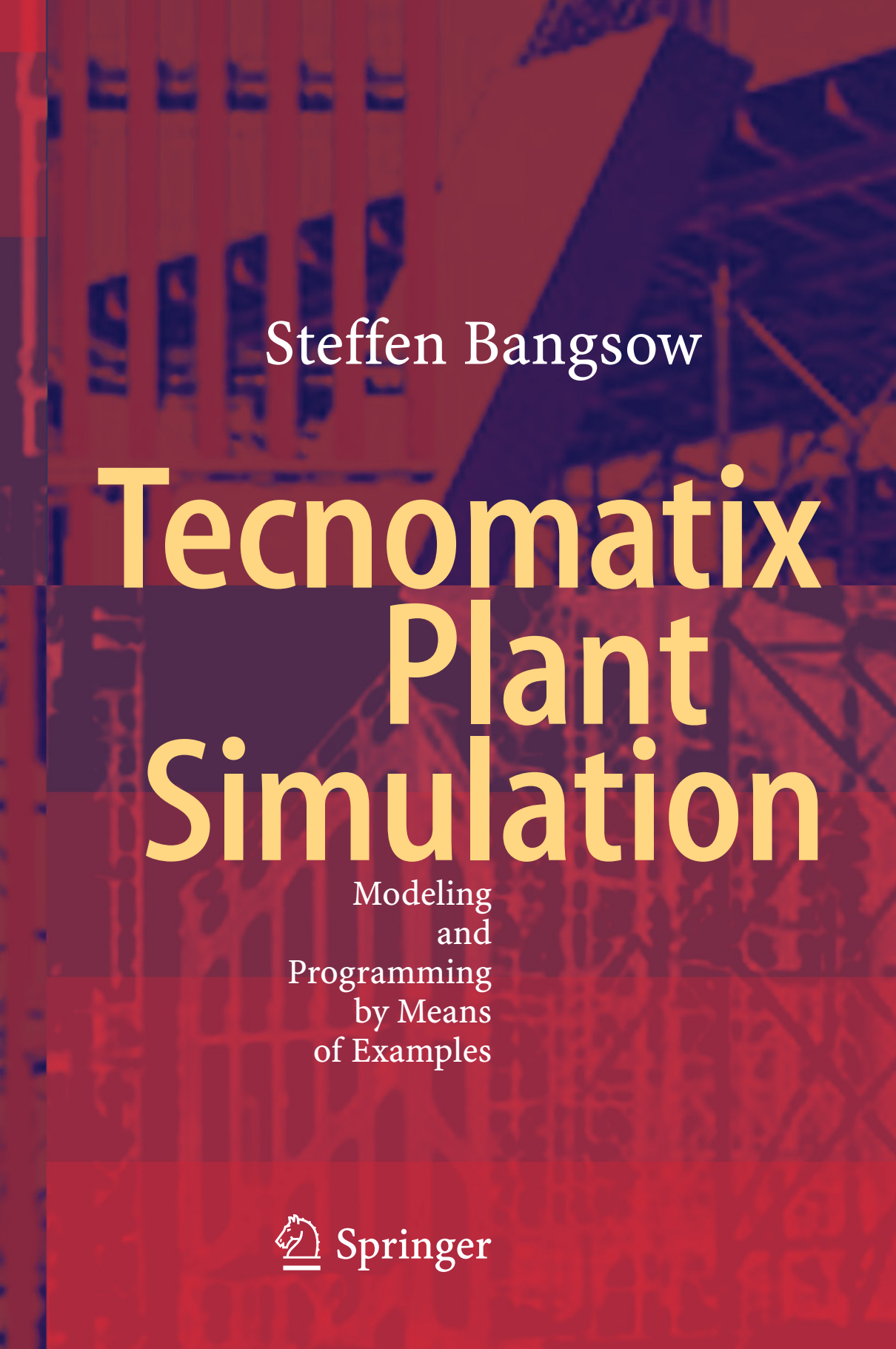
Steffen Bangsow
Tecnomatix Plant Simulation
Modeling and Programming by Means of Examples
Springer

Tecnomatix Plant Simulation

Steffen Bangsow

# Tecnomatix Plant Simulation

## Modeling and Programming by Means of Examples

Springer

Steffen Bangsow
Freiligrathstrasse 23
08058 Zwickau
Germany
steffen@bangsow.net

Translated by Steffen Bangsow

ISBN 978-3-319-19502-5 ISBN 978-3-319-19503-2 (eBook)
DOI 10.1007/978-3-319-19503-2

Library of Congress Control Number: 2015940995

Springer Cham Heidelberg New York Dordrecht London


Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

# Preface

Based on the competition of international production networks, the pressure to increase the efficiency of production systems has increased significantly. In addition, the number of technical components in many products and as a consequence also the requirements for corresponding assembly processes and logistics processes increases. International logistics networks require corresponding logistics concepts.

These requirements can be managed only by using appropriate Digital Factory tools in the context of a product lifecycle management environment, which allows reusing data, supports an effective cooperation between different departments, and provides up-to-date and relevant data to every user who needs it.

Simulating the complete material flow including all relevant production, storage, and transport activities is recognized as a key component of the Digital Factory in the industry and as of today widely used and accepted. Cutting inventory and throughput time by 20–60% and enhancing the productivity of existing production facilities by 15–20% can be achieved in real-life projects.

