

# M-XRAY

## User Guide

Release 1.4  
Dezember 2010

## How to get technical support for M-XRAY

Web: <http://m-xray.com>  
Email: [support@m-xray.com](mailto:support@m-xray.com)  
Tel: +49 (0)30 2091 6463 0  
Fax: +49 (0)30 2091 6463 33

## How to contact Model Engineering Solutions

Mail: Model Engineering Solutions GmbH  
Friedrichstr. 55  
10117 Berlin  
Germany  
Web: <http://www.model-engineers.com>  
Email: [support@model-engineers.com](mailto:support@model-engineers.com)  
Tel: +49 (0)30 2091 6463 0  
Fax: +49 (0)30 2091 6463 33



### Important notice

This document contains proprietary information that is protected by copyright. All rights are reserved. Neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Model Engineering Solutions GmbH.

© 2009-2010: Model Engineering Solutions GmbH, Friedrichstr. 55, 10117 Berlin, Germany

This publication and the contents hereof are subject to change without notice.

# Contents

<b>1 What is M-XRAY and How can it help You? .....</b>	<b>7</b>
1.1 What is M-XRAY? .....	7
1.2 How M-XRAY Supports Model Development.....	7
1.3 How M-XRAY Supports Model Reviews .....	8
1.4 How M-XRAY Supports Quality Assurance .....	8
1.4.1 In-depth information on model structure .....	9
1.4.2 Distribution of complexity across the model .....	9
1.4.3 Comparing complexity of different models.....	10
1.4.4 Measuring and assessing model metrics.....	10
1.4.5 Support in planning reviews .....	10
1.4.6 Support in conducting model reviews .....	10
1.5 How to Read the M-XRAY User Guide .....	11
1.6 Required Knowledge .....	11
1.7 What's New in the New M-XRAY Versions .....	11
1.7.1 Version 1.4 compared with 1.3 .....	11
1.7.2 Up to version 1.3.....	12
<b>2 Installation and System Requirements.....</b>	<b>13</b>
2.1 Installing M-XRAY .....	13
2.2 M-XRAY System Requirements .....	14
<b>3 Quick Start .....</b>	<b>15</b>
3.1 Launching M-XRAY and its Components.....	15
3.2 Conducting your First Analysis with an HTML/Excel Report .....	17
3.2.1 Selecting a model for analysis .....	17
3.2.2 Changing name and path of the report file (optional) .....	17
3.2.3 Generating an HTML analysis report .....	17
3.2.4 Generating an Excel analysis report .....	20
3.3 Generating a Review Report .....	21
3.4 Selecting Models that are not Open.....	21

<b>4</b>	<b>Preparing Models for Analysis with M-XRAY</b>	<b>23</b>
4.1	Displaying Model Information	23
<b>5</b>	<b>Working with M-XRAY</b>	<b>25</b>
5.1	Executing M-XRAY via the GUI	25
5.2	Executing M-XRAY via the Command Window	25
5.2.1	Selecting models for analysis with M-XRAY	26
5.2.2	Configuring analysis reports (1): Defining the execution variant	26
5.2.3	Configuring analysis reports (2): Details of results display	28
5.3	Exporting Metrics	30
<b>6</b>	<b>How M-XRAY Presents Results</b>	<b>33</b>
6.1	Analysis Report Format (HTML)	33
6.1.1	Naming and number of results files	33
6.1.2	Structure of an HTML report	34
6.1.3	Metric overview section	35
6.1.4	Contents section	36
6.1.5	Most complex subsystems section	36
6.1.6	Main part section	37
6.1.7	Library section	38
6.1.8	Structural overview section	39
6.2	Analysis Report Format (Excel)	39
6.2.1	Naming of the results file	39
6.2.2	Template file and post-processing using macros	40
6.2.3	Structure of the tables	40
6.3	Explanation of Columns in Results Display	40
6.3.1	Group 1: Columns 'Path', 'Name', 'Info', and 'Level'	41
6.3.2	Group 2: Columns 'Local Complexity' and 'Global Complexity'	41
6.3.3	Group 3: Columns 'Blocks (local)' and 'Blocks (global)'	42
6.3.4	Group 4: Columns 'Interface In' and 'Interface Out'	42
6.3.5	Group 5: Columns with base values of Simulink metrics	43
6.3.6	Group 6: Columns with base values of Stateflow metrics	44
6.3.7	Group 7: Columns with local number of blocks of different types	45
6.3.8	Group 8: Columns with global number of blocks of different types	46
<b>7</b>	<b>M-XRAY Review Report</b>	<b>47</b>
7.1	Creating a Model Review File	47
7.1.1	Review file template	47
7.1.2	Naming the review file	47

7.2 Tables in the Review File .....	47
7.2.1 Summary table .....	48
7.2.2 Issue tracking table .....	48
7.2.3 Tables providing an overview of model structure .....	50
7.2.4 Maintenance table.....	51
7.2.5 Template table .....	51
<b>8 Measuring Model Complexity.....</b>	<b>53</b>
8.1 Goals of Complexity Measurement.....	53
8.2 Software Metrics Model .....	53
8.3 Calculating Model Complexity (Simulink).....	54
8.3.1 Application to SL/TL systems.....	54
8.3.2 Base quantities of the metric.....	55
8.3.3 Example of a Simulink metric calculation .....	55
8.4 Calculating the Complexity of Loops.....	56
8.5 Calculating Model Complexity (Stateflow).....	57
8.5.1 Complexity of states.....	58
8.5.2 Complexity of transitions.....	58
8.5.3 Complexity of truth tables .....	58
8.5.4 Example of a Stateflow metric calculation .....	59
8.6 Scaling of Metrics .....	59
8.7 Displaying Model Structure.....	59
8.8 Weighting of Blocks.....	60
<b>9 How to Configure M-XRAY .....</b>	<b>63</b>
9.1 Configuration of Mask Types .....	63
9.1.1 Configuring the subsystem under analysis .....	63
9.2 Configuration of Weights of SL/TL Blocks.....	64
9.2.1 Example of changing the weights in the <code>mxray_blockweights</code> file .....	65
9.3 Configuration of Weights of Stateflow Objects.....	66
9.3.1 Example of changing the weights in the <code>mxray_SFweights</code> file .....	67
<b>10 Directory Structure of M-XRAY .....</b>	<b>69</b>
10.1 Files in the Main Directory .....	69
10.2 Files in the Configuration Directory .....	70
10.3 Files in the Templates Directory .....	70
10.4 Files in the Functions Directory .....	70

10.5 Files in the Utils Directory .....	70
<b>11 Tables Appendix.....</b>	<b>71</b>
11.1 Block Weights for Simulink Blocks.....	71
11.2 Block Weights for TargetLink Blocks .....	71
11.3 Weights for Stateflow Elements .....	72

# 1 What is M-XRAY and How can it help You?

The use of MATLAB®/Simulink® and TargetLink is an industry-wide norm in model-based development and code generation. The quality of generated code is determined by the complexity and quality of the developed models. This is why it is crucial that these models be safeguarded by appropriate quality assurance measures.

Complexity determination is an important piece of routine work to assess the quality of model implementations. M-XRAY is a tool that puts detailed complexity information at your disposal, making the task of quality assessment that much easier.

The use of M-XRAY fulfills the following requirements in regard to quality assurance:

- First tool to provide substantiated evidence of model complexity
- Analysis and assessment of model architecture
- Appraisal of error susceptibility of models
- Identification (detection) of complex subsystems
- Appraisal of review and testing effort as a basis for planning preparatory measures
- Monitoring the development of total complexity and breakdown into modules as process-accompanying measures.

## 1.1 What is M-XRAY?

M-XRAY is a tool that has been designed to **perform structural and complexity analysis** on Simulink/Stateflow and TargetLink models, thereby supporting model development and the model review process. M-XRAY has been developed to **analyze models of every size**: from single subsystems to complex models consisting of hundreds of subsystems and dozens of libraries. The results of analysis are **compiled in reports** designed for use in model development and model reviews. The reports can also be used to evaluate model quality as well as for project management purposes.

This User Guide is written by model developers for model developers. It investigates the task or activity to be solved and explains exactly how this can be done with M-XRAY. M-XRAY in combination with this User Guide provide the ideal support for you to analyze your models both quickly and efficiently. But M-XRAY supports much more activities. It also aids the observation of model quality during the life cycle. So M-XRAY also assists testers during quality assurance issues and manager in tasks of project controlling.

## 1.2 How M-XRAY Supports Model Development

M-XRAY provides model developers with:

- a detailed overview of their models' hierarchical structure
- plus further in-depth information concerning content and complexity

Information is presented on every subsystem as to:

- its **structure** (position in the model)
- its **block content**
- the **widths of its** incoming and outgoing **interfaces**
- as well as a **metric value** that represents the overall complexity of its content

Different complexity levels are color-coded for an easier overview.

Moreover, direct links from the M-XRAY report to the model itself allow:

- fast and easy navigation to critical parts of its structure

### 1.3 How M-XRAY Supports Model Reviews

M-XRAY offers ideal support for model reviews by automatically creating a report file with helpful structural information about the models under investigation, as well as special forms for issue tracking. This report file gives the reviewer a good overview of the model, enabling him to structure the reviewing process and keep track of which parts of the model have still to be reviewed and in which components issues have been found.

