-6
- Object transfer 2-20
- Objective function
 - optimizing 10-14
- Occupancy diagram
 - absolute occupancy 11-35
- Offline simulation 31-14
- Online simulation 10-9, 22-1, 31-14
 - decision table 22-1
- Open 2-22
- Operation 24-3
 - shop floor control 24-1
- Operation reference 24-3
- Optimization
 - algorithm 10-16
 - buffer before sink 64-10
 - decoupling 64-6
 - decrease of buffer size before the sink 64-17
 - decrease of buffer size before work stations 64-17
 - diagram 10-17
 - disposal preferred 64-9
 - factory tuning 64-18
 - increase shuttle speed 64-8
 - introduction 10-12
 - mitigation of the sink 64-15
 - objective function 10-14
 - optimization run 10-15
 - parameter 10-14
 - presorting 64-3
 - result table 10-17
 - set-up time 64-13
 - summary 64-19
 - target values 10-15
- Optimization: 64-1
- Order
 - shop floor control 24-1
 - values 18-33
- Order duration 5-25
- Order statistic
 - transport system 5-22
- Order throughput time
 - transport system 5-23
- Order waiting queue 5-28
 - transport system 5-24
- Output file 16-17
- Output time 3-26
- Overlay of failures 8-9
- P—
- Paired shuttle 4-66
 - additional statistic 11-15
 - double cycle 4-68
 - global strategy 4-67
 - priority of double cycle 4-70
 - transport run with one Object 11-16
 - transport run with two objects 11-16
- Pallet changer 4-56
- Parameter 2-40, 16-16
 - 3D-Animation 26-12
 - accumulation conveyor 63-9
 - animation 12-2
 - bitmap 25-13
 - circle 25-9
 - combining station 63-16
 - continuous animation 12-20
 - distributor 63-17
 - line 25-7
 - optimization 10-14



polyline 25-10
 rectangle 25-8
 reference 25-11
 shuttle 63-10
 simulation 10-3
 sink 63-18
 source 63-7
 syntax editor 16-6
 text 25-12
 work area 7-4, 69-5
 work station 63-13
 working place 7-3
 Parameter mask 15-8
 Parameters 33-12
 Passivate
 work area 69-17
 Passivating 3-39
 Paste 2-38
 Path finding
 Shop floor control 24-1
 Path list 69-7
 Percentage
 distribution strategy 3-14
 right-of-way strategy 3-8
 Percentage contents 11-12
 Percentage utilization 11-13
 Periodical failure 8-6
 Petri net 6-1
 Condition/Event Nets 6-1
 DOSIMIS-3 6-2
 Positions/Transitions Networks 6-2
 Petri net event 6-5
 Petri net state 6-4
 Pick-up destination 5-17
 place of work
 work area 69-1
 Polygon 25-10
 Polygons 9-1
 Polyline 25-10
 parameter 25-10
 Positions
 conveying circuit 4-58
 shuttle 4-63
 Pre run statistic 11-11
 Preparation 12-15
 Pretty Print 2-23
 Preview 2-24
 Primary destination 5-17
 Print 2-24
 Priority
 assembly 4-35
 entrances 3-6
 exits 3-14
 object types 3-6
 tasks 7-6
 Priority of double cycle 4-70
 Priority of exits
 secondary strategy 3-16
 Problems 63-22
 Processing times 11-15
 Production diagram 11-43
 Production order 24-7
 Program interface
 editor 30-1
 placing a new control 32-9
 Program interface for controls
 application program interface functions 33-1
 common 31-1
 configuration 29-4
 debugging 31-2
 directory structure 31-12
 example 33-4
 Example 33-16
 introduction 29-1
 menu 29-2
 new interface 32-4
 new interface 32-1
 new user library 32-1, 32-4, 32-7
 placement 32-10
 Program interface for decision tables
 consistency check 35-2
 declaration 35-1
 introduction 35-1
 list of parameters 35-2
 Programming interface for modules
 data input 34-7
 events 34-1
 helper for modules 34-5
 interface functions 34-3
 state machines 34-1
 template 34-10
 template crossing 34-13
 template work 34-10
 Progress indicator 17-7
 Properties 2-24
 —Q—
 Quicktable 17-2
 operations 18-29
 values 18-28
 Quit See Exit
 —R—
 Random failure 8-5
 Rectangle 25-8
 parameter 25-8
 Redo 2-37
 Redraw 2-49
 Reference 25-11
 group selection 25-14
 parameter 25-11
 Reference failure 8-8
 Reference operator quantity 7-1, 7-2
 References 16-18
 delete 16-19
 exchange 16-19
 usage 16-20
 Relation
 rule matrix 16-8
 relector 18-1
 Rename
 decision table 16-14
 Restrictions 3-45, 5-28
 Result