The purpose of running simulations varies from strategic to tactical up to operational goals. From a strategic point of view, users answer questions like which factory in which country suits best to produce the next generation product taking into account factors like consequences for logistics, worker efficiency, downtimes, flexibility, storage costs, etc., looking at production strategies for the next years. In this context, users also evaluate the flexibility of the production system, e.g., for significant changes of production numbers — a topic which becomes more and more important. On a tactical level, simulation is executed for a time frame of 1–3 months in average to analyze required resources, optimize the sequence of orders, and lot sizes. For simulation on an operational level, data are imported about the current status of production equipment and the status of work in progress to execute a forward simulation till the end of the current shift. In this case, the purpose is to check if the target output for the shift will be reached and to evaluate emergency strategies in case of disruptions or capacities being not available unexpectedly.

In any case, users run simulation to take a decision about a new production system or evaluate an existing production system. Usually, the value of those systems is a significant factor for the company, so the users have to be sure that they take the right decision based on accurate numbers. There are several random processes in real production systems like technical availabilities, arrival times of assembly parts, process times of human activities, etc., so stochastic processes play an important role for throughput simulation. Therefore, Plant Simulation provides a whole range of

easy-to-use tools to analyze models with stochastic processes, to calculate distributions for sample values, to manage simulation experiments, and to determine optimized system parameters.

Besides that, results of a simulation model depend on the quality of the input data and the accuracy of the model compared to the behavior of the real production system. As soon as assembly processes are involved, several transport systems with their transport controls, workers with multiple qualification profiles or storage logic, production processes become highly complex. Plant Simulation provides all necessary functionality to model, analyze, and maintain large and complex systems in an efficient way. Key features like object orientation and inheritance allow users to develop, exchange/reuse, and maintain their own objects and libraries to increase modeling efficiency. The unique Plant Simulation optimization capabilities support users to optimize multiple system parameters at once like the number of transporters, monorail carriers, buffer/storage capacities, etc., taking into account multiple evaluation criteria like reduced stock, increased utilization, increased throughput, etc.

Based on these accurate modeling capabilities and statistic analysis capabilities, typically an accuracy of at least 99% of the throughput values is achieved with Plant Simulation models in real-life projects depending on the level of detail. Based on the price of production equipment, a return on investment of the costs to introduce simulation is quite often already achieved after the first simulation project.

Visualizing the complete model in the Plant Simulation 3D environment allows an impressive 3D presentation of the system behavior. Logfiles can be used to visualize the simulation in a Virtual Reality (VR) environment. The support of a Siemens PLM Software unified 3D graphics engine and unified graphics format allows a common look-and-feel and easy access to 3D graphics which were created in other tools like digital product design or 3D factory layout design tools.

The modeling of complex logic always requires the usage of a programming language. Plant Simulation simplifies the need to work with programming language tremendously by supporting the user with templates, with an extensive examples collection and a professional debugging environment.

Compared to other simulation tools in the market, Plant Simulation supports a very flexible way of working with the model, e.g., by changing system parameters while the simulation is running.

This book provides the first comprehensive introduction to Plant Simulation. It supports new users of the software to get started quickly, provides an excellent introduction how to work with the embedded programming language SimTalk, and even helps advanced users with examples of typical modeling tasks. The book focuses on the basic knowledge required to execute simulation projects with Plant Simulation, which is an excellent starting point for real-life projects.

We wish you a lot of success with Tecnomatix Plant Simulation.

November 2009

Dirk Molfenter †
*Siemens PLM Software*

# Contents

# Chapter 1
# Basics

Simulation technology is an important tool for planning, implementing, and operating complex technical systems.

Several trends in the economy lead to shorter planning cycles. These include

- increasing product complexity and variety
- increasing quality demands in connection with high cost pressure
- increasing demands regarding flexibility
- shorter product life cycles
- shrinking lot sizes
- increasing competitive pressure

Simulation is used where simpler methods no longer provide useful results.

## 1.1 Introducing Material Flow and Logistics Simulation

### *1.1.1 Uses*

You can use simulation during the planning, implementation, and operation of equipment. Possible questions can include the following:

- Planning phase
  Identification of bottlenecks in the derivation of potential improvement
  Uncovering hidden, unused potentials
  Minimum and maximum utilization
  Juxtaposition of different planning alternatives
  Testing of arguments regarding capacity, effectiveness of control, performance limits, bottlenecks, throughput speed, and volume of stocks
  Visualization of planning alternatives for decision making
- Implementation phase
  Performance tests
  Problem analysis, performance test on future requirements
  Simulation of exceptional system conditions and accidents
  Training new employees (e.g. incident management)
  Simulation of ramp-up and cool-down behaviors



S. Bangsow, *Tecnomatix Plant Simulation,*
DOI: 10.1007/978-3-319-19503-2_1

- Operational phase
  Testing of control alternatives
  Review of emergency strategies and accident programs
  Proof of quality assurance and fault management
  Dispatching of orders and determination of the probable delivery dates

### *1.1.2 Definitions*

**Simulation** (source: VDI 3633[1])
Simulation is the reproduction of a real system and its dynamic processes in a model. The aim is to achieve transferable findings for reality. In a wider sense, simulation means preparing, implementing and evaluating specific experiments using a simulation model.

**System**: (VDI 3633)
A system is defined as a separate set of components that are related to each other.

**Model**:
A model is a simplified replica of a planned or real system with its processes in another system. It differs from the original in important properties only within specified tolerance levels.

**Simulation Run**: (source: VDI 3633)
A simulation run is the image of the behavior of the system in the simulation model within a specified period.

**Experiment**: (source: VDI 3633)
An experiment is a targeted empirical study of the behavior of a model through repeated simulation runs with a systematic variation of arguments.

### *1.1.3 Procedure of Simulation*

According to VDI guideline 3633, the following approach is recommended:

1. Formulation of problems
2. Test of simulation-worthiness
3. Formulation of targets
4. Data collection and data analysis
5. Modeling
6. Execution of simulation runs
7. Result analysis and result interpretation
8. Documentation

[1] VDI-Richtlinie 3633 Blatt 1(2000), Simulation von Logistik-, Materialfluss- und Produktionssystemen, Grundlagen. VDI-Handbuch Materialfluss und Fördertechnik, Bd. 8, Gründruck Beuth, Berlin, 2000.