The model reviewer's efficiency is increased considerably as a direct result of working with M-XRAY. Moreover, the information provided by M-XRAY can help to approximate in advance the time and effort required for a model review, thereby allowing a more efficient allocation of review resources.

### 1.4 How M-XRAY Supports Quality Assurance

This section discusses examples of using M-XRAY in quality assurance. A general overview is shown in Figure 1-1.

- In-depth information on model structure; see Chapter 1.4.1.
- Distribution of complexity across the model; see Chapter 1.4.2.
- Measuring and assessing model metrics; see Chapter 1.4.4.
- Comparing complexity of different models; see Chapter 1.4.3.
- Support in planning and conducting model reviews; see Chapter 1.4.5 and Chapter 1.4.6.

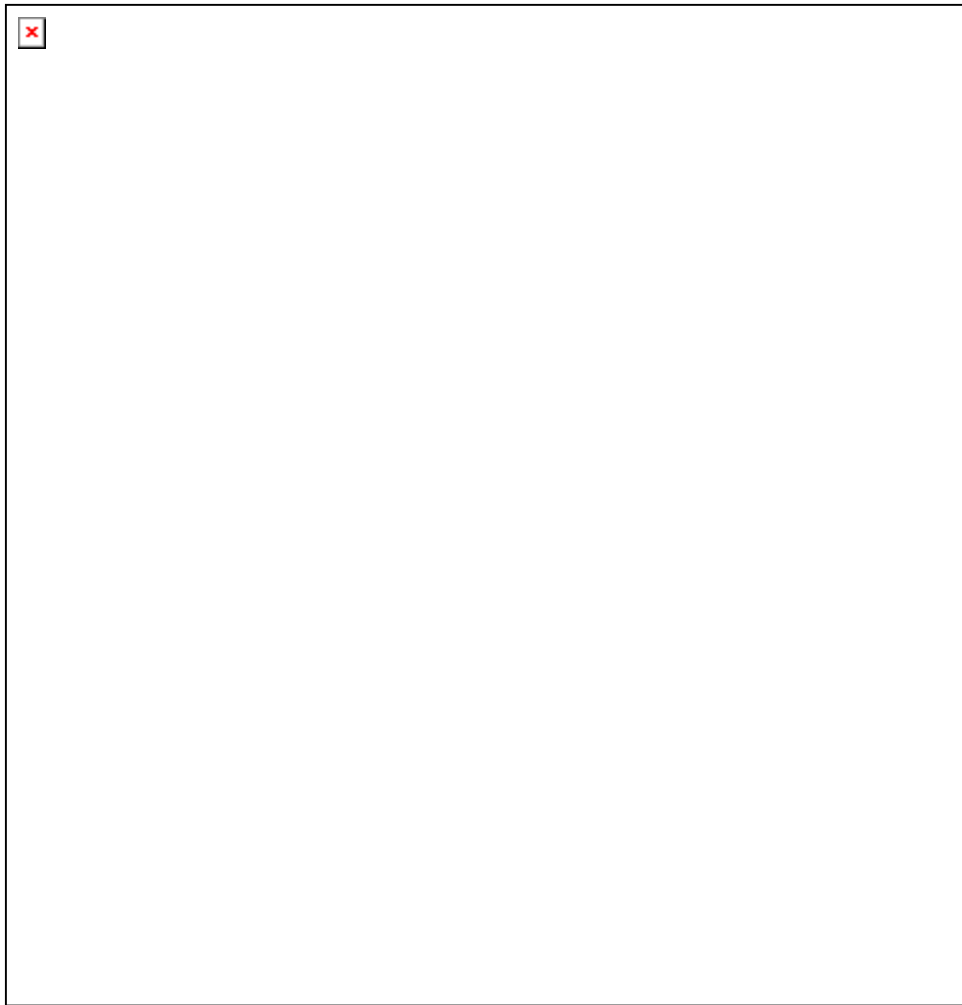


Figure 1-1: Applications of M-XRAY in quality assurance

#### 1.4.1 In-depth information on model structure

---

M-XRAY gives you a detailed overview of the hierarchical structure of models. All subsystems are shown together with their position in the model tree. Libraries are listed separately and are only listed once per model.

Additional information on the content of individual subsystems is given, including the type and number of blocks they contain, the width of incoming and outgoing subsystem interfaces, and the number of loops between blocks. This overview of the structure and content of a model is a great addition to conventional model documentation.

#### 1.4.2 Distribution of complexity across the model

---

Complexity values are computed for single subsystems as well as for model parts and the model as a whole. This enables you to verify overall model structure.

M-XRAY makes it possible to see which model parts exhibit a high degree of complexity (and thus a more complex functionality).

M-XRAY also makes it possible to determine whether overall complexity is more or less evenly distributed across all module parts or whether complexity is heavily concentrated in certain parts of the model.

This makes it easy to control the structure of models and improve on any inherent deficits.

### 1.4.3 Comparing complexity of different models

---

A metric value that represents overall complexity makes it possible to compare the scope of different models. Smaller models tend to have a global (overall) complexity value ranging from 1,000 to 4,000 (e.g. the `fuelsys` demo model example discussed in Chapter 3). Medium-sized models display global complexity values of between 15,000 and 40,000. Very comprehensive models can reach global complexity values of more than 50,000.

### 1.4.4 Measuring and assessing model metrics

---

In addition to the local and global complexity of models, other metrics can also be recorded and evaluated. These include metrics such as the number of blocks, the number of modeling levels, and the number of inputs and outputs.

M-XRAY supplies an evaluation for some of these values as part of the model quality evaluation process. In this way evaluations of local complexity and depth of modeling levels are supplied. The results are shown in the report overview.

### 1.4.5 Support in planning reviews

---

Knowing the complexity of models and model parts is very useful when preparing and planning model reviews.

It allows you to make an accurate estimate of the time and resources needed to perform the entire review in advance, as experience has shown how to correlate the complexity value with the workload involved (number of personnel hours).

At the same time, an understanding of model complexity gives you a much better idea of how long it will take to check individual module parts, enabling the total workload to be divided between different reviewers where necessary.

### 1.4.6 Support in conducting model reviews

---

M-XRAY creates Excel documents that simplify the model review process.

The structure of individual models that have been investigated is displayed on separate Excel sheets. Information on the review status of individual parts of each model can also be recorded here. This enables a structured review of models, as well as providing an overview of which model parts have already been checked, which still remain unchecked, and the parts in which a particularly large number of issues have been detected.

An additional sheet describes and categorizes all detected issues in a clear and concise way. You can find more information on the structure of reports in Chapter 7.2.

## 1.5 How to Read the M-XRAY User Guide

Before beginning to work with M-XRAY, you first need to install the program. The installation procedure for M-XRAY is described in depth in Chapter 2.1, p.13.

Chapter 3, p.15 takes you through model analysis with M-XRAY step by step. It should be possible for you to work through this chapter using the example provided within half an hour, after which you will have a good understanding of the basic workings of M-XRAY. The example model and all required data are already part of your installation. You can therefore begin familiarizing yourself with the functions and options offered by M-XRAY immediately.

Chapter 4, p.23 describes the steps needed to prepare models for analysis with M-XRAY.

Chapter 5, p.25 and subsequent chapters look at individual aspects of working with M-XRAY in greater detail. We begin by describing how to operate M-XRAY via the command line. This allows special configuration of the information contained in the report files resulting from analysis with M-XRAY. The more comfortable way of operating M-XRAY via the GUI is described in detail in the Quick Start introduction in Chapter 3.

The results of analysis with M-XRAY can be stored as an HTML file or an Excel file. Chapter 6, p.33 describes the format and content of M-XRAY report files.

Chapter 7, p.47 describes the structure of review reports, as well as how to create and use them. The content and format of these reports have been specially designed to support the model review process.

Chapter 8, p.53 discusses the topic of model complexity. Following a brief introduction to software complexity metrics, we will then describe the algorithms used by M-XRAY to calculate model complexity and investigate them using examples.

M-XRAY offers the optional function of altering certain parameters for calculating model structure and complexity. These parameters can be found in the configuration files and can be adapted at will. Chapter 9, p.63 describes the structure and configuration of these files.

Finally, Chapter 10, p.69 describes the structure and content of the M-XRAY installation directory.

## 1.6 Required Knowledge

The reader of this User Guide should be familiar with Simulink and Stateflow (dSPACE TargetLink knowledge is optional). A basic understanding of the MATLAB M-file language is useful in order to use some of M-XRAY's extended capabilities.

## 1.7 What's New in the New M-XRAY Versions

### 1.7.1 Version 1.4 compared with 1.3

The following functions have been integrated into the tool and the M-XRAY User Guide with the introduction of M-XRAY 1.4:

- The new metric overview `SubsystemLevel` has been added in addition to the metrics introduced in version 1.3. This gives you an evaluation of the modeling of subsystem levels. Please refer to Chapter 6.1.3, p.35 for more details.
- Identified metrics can now be exported. The command `mxray_metricExport` exports them in CSV or XML file formats. Please refer to Chapter 5.3, p.30 for more details.
- A new section has been added for calculating loop complexity. This describes how complexity is determined and how this calculation is added to local complexity. Please refer to Chapter 8.4, p.56 for more details.