further statistics 11-20
 Result color 11-10
 Result graphic
 block histogram 11-44
 capacity monitoring ⁸⁻¹⁴, 8-15
 control 11-49
 cost 11-26
 cost measurement 11-30
 single 11-31
 further graphics 11-49
 measuring module (single) 8-29
 module cost 11-28
 object cost 11-27
 occupancy diagram 11-32
 optimization 10-17
 production diagram 11-43
 smoothed occupancy 11-34
 state diagram 11-42
 throughput histogram 11-47
 throughput time histogram 11-48
 throughput time measurement (single) 8-22
 time histogram 11-46
 Transport-statistic 5-21
 unsmoothed occupancy 11-33
 work area statistic 7-17
 worker cost 11-29
 worker employment diagram 7-18
 Result parameter 11-2
 Result table
 additional statistic 11-14
 capacity monitoring 8-15
 measuring element 8-30
 optimization 10-17
 shop floor control 24-10
 statistical data 11-11
 throughput time measurement 8-23
 total statistic 11-11
 transport system
 utilization statistic 11-13
 work area 7-18
 Right-of-way 3-10
 Right-of-way strategy 3-4
 bauschuld 3-9
 FIFO 3-5
 maximum absolute occupancy 3-7
 maximum relative occupancy 3-7
 minimum free capacity 3-8
 percentage 3-8
 priority of entrances 3-6
 priority of object types 3-6
 self defined 3-9
 shortest path 3-10
 shortest running route 3-10
 Rotate 2-41
 Rotate (right) 2-42
 Route list 7-14
 Rule 16-2, 16-3
 breakpoint 16-10
 change 16-9
 copy 16-9
 create 16-9
 delete 16-9
 duplicate 16-9
 Rule matrix 16-8
 relation 16-8
 Rule window 16-8
 —S—
 Save 2-22
 As... 2-22
 Save & Compile 30-1
 Save to File 33-13
 Scheduling rules 7-5
 Scroll
 line 2-48
 page 2-48
 Search 2-51
 Secondary destination 5-17
 Secondary direction
 Failure 8-7
 Secondary strategy 3-15
 alternating distribution 3-16
 priority of exits 3-16
 Segment 5-3
 Select 25-4
 line 16-4
 list 2-57
 Select all 2-40
 Selection
 all 2-40
 next 2-41
 previous 2-40
 Selection of modules
 shop floor control 24-4
 Self defined
 right-of-way strategy 3-9
 Self-defined
 distribution strategy 3-17
 Sensor 4-13
 accumulation conveyor2 4-13
 values 18-33
 Service station 5-11
 Set-up times 4-31, 11-15
 Several workers per task 69-12
 Shift model 68-13
 Shop floor control 24-1
 alternative stations 24-4
 assembly operation 24-5
 disassembly operations 24-6
 editor 24-2
 file formats 24-12
 introduction 24-1
 lots 24-7
 operation 24-1, 24-3
 order 24-1
 path finding 24-1
 production order 24-7
 reference 24-3
 selection of modules 24-4
 statistic 24-10
 stock removal 24-6
 warehousing 24-6
 work plan 24-1, 24-3