#### 1.1.3.1 Formulation of Problems

Together with the customer of the simulation, the simulation expert must formulate the requirements for the simulation. The result of the formulated problem should be a written agreement (e.g. a technical specification), which contains concrete problems that will be studied using simulation.

#### 1.1.3.2 Test of Simulation-Worthiness

To assess simulation-worthiness you can, for example, examine:

- The lack of analytical mathematical models (for instance, many variables)
- High complexity, many factors to be considered
- Inaccurate data
- Gradual exploration of system limits
- Repeated use of the simulation model

#### 1.1.3.3 Formulation of Targets

Each company aims at a system of targets. It usually consists of a top target (such as profitability), which splits into a variety of sub-targets that interact with each other. The definition of the target system is an important preparatory step. Frequent targets for simulations are, for example:

- Minimize processing time
- Maximize utilization
- Minimize inventory
- Increase in-time delivery

All defined targets must be collected and analyzed statistically at the end of the simulation runs, which implies a certain required level of detail for the simulation model. Hence, they determine the range of the simulation study.

#### 1.1.3.4 Data Collection

The data required for the simulation study can be structured as follows:

- System load data
- Organizational data
- Technical data

The following overview is a small selection of data to be collected:

**Table 1.1** Data Collection

| | |
|---|---|
| Technical data | |
| Factory structural data | Layout<br>Means of production<br>Transport functions<br>Transport routes<br>Areas<br>Restrictions |
| Manufacturing data | Use time<br>Performance data<br>Capacity |
| Material flow data | Topology |
| | Conveyors |
| | Capacities |
| Accident data | Functional accidents |
| | Availability |
| Organizational data | |
| Working time organization | Break scheme |
| | Shift scheme |
| Resource allocation | Worker |
| | Machines |
| | Conveyors |
| Organization | Strategy |
| | Restrictions |
| | Incident management |
| System load data | |
| Product data | Working plans |
| | Bill of materials |
| Job data | Production orders |
| | Transportation orders |
| | Volumes |
| | Dates |

#### 1.1.3.5 Modeling

The modeling phase includes building and testing the simulation model.

Modeling usually consists of two stages:

1 Derive an iconic model from the conceptual model
2 Transfer the model into a software model

**First Modeling Stage**

First, you must develop a general understanding of the simulated system. Depending on the objectives to be tested, you have to make decisions about the

accuracy of the simulation. Based on the accuracy of the simulation, necessary decisions are taken about which aspects to simplify. The first modeling stage covers two activities:

- Analysis (breakdown)
- Abstraction (generalization)

Using the system analysis, the complexity of the system in accordance with the original investigation targets will be dissolved through meaningful dissection of the system into its components. Using abstraction, the amount of the specific system attributes will be decreased as far as it is practical to form a limited image of the original system. Typical methods of abstraction are reduction (elimination of irrelevant details) and generalization (simplification of essential details).

**Second Modeling Stage**
A simulation model will be built and tested. The result of modeling must be included in the model documentation to make further changes in the simulation model possible. In practice, this step is often neglected; hence, models cannot be used due to the lack of documentation of functionality. Therefore, there is a need for commenting on the models and the source code during programming. This ensures that the explanation of the functionality is still available after programming is complete.

#### 1.1.3.6 Executing Simulation Runs

Depending on the objectives of the simulation study, the experiments based on a test plan will be realized. In the test plan, the individual experiments on output data, arguments of the model, objectives, and expected results are determined. It is also important to define a time span for the simulation experiments, based on the findings of the test runs. Computer runs spanning several hours or frequent repetitive experiments for statistical coverage are not uncommon. In these cases, it is helpful to check whether it is possible to control the experiments using a separate programmed object (batch runs). The realization times for the experiments can be relocated partly at night, so that the available computing capacity can be utilized optimally. Input and output data as well as the underlying parameters of the simulation model must be documented for each experiment.

#### 1.1.3.7 Result Analysis and Result Interpretation

The values, which will change in the modeled system, are derived from the simulation results. The correct interpretation of the simulation results significantly influences the success of a simulation study. If the results contradict the assumptions made, it is necessary to analyze what influences are responsible for the unexpected results. It is also important to realize that complex systems often have a ramp-up phase. This phase may run differently in reality and in the simulation. Therefore, the results obtained during the ramp-up phase are often not transferable to the modeled system and may have no influence on the evaluation (Exception: The ramp-up phase of the original system has to be fully modeled).

#### 1.1.3.8 Documentation

For the documentation of a simulation study, the form of a project report is recommended. The documentation should provide an overview of the timing of the study and document the work conducted. Of interest in this context is the documentation of failed system variants and constellations. The core of the project report should be a presentation of the simulation results based on the customer requirement specification. Resulting from the simulation study, it makes sense to include proposals for actions in the documentation. Finally, we recommend describing the simulation model in terms of its structure and functionality.

## 1.2 Plant Simulation: First Steps

If you are new to Plant Simulation, you should start with the material provided by Plant Simulation. As a first step, you should reproduce all examples from the Plant Simulation Tutorial.

### *1.2.1 The Tutorial*

You gain access to the Tutorial via the Start Page of Plant Simulation. Until version 11, select View—Start Page; from version 12 onwards, click Window—Start Page (Fig. 1.1).