### 1.7.2 Up to version 1.3

---

With the introduction of M-XRAY 1.3 the following functions have been integrated into the tool and the M-XRAY User Guide:

- A new overview has been introduced into the analysis report. An overview bar shows the proportion of complex subsystems within the system as a whole. This gives the user a quick and clear overview of model quality. Please refer to Chapter 6.1.3, p.35 for more details.
- The introduction of black and white lists enables the user to determine at will which subsystems should be examined during analysis. This allows you to determine the scope and depth of detail you require from your analysis. Please refer to Chapter 9.1, p.63 for more details.
- The weighting of Simulink/TargetLink and Stateflow blocks in the complexity calculation is described in detail in this User Guide. Chapter 8.8, p.60 has been added for this purpose.
- To make working with the M-XRAY GUI more simple, the settings you apply are stored as a configuration and are reapplied after a restart.

Additional analysis options make it easier to control the degree of detail provided in the analysis report

## 2 Installation and System Requirements

This chapter describes the M-XRAY installation procedure and system requirements.



For the purposes of this User Guide, entries in MATLAB's command window will be shown as follows: `>> mxray`. To start M-XRAY the text to be entered is `mxray;`. The text `>>` simply represents MATLAB's command line prompt.

### 2.1 Installing M-XRAY

If you have received M-XRAY in a single zip file, begin by extracting it into a folder. Please retain all the subdirectories. Here are a few suggestions that we often encounter in our projects:

- C:\Program Files\MXRAY
- x:\project\matlab\mxray
- %MATLAB\_ROOT%\toolbox\MXRAY

There are three possible ways to set up your M-XRAY installation.

If you want to make M-XRAY available for all future MATLAB sessions, please perform the following: Include the `...\mxray` directory in your MATLAB path:

Start MATLAB

1. Select `File/Set Path...` from the MATLAB menu
2. Add folder
3. Select the `mxray` folder in your M-XRAY installation directory (Figure 2-1)
4. Push OK
5. Select `Save` to save the adjusted path for future MATLAB sessions

Alternatively, if you only want to add M-XRAY temporarily to the MATLAB search path:

1. Start MATLAB
2. Navigate to the `mxray` folder in your M-XRAY installation directory
3. Right-click on the file `mxray_path.m` and select `Run` from the context menu. This temporarily adds M-XRAY to MATLAB's search path.

If you use the M-XRAY GUI (as described below), the temporary addition of M-XRAY to the MATLAB search path is performed automatically during GUI initialization.

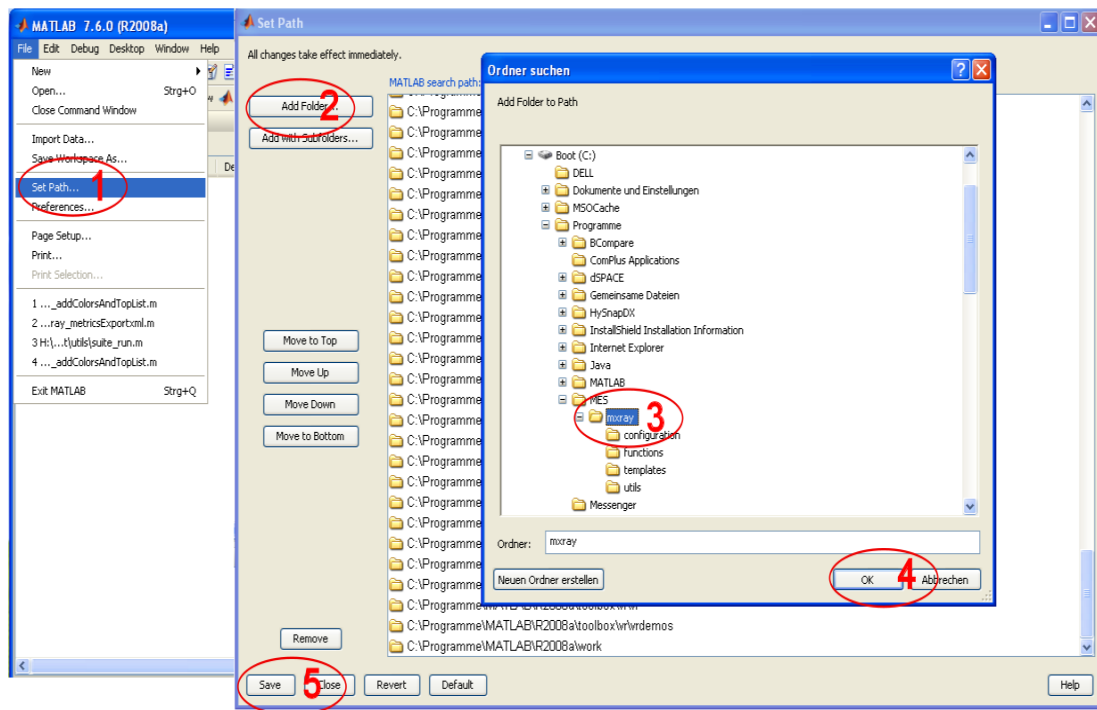


Figure 2-1: Select `mxray` folder in your M-XRAY installation to add it to MATLAB's search path

You are now ready to begin working with M-XRAY. M-XRAY can be accessed from within MATLAB from any directory you choose.

There are two alternative ways to launch and use M-XRAY:

- The M-XRAY GUI can be launched by entering  
`>> mxray`  
 on the MATLAB command line (see Chapter 3 for a full explanation of the M-XRAY GUI).
- The execution of M-XRAY can also be launched directly by entering  
`>> mxray_createReport(<list of models>,<list of parameters>)`  
 on the MATLAB command line (see Chapter 5.2).

## 2.2 M-XRAY System Requirements

The following system requirements must be in place in order to use M-XRAY with MATLAB/Simulink/Stateflow and TargetLink:

- MATLAB R14SP3 or newer, also tested with R2006b, ..., R2010b
- Simulink and Stateflow, depending on the models you wish to analyze
- (Optional) TargetLink V 2.x and 3.x (base suite)

## 3 Quick Start

This Quick Start chapter will take you through the steps involved in performing model analysis with M-XRAY using a simple model as an example. Working through this chapter will give you a good idea of the functions of M-XRAY in under half an hour, while at the same time helping you become accustomed to working with M-XRAY.



The steps described in the following sections are the same steps that you will need to follow when performing model analysis of your own models. This chapter therefore not only serves as an introduction to working with M-XRAY, it can also be used as a reference manual when you begin analyzing your own models.



Chapter 5.2 describes an alternative method for calling M-XRAY directly (not via the GUI) via a function call with parameters.

### 3.1 Launching M-XRAY and its Components

After M-XRAY has been added to the search path (or the working directory is the mxray installation directory), launch M-XRAY by typing:

```
>> mxray
```

in the MATLAB command window. This will open the M-XRAY GUI (see Figure 3-1).

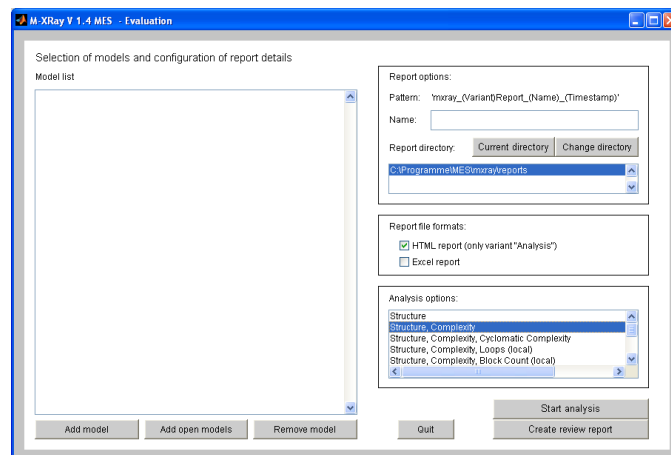


Figure 3-1: M-XRAY GUI after calling `mxray` from the MATLAB command window

The GUI consists of the following items:

- **Model list:** The model list box contains a list of all the models that have been selected for subsequent analysis/review report generation with M-XRAY. By clicking [Add model](#), you can add Simulink mdl-files to the list via a file selection dialog. Alternatively, you can add all open Simulink models by clicking the [Add open models](#) button. Clicking the [Remove model](#) button removes the selected model from the list as does double-clicking a model in the list.
- **Report file name:** M-XRAY automatically chooses suitable file names for HTML/Excel reports. If you wish, however, you can customize the report file name by specifying the name part of the report file name pattern (please see Chapter 5.2.2, p.26 for more details).
- **Report file directory:** M-XRAY saves all report files by default to the directory from which you started M-XRAY. You can select another destination directory by clicking the [Change report directory](#) button.
- **Report file format:** M-XRAY creates an HTML analysis report by default. If you wish, you can switch to Excel or Excel and HTML report generation by checking the appropriate boxes.
- **Analysis options:** In this list box you can specify the level of detail documented in the report file. This selection only applies to creating an analysis report and not a review report (the settings of the latter are predefined).

In the following example we will demonstrate how to operate M-XRAY via its GUI. The MATLAB demonstration model `fuelsys` will be used as demonstration. This example model is part of your MATLAB distribution.