- Shortest path 3-10
- Shortest route 3-17
- Shortest running route 3-10
- Shortest turning time 3-10
- Shuttle 4-60, 62-3
 - additional statistic 11-15
 - base point 4-63
 - basic position 4-61
 - empty run 11-16
 - initial position run 11-16
 - loading times 11-16
 - multiple load 4-64
 - parameter 63-10
 - positions 4-63
 - table 4-65
 - travel behavior 4-60
- Shuttle control 8-18
- Signal display 17-6
- Simulated failures in the model 68-1
- Simulation 27-7, 63-19
 - check output 10-1
 - consistency check 10-1
 - online 10-9
 - start 10-7
 - statistic output 10-2
 - trace output 10-2
- Simulation parameter
 - interval statistic 10-3
 - pre-run 10-3
 - random number 10-4
 - simulation time 10-3
- Simulator 33-7
 - part 2 33-10
- Single 37-2
- Single measuring 11-39
- Sink 4-9, 62-4
 - parameter 63-18
- Snap
 - Individual 2-42
 - parameters 2-42
 - relative 2-42
- Software 37-2
- Source 4-1, 62-3
 - attributes from file 4-8
 - generating from file 4-5
 - generating of objects 4-3
 - generation of objects 4-2
 - multiple tasks 4-3
 - number of task 4-2
 - parameter 63-7
 - time of generation 4-2
- Stacker crane 4-25
- Standard area data 4-20
- Standard values See Defaults
- Start
 - DOSIMIS-3 2-1
- Start work
 - Failure 8-7
- State cost 7-16
- State machines 34-1
- Statistic
 - abort statistic 11-11
 - additional statistic 10-2, 11-14
 - distribution of random time 11-19
 - final statistic 11-11
 - interim statistic 10-3, 11-11
 - interval statistic 11-11
 - Interval statistic 10-4
 - pre-run statistic 11-11
 - Pre-run statistic 10-4
 - select list 2-57
 - total statistic 11-11
 - utilization statistic 11-13
 - work area 69-10
- Statistic data
 - average occupancy 11-12
 - final statistic 11-14
 - occupancy 11-12
- Statistical data 11-11
- Statistics 63-20
 - failure 68-10
- Status bar 2-2, 2-45
- Stock removal
 - shop floor control 24-6
- Storage 4-19
 - area 4-19
 - attributes from file 4-8
 - capacities of areas 4-20
 - cost 4-26
 - detailing 4-21
 - double cycle 4-20
 - efficiency factor 4-21
 - exact location 4-21
 - exact locations 4-24
 - shop floor control 24-6
 - single cycle 4-20
 - stacker crane 4-25
 - storage status report 4-27
- Storage control 8-19
- Storage status report
 - storage 4-27
- Storage time 4-22
 - storage 4-22
- Strategy
 - alternating 3-13
 - alternating distribution 3-16
 - automatic path finding 3-18
 - bauschuld 3-9, 3-15
 - cycled See fixed
 - delivery strategy 4-23
 - destination 4-5
 - destination oriented 3-15
 - distribution strategies 3-11
 - erlang distributed 3-25
 - exponential distributed 3-24
 - FIFO 3-5
 - fixed distribution 3-19
 - forward control 3-29
 - histogram distributed 3-26
 - maximum absolute occupancy 3-7
 - maximum free capacity 3-13
 - maximum relative occupancy 3-7



minimum absolute occupancy 3-12
 minimum free capacity 3-8
 minimum relative occupancy 3-12
 modules with a length of 0 3-29
 normal distributed 3-22
 percentage 3-8, 3-14
 priority of double cycle 4-70
 priority of entrances 3-6
 priority of exits 3-14
 priority of exits 3-16
 priority of object types 3-6
 right-of-way strategy 3-4
 secondary strategy 3-15
 self defined 3-9
 self-defined 3-17
 shortest path 3-10
 shortest route 3-17
 shortest running route 3-10
 shortest turning time 3-10
 storage time 4-22
 transport system 5-12
 triangular distribution 3-21
 uniformly distributed 3-20
 user defined 15-2
String 18-4
String attribute 16-16
Structure of table 27-1
Structure view 16-12
Sub decision table 16-13
Subsequent dispatch 5-18
Swiveling belt 4-45
Symbol 33-5
Symbols
 define 25-15
 delete 25-3
 rename 25-3
 syntax editor 16-6
Syntax editor 16-5
 auto position 16-6
 parameter 16-6
 symbols 16-6
System start 2-1
—T—
Target values
 optimizing 10-15
Task 62-1
 cleaning 7-9
 elimination of failures 7-9
 maintenance 7-9
 object processing 7-7
 other activities 7-10
 set-up 7-9
 supervising 7-11
Task allocation 7-6
Task list 7-6
 completion 7-7
Task protocol 13-2
Template 34-10
 crossing 34-13
 files 31-12
 work 34-10
Terminating input 16-21
Test 3-37
Text 25-12
 parameter 25-12
Text editing 9-1
Thaw 2-43
Theory
 failure 68-1
 work area 69-1
Throughput 11-12
 histogram 11-47
Throughput dependent failure 8-7
Throughput time
 histogram 11-48
 measurement 8-20
 single 8-22
Throughput time measurement
 result table 8-23
Time between failure 68-2
Time histogram 11-46
Time of generation 4-2
Time to repair 68-2
Timer 17-5
 operations 18-30
 values 18-30
Toolbar 2-1, 2-45
Toolbars 2-10
Tooltip 2-11
Trace list
 select list 2-57
Track 5-2
 initialization 5-3
 object length 5-3
 segment 5-3
Track matrix 5-28
Transport duration 5-25
Transport entrance 4-34
Transport order
 values 18-31
Transport runs 11-16
Transport strategy 5-16
Transport system 5-1
 consistency check 5-27
 files 5-28
 loading station 5-4
 modules and strategies 5-1
 operations 18-33
 order statistic 5-22
 order throughput time 5-23
 order waiting queue 5-24
 result table 5-24
 service station 5-11
 strategy 15-6
 track 5-2
 unloading station 5-7
 waiting position 5-11
 workstation 5-9
Transport System
 Histogram 5-21
Transport System - Destination
 delivery 5-13