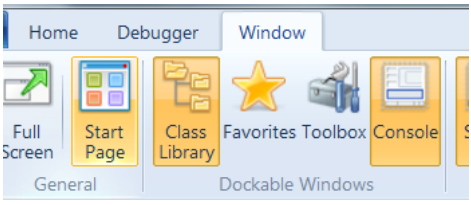


**Fig. 1.1** Start Page

Click on Tutorial. The online tutorial offers a quick start and guides you systematically in creating a simple simulation model. As a first step, you should practice all examples from the Plant Simulation Tutorial. After completing the tutorial, you will be ready to work with the examples in this book.

### *1.2.2 Step-by-Step Help*

The Step-by-Step Help feature provides descriptions of steps that are necessary to model several tasks. It is part of the chapter on online help entitled “Step-by-Step Help” (Fig. 1.2). Note: In version 12, the Help feature is located in the File menu.


**Fig. 1.2** Step-by-Step Help

The context-sensitive help in the dialogs of objects provides additional explanations of the dialog elements. To show context-sensitive help, click on the question mark on the top right corner of the dialog, and then click on the dialog element. The window of the context-sensitive help shows a reference to the corresponding SimTalk attribute at its end (professional license).

### *1.2.3 Example Collections and Demo Videos*

Plant Simulation contains a variety of examples of small models that are thematically ordered and show how and with which settings you can use the components and functions. You find a link to the examples on the Start Page of Plant Simulation (Fig. 1.3). You also find some demo videos here.

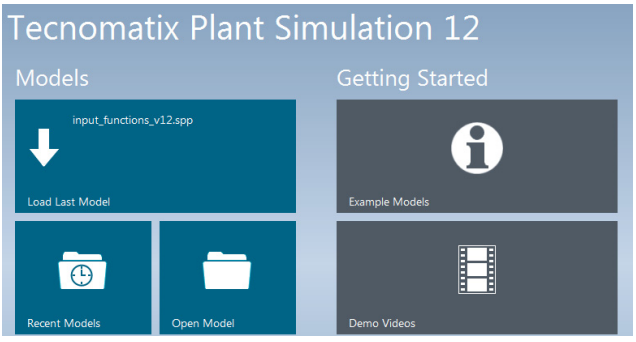


**Fig. 1.3** Example Collection

Most of the examples in this book (and many more) can be downloaded from my website: http://www.bangsow.de.

### *1.2.4 The Siemens PLM Software Community*

If you have questions regarding Plant Simulation, you can visit the PLM Software Community (Fig. 1.4). Just follow the link from the Start Page of Plant Simulation (starting from version 12), or type this address in your browser: http://community.plm.automation.siemens.com/.

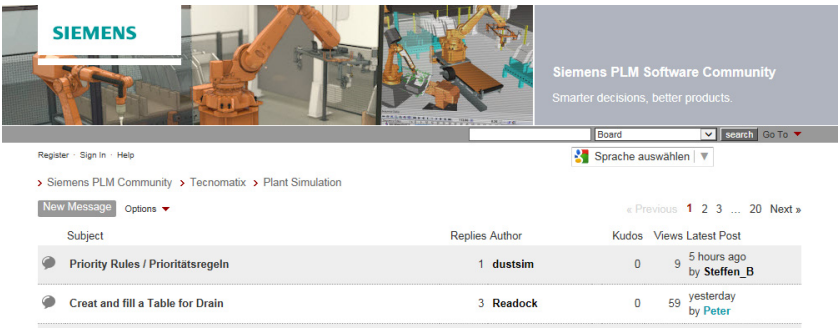


**Fig. 1.4** Siemens PLM Software Community

## 1.3 Introductory Example

Start Plant Simulation by clicking on the icon in the program group or the desktop icon.

### The Program Window (v. 12)

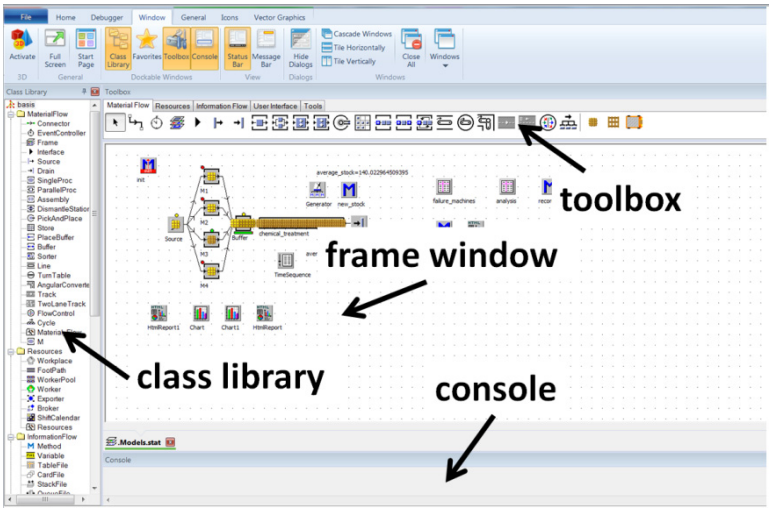


**Fig. 1.5** Program Window

To define the layout, you can use the menu item: WINDOW. Here, you set what you want to see on the screen. A standard Plant Simulation window can, for example, contain the elements from Fig. 1.5.

You show and hide elements from the program in the Window menu (Fig. 1.6).

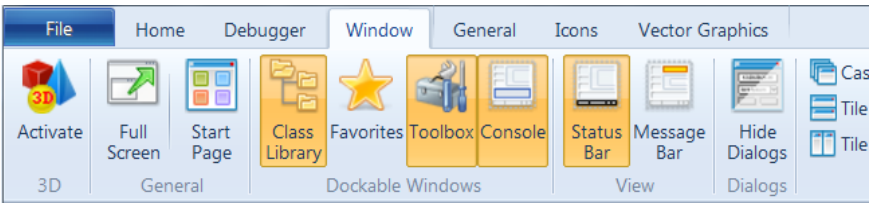


**Fig. 1.6** Window menu

**The Class Library**
In the class library, you find all objects required for the simulation. You can create your own folders, derive and duplicate classes, create frames, or load objects from other simulation models.