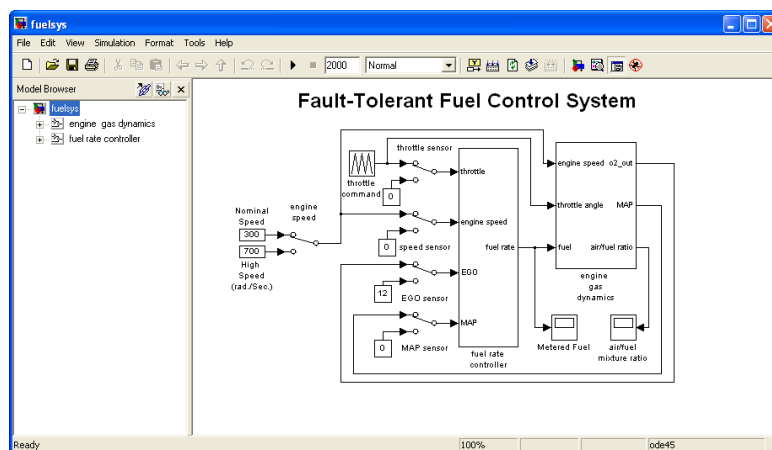


Figure 3-2: Simulink `fuelsys` example model

## 3.2 Conducting your First Analysis with an HTML/Excel Report

### 3.2.1 Selecting a model for analysis

The easiest way of selecting a model for analysis is when the model is already open in MATLAB. We will therefore begin by opening the aforementioned `fuelsys` model by entering the following command in the command window:

```
>> fuelsys
```

Now all you have to do is click the [Add open models](#) button in the GUI and the `fuelsys` model will appear in the model list.

### 3.2.2 Changing name and path of the report file (optional)

The report name field shows the name of the file with the results of analysis. The default setting is always the name of the uppermost entry in the model list, in this case `fuelsys`. You can change this name if you wish by entering a new name in the report name field.

If you want M-XRAY to save the report file in a different directory to the one shown in the report directory field, you can select another folder by clicking [Change directory](#).

If you want to save a report in the current MATLAB directory, click [Current directory](#) to change the report directory to the current one.

### 3.2.3 Generating an HTML analysis report

Now launch analysis by clicking the [Start analysis](#) button at the bottom right (see Figure 3-3).

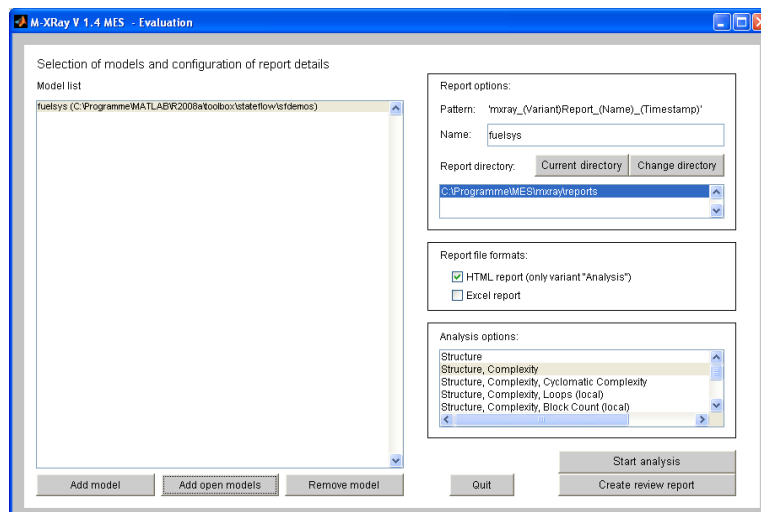


Figure 3-3: Launching analysis in the M-XRAY GUI

While M-XRAY is performing model analysis, a waitbar shows the current progress. When analysis is complete, the MATLAB HTML browser opens to display the generated HTML report (a report

excerpt is shown in Figure 3-4). The report details subsystems contained in the analyzed model, which are sorted according to their local complexity. Please refer to Chapter 8 for more details on how complexity is measured.

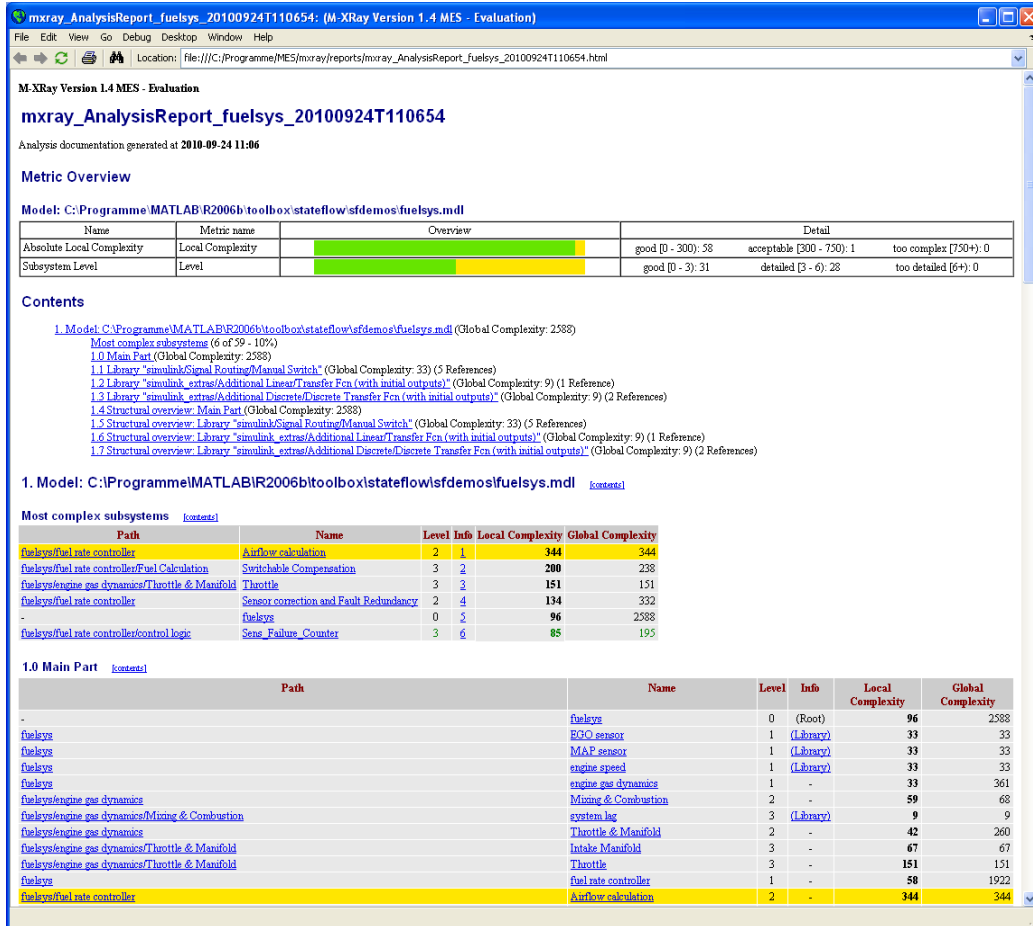


Figure 3-4: Excerpt from generated HTML analysis report in the MATLAB browser

We will now provide a general overview of the analysis report. If you require greater detail, please refer to Chapter 6.

The analysis report begins with general information about the model, such as the name `mrxray_AnalysisReport_fuelsys_20100924T110654`, the date 24.09.2010, and time 11:06.

The metric overview (Figure 3-5) represents the local complexity of subsystems of the `fuelsys` model. The overview displays the percentage share of 'good', 'acceptable', or 'too complex' subsystems. The color-coded representation shows you that `fuelsys` possesses a good level of model complexity as there are no overly complex subsystems. This first impression is confirmed by the detailed overview. In this detailed view, the subsystems shown, their limits, and the number of subsystems assigned to them. Hence 58 'good' subsystems and one with an 'acceptable' complexity are indicated for `fuelsys`.



Figure 3-5: Generated HTML analysis report in the MATLAB browser

The main part of the report (Figure 3-6) shows structural information about the analyzed model. The subsystems with the highest local complexity values from Figure 3-4 are also part of this complete model hierarchy (e.g. 'Airflow calculation' marked in yellow). The lower part of Figure 3-6 contains the 'Control Logic' Stateflow chart and its states (with their respective complexity values marked in green).