pick-up 5-13
 primary 5-13
 secondary 5-13
Transport System Scheduling
 task 5-12
 Transport time 11-14
 Transport vehicles 5-1
 Triangular distribution 3-21
 TS strategy level 5-12
Turning table 4-53
 additional statistic 11-15
 basic position 4-54
Tutorial: 61-1
Type of decision table 16-12
 activating decision table 16-12
 final decision table 16-13
 global decision table 16-12
 initial decision table 16-13
 sub decision table 16-13
—U—
Undo 2-36
Unhide 2-43
Uniformly distributed 3-20
Unloading station 5-7
 additional statistic 11-14
 default unloading time 5-8
 destination code 5-7
 object 5-8
Unloading times 11-16
Usage
 COM Server und VBA 28-13
 COM Server und Visual C++ 28-18
 references 16-20
User library
 new 32-1, 32-4, 32-7
Utilization statistic 11-13
—V—
Variable 16-15
 global 16-16
 local 16-16
Variable view 16-15
Varying distribution function
 functionality 3-28
Vehicle - Bypass 5-20
Version 2.3 2-23
Version 3.0 2-23
Version 3.1 2-23
Version 3.2 2-23
View 2-45
 control palette 2-45
 modules palette 2-45
 status bar 2-45
 toolbar 2-45
—W—
Wait in station 5-27
Waiting for assembly 4-34, 11-15
Waiting for worker 11-13
Waiting position 5-11
Waiting time 5-27, 11-14, 11-16
Waiting times 11-15
Warehousing
 shop floor control 24-6
Watch window 16-11
Way Point 2-14
Wildcard 4-30
Without statistic 11-14
Work
 values 18-32
Work area
 analysis 69-8
 assignment of worker 7-1
 breaks 69-6
 distances 69-7
 duration of tasks 7-2
 groups of workers 7-14
 include in the model 69-3
 interrupt of tasks 69-15
 introduction 7-1
 list of worker 7-4
 parameter 7-4, 69-5
 passivate 69-17
 place of work 69-1
 result table 7-18
 scheduling rules 7-5
 several workers per task 69-12
 state cost 7-16
 statistic 69-10
 strategies 15-7
 task 69-1
 task allocation 7-6
 task list 7-6
 theory 69-1
 work stations 69-3
 worker allocation 7-5
 worker diagram 7-17
 worker request 7-1
Work breaks 7-12
 show break 7-13
Work plan 24-3
 shop floor control 24-1
Work procedure 4-29, 4-40
Work station 4-28, 62-3
 and work area 69-3
 parameter 63-13
Worker
 values 18-31
Worker
 allocation 7-5
 conception 7-1
 cost diagram 11-29
 employment diagram 7-18
 list of worker 7-4
 request 7-1
Working method 27-1
Working place
 parameter 7-3
Working plan
 values 18-32
Working step
 values 18-32
Workstation
 additional statistic 11-14



change of object type 4-30
configuration 4-31
set-up times 4-31
with transport systems 5-9
—Z—
Zip-Archive 2-23
Zoom
elements 2-47

enlarge 2-46
model 2-47
next view 2-48
previous view 2-48
reduce 2-47
scaling 2-49
window 2-46
work area 2-47