**The Console**
The console provides information during the simulation (e.g. error messages). You can use the Print command to output messages to the console. If you do not need the console, you can hide it.

**The Toolbox**
The toolbox provides quick access to the classes in the class library. You can easily create your own tabs in the toolbox and fill it with your own objects.

## 1.4 First Simulation Example

As a first example, a simple production line is to be built with a source (material producer), two workstations, and a drain (material consumer). Start Plant Simulation and select the menu command File—New. This opens a frame (window). Simulation models are created in the frame object.

### *1.4.1 Insert Objects into the Frame*

To insert objects into the frame, you have two options:

- Click on the object icon in the toolbox, and then click in the frame. The object will be inserted into the frame at the position at which you clicked.
- Another way is to drag the object from the class library to the frame and drop it there (drag and drop).

Insert objects into the frame as in Fig. 1.7.

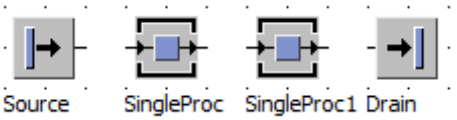


**Fig. 1.7** Model

### 1.4.2 Connect the Objects

You have to connect the objects along the material flow, so that the different parts can be transported from one object to the next. This is what the connector object does. The Connector has an icon as depicted in Fig. 1.8 in the toolbar.

**Fig. 1.8** Connector icon

Click the connector in the toolbar, then the object in the frame that you want to connect (the cursor changed its icon on connector), and click on the next object. If you want to insert several connectors successively, hold down the CTRL key. The result should look like in Fig. 1.9.

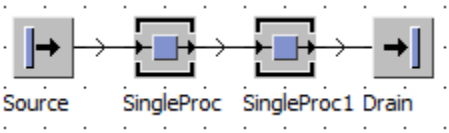


**Fig. 1.9** Connected objects

### 1.4.3 Define the Settings of the Objects

You have to define some settings in the objects such as processing times, capacity, information for setup, failures, breaks, etc. Properties can be easily set in the dialogs of the objects. You can open a dialog by double-clicking an object. For example, set the following values: SingleProc stations: processing time: two minutes (Fig. 1.10); drain: processing time: zero seconds; source: two-minute interval.

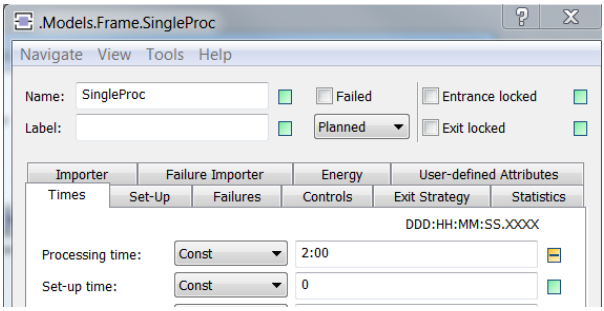


**Fig. 1.10** Dialog window

The Apply button saves the values, but the dialog remains open. OK saves the values and closes the dialog. Finally, you need the event controller. It coordinates the processes that run during a simulation. By default, Plant Simulation inserts an EventController into the frame.

### *1.4.4 Run the Simulation*

You can control the simulation with the icons in the left, upper corner of the Home menu (Fig. 1.11).

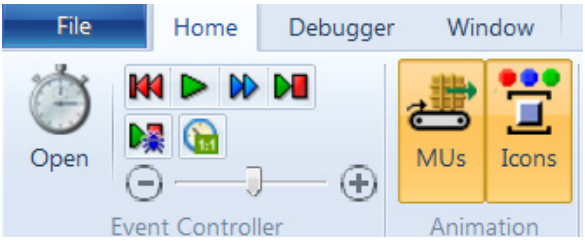


**Fig. 1.11** EventController Buttons

Click the Start button to start the simulation and Stop to stop the simulation. In the frame, the material movements are graphically displayed (Fig. 1.12). You can now change the model to see what happens. In the Statistics tab of the objects you will find statistical values that are collected by Plant Simulation.

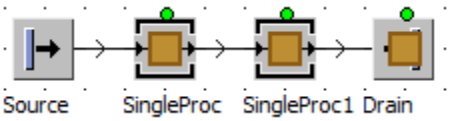


**Fig. 1.12** Animated MUs

## 1.5 Modeling

### *1.5.1 Object-Related Modeling*

In general, only a limited selection of objects is available for representing the real world. They are named, e.g. as model blocks, which show the real system with all the properties to be investigated. Hierarchically structured system models are best designed top-down. In this way, the real system will be decomposed into separate functional units (subsystems). If you are not able to model in a sufficiently precise manner using the available model objects, you should continue to decompose, etc. Each object must be described precisely (Fig. 1.13).

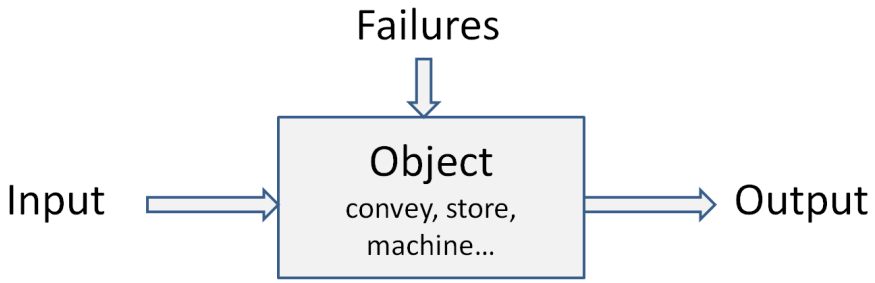


**Fig. 1.13** Simulation objects

The individual objects and the operations within the objects are linked to an overall process. This creates a frame. Using the objects and the frame, various logistical systems can be modeled.