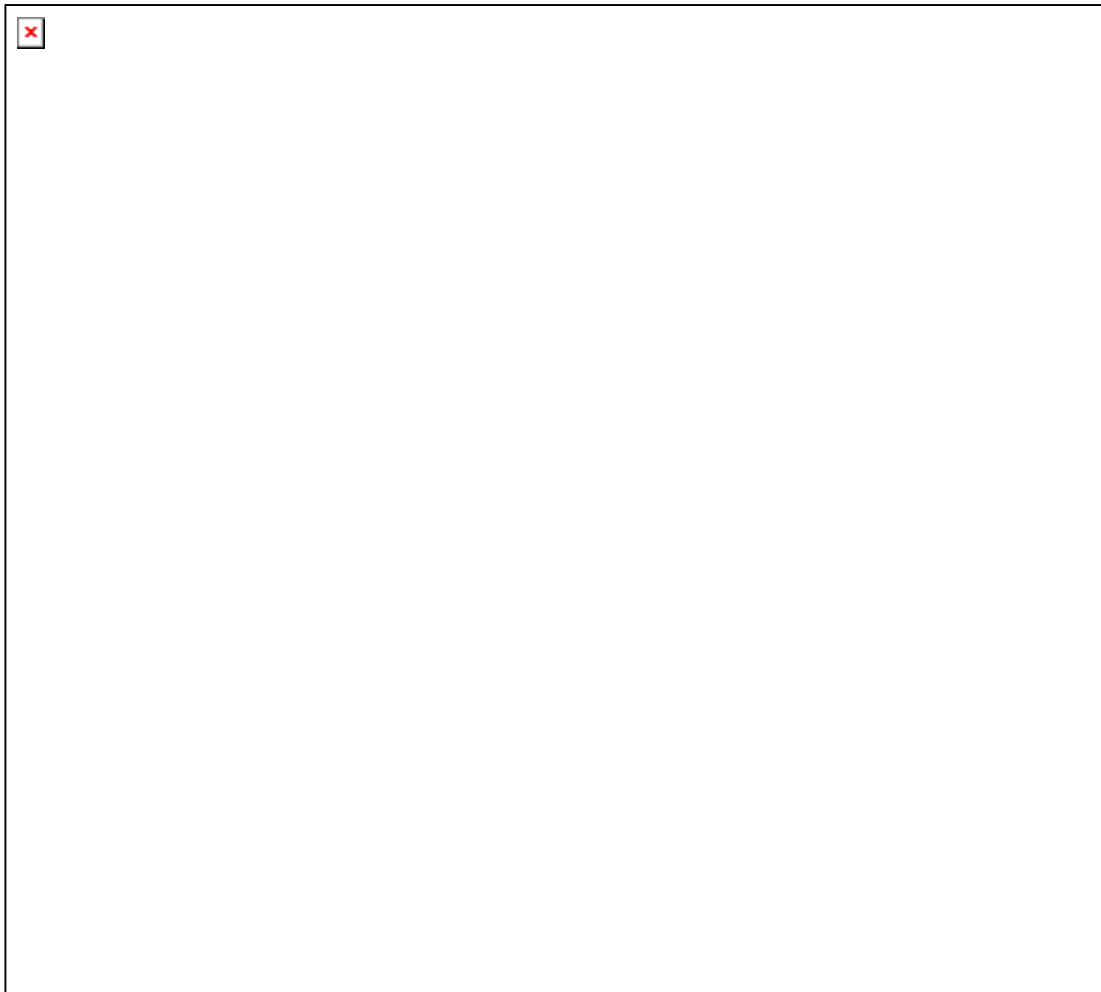


Figure 3-6: Main part of the analysis report



The HTML report contains links to the analyzed subsystems. If you wish to jump straight to a subsystem via one of these links, the analyzed model must already be open. In our example, type `>> fuelsys` in the MATLAB command window to reopen the model.

### 3.2.4 Generating an Excel analysis report



Please note that the Excel report generation feature relies on a local Excel installation.

Return to the M-XRAY GUI and deselect the HTML report checkbox under the report file format options. Instead check Excel report and launch analysis as described above. M-XRAY will perform analysis as before, but this time an Excel report file is generated and opened in Excel. By selecting both checkboxes, it is possible to create both an HTML report and an Excel file simultaneously.

After automatic execution of a formatting macro, the analysis report resembles the one shown in Figure 3-7.

	A	B	C	E	F	G
		Name	Level	Info	Local Complexity	Global Complexity
1	M-XRAY Version 1.4 MES - Evaluation					
2	-	fuelsys	0	(Root)	96	2588
3	fuelsys	EGO sensor	1	(Library)	33	33
4	fuelsys	MAP sensor	1	(Library)	33	33
5	fuelsys	engine speed	1	(Library)	33	33
6	fuelsys	engine gas dynamics	1	-	33	361
7	fuelsys/engine gas dynamics	Mixing & Combustion	2	-	59	68
8	fuelsys/engine gas dynamics/Mixing & Combustion	system lag	3	(Library)	9	9
9	fuelsys/engine gas dynamics	Throttle & Manifold	2	-	42	260
10	fuelsys/engine gas dynamics/Throttle & Manifold	Intake Manifold	3	-	67	67
11	fuelsys/engine gas dynamics/Throttle & Manifold	Throttle	3	-	151	151
12	fuelsys	fuel rate controller	1	-	58	1922
13	fuelsys/fuel rate controller	Airflow calculation	2	-	344	344
14	fuelsys/fuel rate controller	Fuel Calculation	2	-	75	313
15	fuelsys/fuel rate controller/Fuel Calculation	Switchable Compensation	3	-	200	238
16	fuelsys/fuel rate controller/Fuel Calculation/Switchable Comp	LOW Mode	4	-	10	19
17	fuelsys/fuel rate controller/Fuel Calculation/Switchable Comp	Discrete Transfer Fcn (with	5	(Library)	9	9
18	fuelsys/fuel rate controller/Fuel Calculation/Switchable Comp	RICH Mode	4	-	10	19
19	fuelsys/fuel rate controller/Fuel Calculation/Switchable Comp	Discrete Transfer Fcn (with	5	(Library)	9	9
20	fuelsys/fuel rate controller	Sensor correction and Fault	2	-	134	332
21	fuelsys/fuel rate controller/Sensor correction and Fault Redun	MAP Estimate	3	-	66	66
22	fuelsys/fuel rate controller/Sensor correction and Fault Redun	Speed Estimate	3	-	66	66
23	fuelsys/fuel rate controller/Sensor correction and Fault Redun	Throttle Estimate	3	-	66	66
24	fuelsys/fuel rate controller	control logic	2	Chart	5	875
25	fuelsys/fuel rate controller/control logic	Oxygen_Sensor_Mode	3	State	80	110
26	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_fail	4	State	10	10
27	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_warmup	4	State	10	10
28	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_normal	4	State	10	10

Figure 3-7: Excel analysis report

M-XRAY generates an Excel file for `fuelsys` with a worksheet entitled 'fuelsys'. This lists subsystems in black and states in green. Each element is shown with its model path, name, and the subsystem level. The Info column shows whether subsystems from libraries are used or whether you are dealing with a state diagram or state. The last two columns show the calculated local and global complexity of `fuelsys`.

Subsystems and states are color-coded for easier evaluation of the detected complexity. In the HTML report, for example, the subsystem `Airflow calculation` is marked acceptable.

You can find more details about Excel reports in Chapter 6.2.

### 3.3 Generating a Review Report

Return to the M-XRAY GUI and launch review report generation by clicking the [Start Review](#) button. M-XRAY performs model analysis and generates an Excel file containing the extracted model information and a formatting macro. After automatic execution of the macro, the structure and complexity analysis worksheet resembles the one shown in Figure 3-8.

	A	B	C	E	F	G	I	J	K	L
	M-XRay Version 1.4 MES - Evaluation	Subsystem Name	Level	Info	Local Complexity	Global Complexity	Subsystem Review	Quality of Documentation	Requirements fulfilled in functionality	REQ-ids
1										
2		fuelsys	0	[Root]	96	2588	not started	not reviewed		
3	fuelsys	EGO sensor	1	[Library]	33	33	not started	not reviewed		
4	fuelsys	MAP sensor	1	[Library]	33	33	not started	not reviewed		
5	fuelsys	engine speed	1	[Library]	33	33	not started	not reviewed		
6	fuelsys	engine gas dynamics	1	-	33	361	not started	not reviewed		
7	fuelsys/engine gas dynamics	Mixing & Combustion	2	-	59	68	not started	not reviewed		
8	fuelsys/engine gas dynamics/Mixing & Combustion	system lag	3	[Library]	9	9	not started	not reviewed		
9	fuelsys/engine gas dynamics	Throttle & Manifold	2	-	42	260	not started	not reviewed		
10	fuelsys/engine gas dynamics/Throttle & Manifold	Intake Manifold	3	-	67	67	not started	not reviewed		
11	fuelsys/engine gas dynamics/Throttle & Manifold	Throttle	3	-	151	151	not started	not reviewed		
12	fuelsys	fuel rate controller	1	-	58	1922	not started	not reviewed		
13	fuelsys/fuel rate controller	Airflow calculation	2	-	344	344	not started	not reviewed		
14	fuelsys/fuel rate controller	Fuel Calculation	2	-	75	313	not started	not reviewed		
15	fuelsys/fuel rate controller/Fuel Calculation	Switchable Compensation	3	-	200	298	not started	not reviewed		
16	fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation	LOW Mode	4	-	10	19	not started	not reviewed		
17	fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation/LOW Mode	Discrete Transfer Fcn (with Init)	5	[Library]	9	9	not started	not reviewed		
18	fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation	RICH Mode	4	-	10	19	not started	not reviewed		
19	fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation/RICH Mode	Discrete Transfer Fcn (with Init)	5	[Library]	9	9	not started	not reviewed		
20	fuelsys/fuel rate controller	Sensor correction and Fault R	2	-	134	332	not started	not reviewed		
21	fuelsys/fuel rate controller/Sensor correction and Fault Redundancy	MAP Estimate	3	-	66	66	not started	not reviewed		
22	fuelsys/fuel rate controller/Sensor correction and Fault Redundancy	Speed Estimate	3	-	66	66	not started	not reviewed		

Figure 3-8: Excel review report

### 3.4 Selecting Models that are not Open

If you wish to analyze models that are not already open, the following procedure replaces the one described in Chapter 3.2.1. We will once again use the `fuelsys` model as an example. If `fuelsys` is already open, please close the model before you proceed. If necessary, empty the model list in the GUI by deleting the entry `fuelsys` (via [Remove model](#)).

First you need to know the path to the model to be opened. If the current path to the Simulink `fuelsys.mdl` model file is unknown, it can be retrieved by entering

```
>> which('fuelsys')
```

in the MATLAB console.

Now click the [Add model](#) button in the M-XRAY GUI. The file selection dialog will open (see Figure 3-9, left). Navigate to the Stateflow demo directory of your MATLAB distribution (usually `%MATLABROOT%\toolbox\stateflow\sf demos`) and select the `fuelsys.mdl` file.

The model list now shows the entry: `fuelsys`. The remaining steps of analysis are identical to those described in Chapter 3.2.3.



Please note that all models and model parts contained in libraries must be located in the MATLAB search path for analysis to be successful.

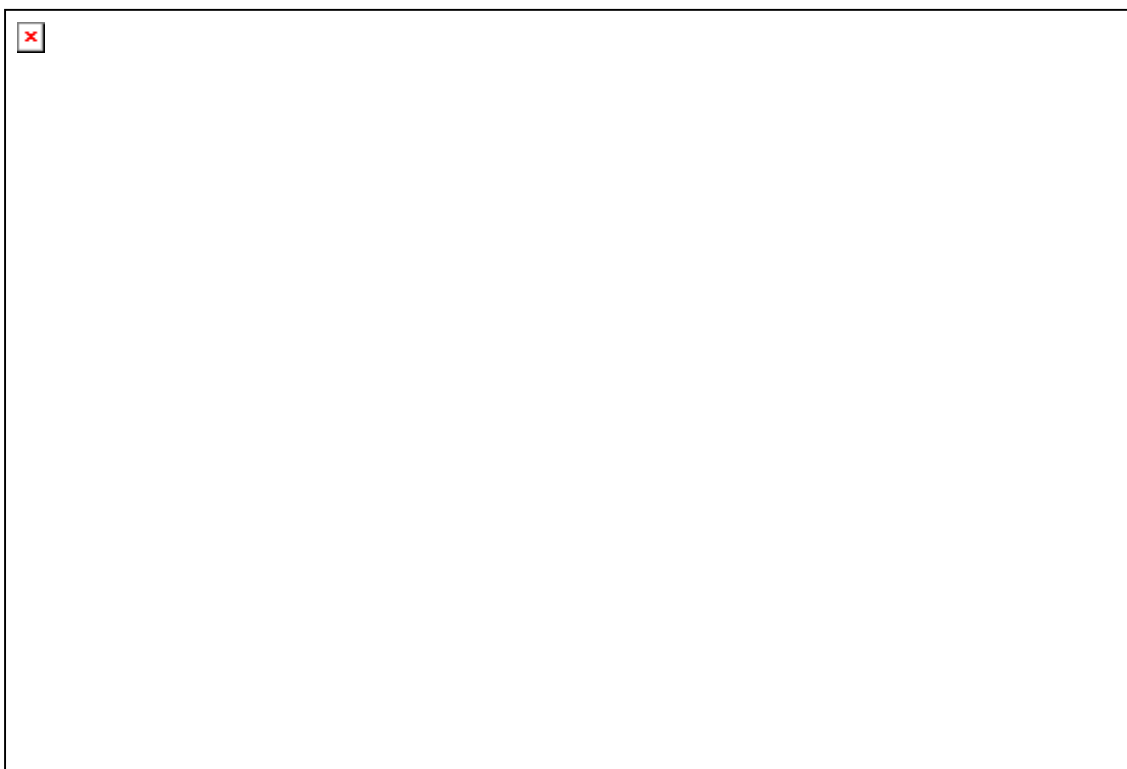


Figure 3-9: Adding the `fuelsys.mdl` model to the model list and launching analysis from the M-XRAY GUI

## 4 Preparing Models for Analysis with M-XRAY

When conducting analysis with M-XRAY, the models under analysis must be fully loadable. In other words, all libraries and referenced models must be accessible via the MATLAB search path.

Whether you permanently add these library paths to the MATLAB search path or choose to only do so for the purposes of working on the model depends entirely on your favored approach. In our projects it has proven good practice to create a small M-file, which adds the necessary paths to the MATLAB search path. If this continues to occur with relative paths, it can be implemented on different computers or used by different users in the same network.



We do not recommend permanently including project paths in the MATLAB path. If you were to do this for all your projects, it would soon lead to a completely overloaded MATLAB path and you would lose track of what the paths are being used for.

### 4.1 Displaying Model Information

As detailed in the preceding section, models must be capable of being loaded, and libraries and model references must be in the MATLAB path. Use the command `mxray_getModelInfo` to determine which libraries and references are needed. This command displays information on size, path, and version number, as well as the necessary libraries and model references. The status as to whether these dependencies are resolved or not is also shown.

We will now explain how to apply the `mxray_getModelInfo` command using the `fuelsys` model supplied by The MathWorks as an example. Information is output on the console and comprises of two areas. Model information gives you various details regarding the model. Below this is an area for libraries and model references. This shows the libraries and model references used and whether they can be accessed.

The following command requests information on `fuelsys`:

```
>> mxray_getModelInfo('fuelsys');
```

```
- Model Information -----
      Name: fuelsys
     FullName:
C:\Programme\MATLAB\R2006b\toolbox\stateflow\sfdemos\fuelsys.mdl
      Size: 180739
   ModelVersion: 1.107
        Version: 6.5
```

```
Created: Tue Jun 02 16:11:43 1998
Creator: The MathWorks Inc.
LastModifiedBy: Axel
ModifiedBy: Axel
- Libraries -----
Library: simulink      (resolved)
Library: simulink_extras (resolved)
-----
```

In addition to standard information, the `fuelsys` model also uses two standard libraries. These can both be accessed as they can be detected via the MATLAB path. The model does not use any model references.

## 5 Working with M-XRAY

The easiest way of operating M-XRAY and configuring your reports is to work via the M-XRAY GUI. Alternatively you have the option of operating M-XRAY by entering your commands in MATLAB's command window.



M-XRAY can be adapted to your own requirements. Please refer to Chapter 9, p.63 for more details.

### 5.1 Executing M-XRAY via the GUI

The Quick Start chapter (Chapter 3) explains how to operate M-XRAY via the GUI.

### 5.2 Executing M-XRAY via the Command Window

Operating M-XRAY via the command window is less comfortable than using the GUI, however it enables a more precise configuration of M-XRAY execution and offers several less frequently needed, extra features.

The directories of the M-XRAY installation directory must first be added to the MATLAB path. This can be achieved as follows (see also Chapter 2.1):

1. Start MATLAB
2. Navigate to the `mxray` folder in your M-XRAY installation directory
3. Right-click on the `mxray_path.m` file and select `Run` from the context menu. This temporarily adds the M-XRAY directories to MATLAB's search path.

The next step is to define the models you wish to analyze. These must also be located in the MATLAB path (including any libraries used with model parts) and may be opened by hand before executing M-XRAY. Models with distinct names (i.e. no multiple models with the same name in the MATLAB path) can also be opened automatically.

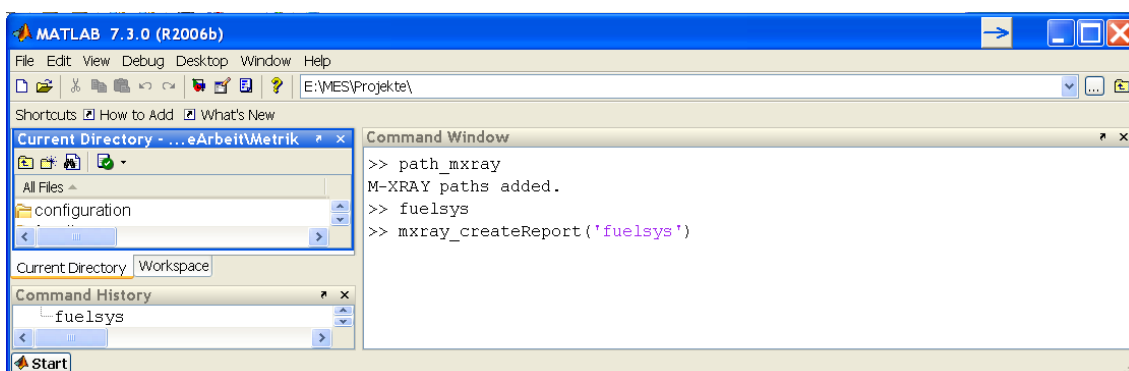


Figure 5-1: MATLAB command window entry when calling M-XRAY (one example)

The `fuelsys` example model used in Chapter 3 is already in the path and can be opened automatically (in Figure 5-1 it is opened explicitly by entering `>> fuelsys`).

The actual execution of M-XRAY's functionality can be controlled by calling `mxray_createReport` with a range of parameters. These parameters allow the standard analysis report to be adapted to include project-specific requirements. The parameters must be written correctly otherwise they cannot be recognized internally and will thus be ignored. Case sensitivity is not requisite.

The following sections describe the different variants of calling `mxray_createReport` with different combinations of parameters.