### *1.5.2 Object-Oriented Modeling*

**Objects and Properties**
The hierarchical structure of an object allows being addressed exactly (analogous to a file path). A robot Rob1 may be addressed in the hierarchy (the levels are separated by a period) as follows:

production1.press_hall.section1.cell1.Rob1

Rob1 is described by a number of properties, such as type of handling, speed, capacity, lead times, etc. All properties that describe Rob1 are called object. An object is identified by its name (Rob1) and its path:

(production1.press_hall.section1.cell1.Rob1)

The properties are called attributes. They consist of a property description (attribute type)—e.g. engine type—and a property value (attribute value)—e.g. HANUK-ZsR1234578.

**Classes and Instances**
In object-oriented programming, a class is defined as follows: A class is a user-defined data type. It designs a new data type to create a definition of a concept that has no direct counterpart in the fundamental data types.

Example: You want to create a new type of transport unit that cannot be defined by standard types. All definitions (properties, methods, behavior) required for creating a new type, are called a class. The individual manifestation of the class is called an instance of the class (e.g. Transport—Forklift [general]; Instance: Forklift 12/345 [concrete]). The instance has the same basic properties as the class and some special characteristics (such as a specific name).

**Inheritance**
In Plant Simulation, you can create a new class based on an existing class (derive a class, create a subclass). The original class is called base class and the derived

class is called subclass. You can expand a data type through the derivation of a class without having to redefine it. You can use the basic objects of the class by employing inheritance.

Example: You have several machines of the same type; hence, most of the properties are the same. Instead of defining each machine individually, you can define a basic machine. All other machines are derived from this basic machine. The subclasses inherited the properties of the base class they apply to these classes as if they were defined there.

**Duplication and Derivation**

Try the following example: Select the SingleProc in the class library. Click the right mouse button to open the context menu. Select Duplicate from the Context menu. Plant Simulation names the duplicate SingleProc1. Change the processing time in the class SingleProc to two minutes. Open the dialog of the SingleProc1. The processing time has not changed.

The duplicate contains all the attributes of the original, but there is no connection between the original and the duplicate (there is no inheritance). You can also create duplicates using the mouse: Press the Control key and drag the object to its destination, then drop it.

Now do the same with Derive. Select the SingleProc again, click the right mouse button and click on Derive from the Context menu. Plant Simulation names the new class SingleProc2. With Derive you created an instance of the class. This instance can either be a new class (in the library) or an object (e.g. in a frame object). Initially, the instance inherits all the characteristics of the original class. Now, change the processing time of the SingleProc class to 10 minutes (10:00). Save the changes in the SingleProc and open the dialog of SingleProc2. SingleProc2 has applied the change in the processing time of the SingleProc class.

You can also derive in the class library using CTRL + SHIFT and dragging the mouse. You can navigate to the original class from an object or from a derived class. Double-click the class/the object and then select Navigate—Open Origin. You can also find a button on the Home menu (Fig. 1.14).

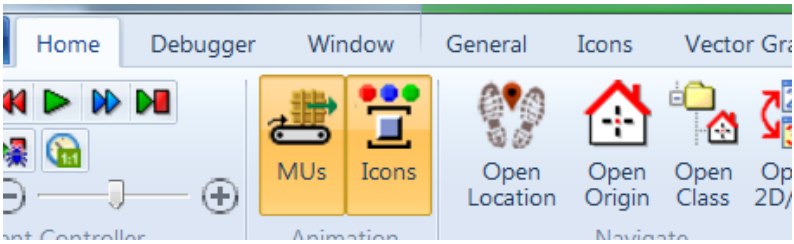


**Fig. 1.14** Home Menu

This will open the dialog of the original class. If you drag a class from the class library into a frame object, the new object is derived. Try this: Open a frame object. Add a SingleProc to the frame object (via drag and drop from the library).

Change the processing time of the SingleProc in the library and check the processing time in the frame. The values are inherited from the object in the library.

**Duplicating Objects in the Frame**
To duplicate an object in the frame, hold down the Ctrl key and drag the object to a free spot on the frame (the mouse pointer shows a "+").

Using the object, you can also activate inheritance to the initial class.

Test: Change the processing time of the SingleProc in the library; then the processing time of both SingleProcs in the frame also changes. If you change the processing time of one SingleProc in the frame, the processing time of the other SingleProc in the frame will not be changed (the duplicate has no inheritance relationship with its original, but with the original class of the original). In the object dialogs, you can easily identify which values are inherited and which have been entered in the instance. Each attribute shows a green toggle button to the right. This is green if the value is inherited and yellow with a minus sign inside if the values are not the same as in the original class.

| Value inherited | Value changed (inserted) |
|---|---|
| 1:00 | 5:00 |

To restore the inheritance of a value, click the button after the value and then click Apply.

## 1.6 Student and Demo Version

The student and demo versions are restricted to 80 elements. In this context, method blocks, connectors and some other elements are not counted. To find out how many of the restricted elements are included in your model, proceed as follows: Add a method block into your frame. Enter the following code:

```
is
do
  print numOfLimitedObjects;
end;
```

The console displays the number of limited elements. In version 12, the model already contains after its creation 33 limited elements. This is the content of the Tools folder. You can remove any items you do not need from the model. In version 12, select Home—Model—Manage Class Library. Uncheck all items in the Libraries tab (Fig. 1.15).