### 5.2.1 Selecting models for analysis with M-XRAY

Selecting the models to be analyzed always occurs by means of the first parameter of `mxray_createReport`.

- If you only wish to analyze one model, then its name is transferred to the function as a string.

Example of analyzing one model:

```
>> mxray_createReport('fuelsys')
```

- If you wish to analyze multiple models, these can be defined as a cell array of strings.

```
>> mxray_createReport({'fuelsys', 'powerwindow01',  
'powerwindow02'})
```



With the exception of the first parameter, which defines the models under analysis, the sequence of all other parameters is arbitrary. The only exceptions are the parameters 'BT', 'Name', and 'Directory' described below, where the content of the subsequent parameters is of relevance.

### 5.2.2 Configuring analysis reports (1): Defining the execution variant

The parameters discussed below are not required to conduct standard model analysis with M-XRAY. However, they can be used to modify the type of execution in cases where specific

results are required. In some application scenarios, it may make sense to include or exclude (i.e. not display) certain parts of the analysis report. If there is no need to show the distribution of block types in the model, for example, the columns for the block type count become redundant and need not take up a disproportionate amount of space in the report.

- By additionally specifying the `StructOnly` parameter, only the hierarchical structure of models is examined and displayed, without analyzing the content of their subsystems and their complexity.

This also makes execution much faster than in standard analysis. The reduced number of necessary calculations means this variant can only be combined with the following other parameters: `TL`, `Excel`, and `Separate`.

- Example of how to use the `StructOnly` parameter (create a report of the structure of the `fuelsys` model):

```
>> mxray_createReport('fuelsys', 'StructOnly')
```

- By specifying the `Review` parameter, the results of analysis are stored in an Excel file format that has been tailored towards conducting model review. A specific excerpt of the results is displayed in this case. This parameter cannot be combined with any other parameters. The only exception is the `Name` parameter described below, which modifies the naming of the results file.

- Example of how to use the `Review` parameter:

```
>> mxray_createReport('fuelsys', 'Review')
```



The following parameters are not required for most applications. They simply enable particular forms of results presentation and can be added if necessary.

- The results of analysis are stored in HTML files by default. The `NoHTML` parameter serves to suppress this kind of output.
- Example of how to use the `NoHTML` parameter:

```
>> mxray_createReport('fuelsys', 'NoHTML')
```



No differentiation is made between upper and lower case: `nohtml` therefore produces the same results as `NoHTML`.

- By specifying the `Excel` parameter, an Excel file is generated in addition to the default output of results in HTML files:

```
>> mxray_createReport('fuelsys', 'Excel')
```

If you combine the `Excel` parameter with the `NoHTML` parameter as described above, results are only output in an Excel file and not HTML:

```
>> mxray_createReport('fuelsys', 'Excel', 'NoHTML')
```

- For models that contain exactly one `TargetLink` model part, you have the option to only examine this part and to omit all the other parts of the model from the results, as well as the superordinate part of the path. This can be done using the `TL` parameter. If this parameter is used by a purely Simulink model, the report will remain unchanged.

```
>> mxray_createReport('fuelsys_sf6', 'TL')
```

Model results are displayed by default with a sorted list of most complex subsystems (cf. Chapter 8).

- The `NoTop` parameter turns off this display:

```
>> mxray_createReport('fuelsys', 'NoTop')
```

The default display of analysis results is in one HTML file for all models.

- The `Separate` parameter can be used if you wish to output the results per model in separate HTML files. The compilation of particularly complex subsystems is based on the average complexity of the respective model (instead of the average complexity of subsystems of all analyzed models). A content file with an overview of the results and links to the created files is also saved.

```
>> mxray_createReport('fuelsys', 'Separate')
```

The default name given to the results file is that of the first model to be analyzed.

- If you wish to give the results file a different name, then you can do so after the `Name` parameter. This enables you to use a common name for a group of models.

Example of how to use the `Name` parameter:

```
>> mxray_createReport({'fuelsys', 'fuelsys_sf6'}, ...  
                      'Name', 'Fuelsys_SL_and_TL')
```

The default storage location of the results file is the current working directory.

- If you wish to use a different directory, then this can be given after the `Directory` parameter.

Example of how to use the `Directory` parameter:

```
>> mxray_createReport('fuelsys', 'Directory', 'E:\etc\Results')
```

Different weighting is employed for different block types when calculating complexity metrics. As detailed in Chapter 9, these weights can be modified or supplemented as required. The latter case concerns block types that have not yet been included, which will otherwise receive a default weight as standard.

- To find out whether such unknown block types are part of the model, the `Unknown` parameter can additionally be set. This causes the names of unknown block types to be output in the command window. These can then be supplemented in the appropriate file (see Chapter 9) and a weight can be added.

```
>> mxray_createReport('fuelsys', 'Unknown')
```

### 5.2.3 Configuring analysis reports (2): Details of results display

---

You are also free to display additional details of the analysis results in the generated HTML or Excel files. The parameters required in order to do so are described here.

- Specifying the `Interface` parameter displays the number of ingoing and outgoing lines for each subsystem, thereby giving the interface width of the subsystem. The lines are counted separately in the case of buses.

```
>> mxray_createReport('fuelsys', 'Interface')
```

- The strings following the `BT` parameter are interpreted as names of block types and their incidence in the model is recorded in the results. This allows you to display the distribution of particular block types that might interest you. Since all parameters following `BT` are treated as names of block types, we recommend that other parameters being used simultaneously are listed prior to this one.

Example of how to use the `BT` parameter:

```
>> mxray_createReport('fuelsys', 'BT', 'Inport', 'Outport')
```

- Variant: Specifying the `AllBT` parameter means you will display all existing block types instead of a list of individual block types.

Only the local number of block types per subsystem is displayed by default.

- If you wish to also display the global number, i.e. including the frequency of block types in subordinate subsystems, you can do this using the `Global` parameter.

Example of how to combine the `Global` parameter with `AllBT` to output the local and global incidence of all block types:

```
>> mxray_createReport('fuelsys', 'Global', 'AllBT')
```

If you wish to understand how complexity values are calculated, the base values used and other information can also be displayed. Complexity metrics and the significance of base values are discussed in greater detail in Chapter 8.

- With the help of the `AlgoSL` and `AlgoSF` parameters, base values for the metrics for Simulink subsystems or Stateflow objects can be shown.

Example of how to use the `AlgoSL` parameter to show the base values for metrics for Simulink subsystems:

```
>> mxray_createReport('fuelsys', 'AlgoSL')
```

Example of how to use the `AlgoSF` parameter to show the base values for metrics for Stateflow objects:

```
>> mxray_createReport('fuelsys', 'AlgoSF')
```

Example of how to use both parameters:

```
>> mxray_createReport('fuelsys', 'AlgoSL', 'AlgoSF')
```

- The `AllColumns` parameter can be used to precipitate the output of **all available** results. It summarizes the effects of all the parameters described in this subsection under one function.

Example of how to use the `AllColumns` parameter:

```
>> mxray_createReport('fuelsys', 'AllColumns')
```

- The `AllColumnsNoAlgoGlobal` parameter generates the same output as `AllColumns`, only without the metric base values (see `AlgoSL` and `AlgoSF`).

Example of how to use the `AllColumnsNoAlgoGlobal` parameter:

```
>> mxray_createReport('fuelsys', 'AllColumnsNoAlgoGlobal')
```



If not otherwise stated, all parameters are freely combinable in any given sequence. Example (see Figure 5-2):

```
>> mxray_createReport('fuelsys', 'Excel', 'AllBT', NoHtml)
does the same thing as
>> mxray_createReport('fuelsys', 'Excel', 'AllBT', 'NoHtml')
creating only an excel report with all block types.
```

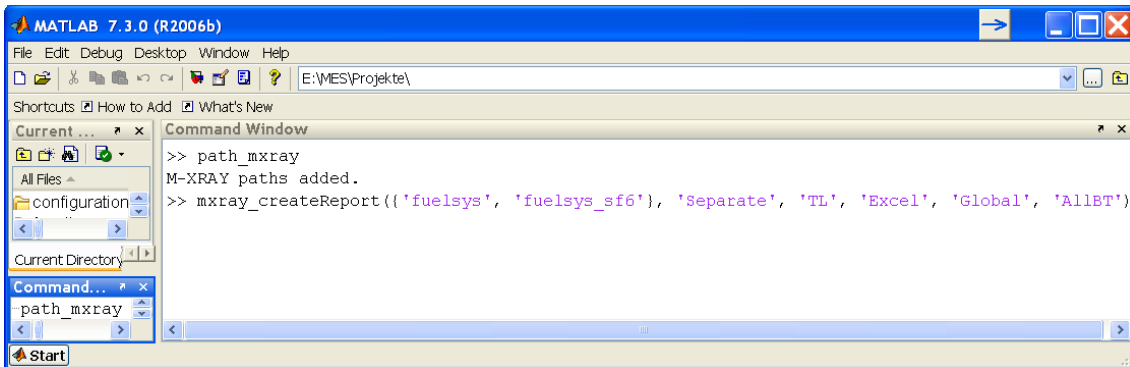


Figure 5-2: Entry in MATLAB's command window when calling M-XRAY with multiple parameters (example)

### 5.3 Exporting Metrics

If you wish to use metrics in more than just an HTML or Excel report, you can export them using the command `mxray_metricsExport`. This command conducts analysis of the specified model and exports the metrics as an XML or CSV file.