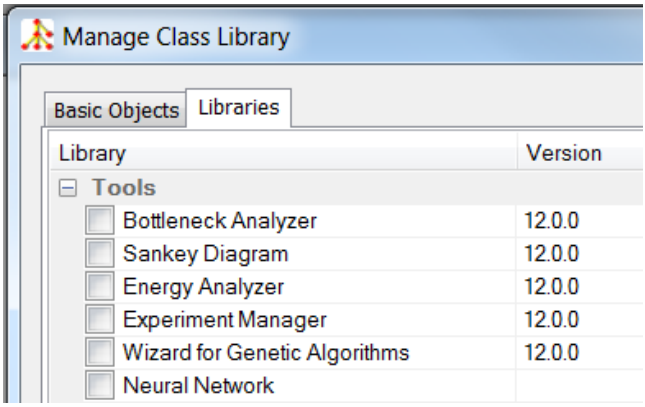


**Fig. 1.15** Manage Class Library

If you then check the number of elements, the result should be zero.

# Chapter 2
# SimTalk and Dialogs

The basic behavior of Plant Simulation objects in practice is often insufficient for generating realistic system models. To extend the standard features of the objects, Plant Simulation provides the programming language SimTalk. This allows modifying the basic behavior of individual objects. SimTalk can be divided into two parts: Control structures and language constructs (conditions, loops, etc.).

Standard methods of the material and information flow objects. These are built-in and form the basic functionality that you can use. You develop SimTalk programs in an instance of the information flow object Method.

## 2.1 The Object Method

You can create controls with Method objects, which are then called and started from the basic objects using their names. You find the Method in the class library within the folder InformationFlow (Fig. 2.1).


**Fig. 2.1** Class Library

**Example: Stock Removal**
We want to simulate a small production using a store. The capacity of the store is 100 parts. SingleProc produces one part every minute (the source delivers). Create a frame according to Fig. 2.2. Name the frame "storage."

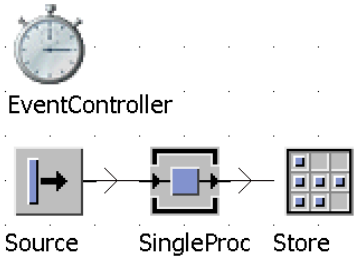


**Fig. 2.2** Example Frame



S. Bangsow, *Tecnomatix Plant Simulation,*
DOI: 10.1007/978-3-319-19503-2_2

If you now run the simulation, it soon results in a jam (storage is full). This is logical since there is no inventory decrease. You need a method that simulates an inventory decrease. Drag a method object to the frame. Double-clicking the icon opens the method. The methods (functions) always have the same structure:

```
is
do
  -- Statements
end;
```

Declare variables between "is and do." Enter your source code between "do and end." First, turn off the inheritance. Click on the icon (Fig. 2.3) in the editor or select Tools—Inherit Source Code.

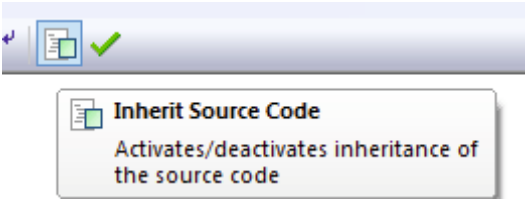


**Fig. 2.3** Inheritance

Formulate the instructions in SimTalk (call your method "stockRemoval"). Create the content according to Fig. 2.4.

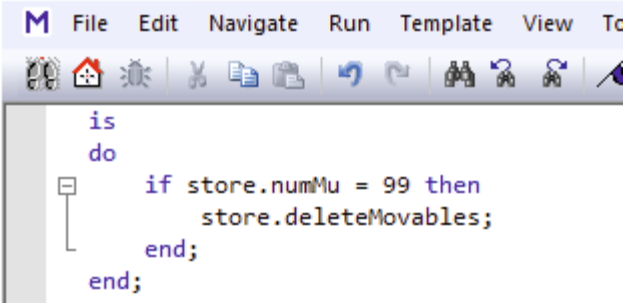


**Fig. 2.4** Method Editor

Confirm your changes with F7 or  and assign the method to an object. For this purpose, each object has one or more sensors. When an MU (Movable Unit) passes through a sensor, the relevant method is triggered. Double-click on Store—Controls—Entrance (Fig. 2.5); select the correct method and you are ready to proceed.

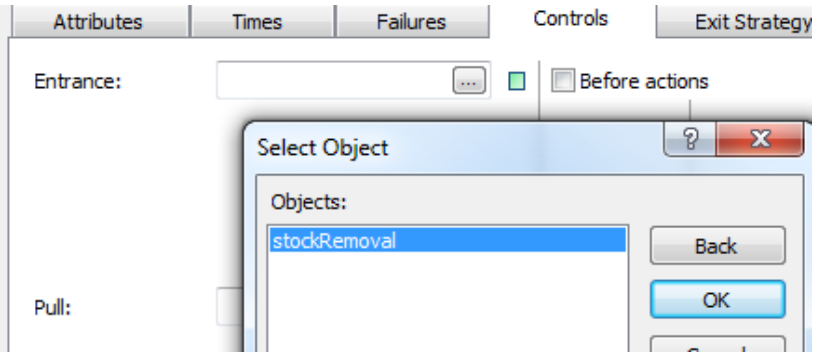


**Fig. 2.5** Entrance Control

Now start the simulation. If you were successful, there would be no jam. The store it will verify the quantity for each entry. If it is 99, then the store will be emptied (a simple solution).

## 2.2 The Method Editor

Double-clicking a method object opens an editor. You will find a number of functions in the editor that facilitate your work during programming. If you cannot enter your source code into the method editor, inheritance is still turned on (see above).

### *2.2.1 Line Numbers, Entering Text*

You can display line numbers using the command: View—Display Line Numbers. The following rules apply for entering text:

- Double-clicking selects a word
- Clicking three times selects a row
- Ctrl + A selects everything
- Copy does not work with a right click (until version 9). Use Ctrl + C to copy and Ctrl + V to insert text or use the menu commands Edit—Copy, etc.
- Use Ctrl + Z to undo the last change (or Edit Undo)
- Move also works by dragging with the mouse

### *2.2.2 Bookmarks*

For faster navigation, set bookmarks in your code. The bookmarks are displayed in red (see also Table 2.1). To insert a bookmark, select any text and click:  .

**Table 2.1** Bookmark functions

| Icon | Description |
|---|---|
|  | Deletes all bookmarks in the method |
|  | Cursor moves to previous bookmark |
|  | Cursor moves to next bookmark |

### *2.2.3 Code Completion*

The editor supports automatic code completion. If there is only one possibility of completion, Plant Simulation shows the attribute, the method or variable as a light blue label (Fig. 2.6). You can accept the suggestion with Ctrl + space bar.

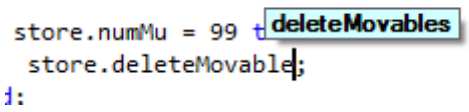


**Fig. 2.6** Code Completion

Starting from an object, you can display all possible completions. Simply press CTRL + SPACE (Fig. 2.7). In the list you can scroll using the direction buttons; an entry will be accepted with Enter.

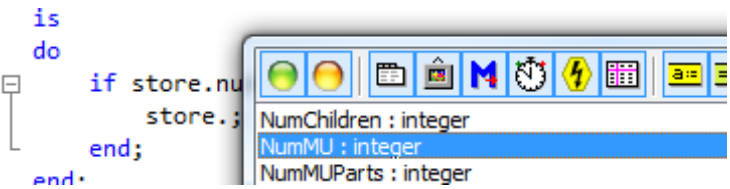


**Fig. 2.7** Object Attributes and Methods

### *2.2.4 Information about Attributes and Methods*

You can always get information about the built-in attributes and methods of an object through Show Attributes and Methods in the context menu of an object. In the table, you can see all methods and attributes (even those you have defined, Fig. 2.8) of the object.

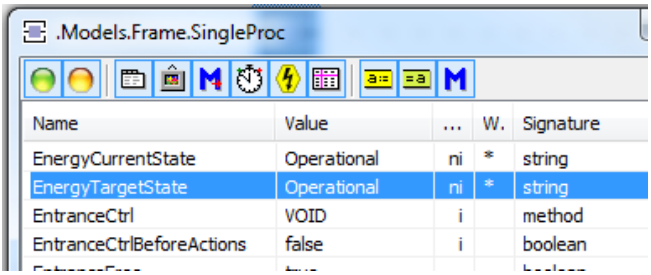


**Fig. 2.8** Attributes and Methods

The column signature allows you to deduce whether it is a method or an attribute and which data you need to pass or what type is returned. If the column signature shows only the data type, then the entry is an attribute.

Example: Show the attributes and methods of the store. Look for RecoveryTime (Fig. 2.9).

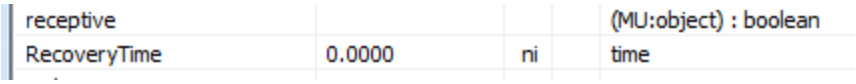


**Fig. 2.9** Attribute

Recovery time is an attribute; the data is not in parentheses. In order to set the recovery time, you have to type:

```
Store.recoveryTime:=120;
```

Set the value of an attribute with ":=".


**Fig. 2.10** Method

PE is a method (Fig. 2.10). It expects two arguments of data-type integer and returns any possible type (usually an object). PE allows you to access a particular place of the store. Call the method using parentheses:

```
Store.PE(1,1);
```


**Fig. 2.11** Method without Arguments

The row mirrorX does not contain a value in the column signature (Fig. 2.11). MirrorX flips the icon on the x-axis. It is a method that has no arguments and returns no value. You will call this method without parentheses.

```
Store. mirrorX;
```


**Fig. 2.12** Optional Parameter

The parentheses in the column signature indicate that startPause is a method. The data-type is given within square brackets. This means that the argument is optional, which results in two possibilities for the call:

```
Store.startPause; -- no time limit

Store.startPause(120); -- pause for 120 seconds
```

Note: Some attributes are read-only; you can assign no value to them. Online help describes whether an attribute is read-only or not.

### *2.2.5 Templates*

For a number of cases, Plant Simulation includes templates that you can insert into your source code or use as a starting point for developing controls. You reach Templates via the menu Template (Fig. 2.13):

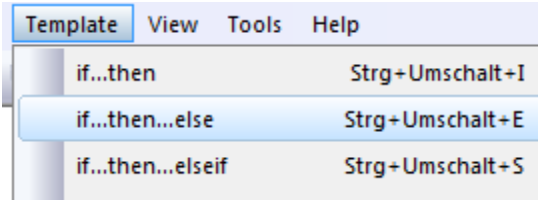


**Fig. 2.13** Templates

First click through the method editor on the row in which the snippet should be inserted. Then click, e.g. Template—if . . . then . . . . Under Select Template, you will find more templates. However, most templates in this selection will completely replace your source code. You can use the tab key to move through the template (between the areas in angle brackets).

### *2.2.6 The Debugger*

The debugger helps you to correct your methods. Using the F11 key, you can quickly change between editor and debugger (or Run—Debug). Example: Open the method from the example above, place the cursor in the text and then press F11 (Fig. 2.14).