In the following example `fuelsys` is analyzed and the local complexity is exported as an XML file entitled `fuelsys.xml`.

```
>> mxray_metricsExport('fuelsys', {'Local Complexity'}, 'xml')
```

```
<!-- This document was automatic generated by MXRAY -->
<model xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mxray_metrics.xsd">
<name>fuelsys</name>
<subsystem>
<name>fuelsys</name>
<path>-</path>
<metric>
<name>Local Complexity</name>
<value>96</value>
</metric>
</subsystem>
<subsystem>
<name>EGO sensor</name>
<path>fuelsys</path>
<metric>
<name>Local Complexity</name>
<value>33</value>
</metric>
</subsystem>
```

The following command conducts an analysis of `fuelsys` and exports both the local and global complexity as the CSV file `fuelsys.csv`. The file is saved in the `e:\tmp` directory.

```
>> mxray_metricsExport('fuelsys', {'Local Complexity',...  
    'Global Complexity'}, 'csv', 'Directory', 'e:\tmp');
```

```
Path;Name;Local Complexity  
-;fuelsys;96  
fuelsys;EGO sensor;33  
fuelsys;MAP sensor;33  
fuelsys;engine speed;33  
fuelsys;engine gas dynamics;33  
fuelsys/engine gas dynamics;Mixing & Combustion;59  
fuelsys/engine gas dynamics;Mixing & Combustion;system lag;9  
fuelsys/engine gas dynamics;Throttle & Manifold;42  
fuelsys/engine gas dynamics;Throttle & Manifold;Intake Manifold;67  
fuelsys/engine gas dynamics;Throttle & Manifold;Throttle;151  
fuelsys;fuel rate controller;58  
fuelsys/fuel rate controller;Airflow calculation;344  
fuelsys/fuel rate controller;Fuel Calculation;75  
fuelsys/fuel rate controller;Fuel Calculation;Switchable Compensation;200
```



## 6 How M-XRAY Presents Results

A report is generated for every M-XRAY execution detailing the results of model analysis. These reports provide comprehensive documentation of the structure of analyzed models and the complexity of their components. Reports are ideal for giving a quick and general overview of the structure of your model, as well as a more detailed description of its individual components. A report offers you the following features:

- Links straight to the respective level in the model (HTML reports only)
- Separate display of libraries with navigation via links
- List of values for local and global complexity
- Color-coded indication of different complexity levels
- Information regarding the width of incoming/outgoing interfaces
- Records the different block types used in the model and shows in which subsystems they can be found

Reports can be generated in (X)HTML format or as an Excel file. The default format of HTML reports is shown in Chapter 6.1; the default format of Excel reports is shown in Chapter 6.2. The significance of the column entries in the reports is identical for both file formats and will be described in detail in Chapter 6.3.

### 6.1 Analysis Report Format (HTML)

#### 6.1.1 Naming and number of results files

---

The results of analysis of a single model are stored in a file under a name that begins with `mxray_AnalysisReport_`, followed by the name of the model, plus the date and time of analysis.

Here, for example, is the name of a results file for the `fuelsys` model that was analyzed on 28.06.2010 at 13:44 am: `mxray_AnalysisReport_fuelsys_20100628T134447.html`

If a number of models are analyzed simultaneously during a single M-XRAY execution, the results are shown in a common file. Alternatively, an individual file can be generated for each model. If M-XRAY is executed via the GUI, only the first option is supported at present; in the case of execution via the command line, both options are available.

If you opt for a common results file, it is named as in the above example and the name of the first model to be analyzed is used. If you prefer, you can call the appropriate function and give the file a more suitable name.

If you opt for separate results files, each of these is named after the respective model. An additional file with a list of contents and links to the different results files is also generated. The name of this contents file is made up of `mxray_AnalysisReport_Content_` plus the date and time. Here is a

complete example of how a content file is named:  
 mxray\_AnalysisReport\_fuelsys\_20100628T134447.html

### 6.1.2 Structure of an HTML report

The analysis report presents the complete information on the structure and content of a model in compact form. The results of analysis are sorted by model and are divided into multiple sections per model. The basic structure of an M-XRAY analysis report is as follows:

- Metric Overview
- Contents
  - 1. Model**
    - Top List** (optional)
    - main part**
    - 1. Library** (if existent)
    - 2. Library** (if existent)
    - (...)
    - Structure Overview Part**
    - 1. Library** (if existent)
    - 2. Library** (if existent)
    - (...)
  - 2. Model** (if existent)
  - (...)

In the case of multiple models, a dedicated report file can also be saved for each model (see Chapter 5.2.2, *Separate* parameter).

Figure 6-1 shows an excerpt from an HTML report, which was created by entering

```
>> mxray_createReport({'fuelsys_sf6','fuelsys'}, 'TL', 'Name',
                      'FuelsysSystems')
```

in the command window. Report structure is explained in the following sections. The significance of the columns with the inclusion of different amounts of information is described in Chapter 6.3.

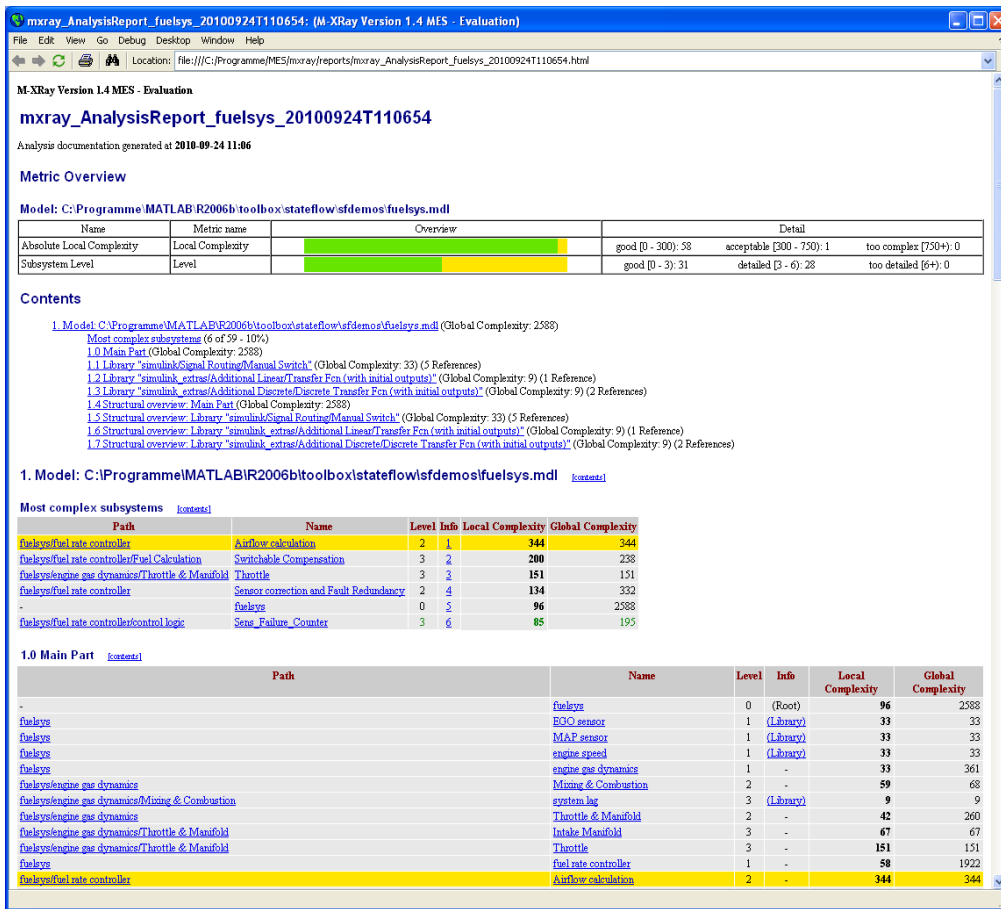


Figure 6-1: HTML report header

### 6.1.3 Metric overview section

The metric overview (Figure 6-2) gives a clear overview of analyzed models. The complexity of each subsystem is evaluated with a classification. If, for example, the classes ‘good’, ‘acceptable’, and ‘too complex’ are used, all subsystems with a local complexity of less than 300 are marked ‘good’ and all subsystems with a value larger than 750 are marked ‘too complex’. The evaluation of each subsystem is collated in one overview.

Metric Overview

Model: C:\Programme\MATLAB\R2006b\toolbox\stateflow\sf\demos\fuelsys.mdl

Name	Metric name	Overview	Detail		
Absolute Local Complexity	Local Complexity		good [0 - 300]: 58	acceptable [300 - 750]: 1	too complex [750+]: 0
Subsystem Level	Level		good [0 - 3]: 31	detailed [3 - 6]: 28	too detailed [6+]: 0

Figure 6-2: Metric overview with color-coded representation and detailed view

The overview is divided into four columns. The Name column shows the classification (Absolute Local Complexity). This is followed by the metric in the column Metric name, for example the local complexity with which the subsystems were evaluated.

The following two columns display the evaluation results. In the Overview column, the proportion of subsystems assigned to the respective classes is color-coded. The color-coding gives you a

quick impression of the complexity of the model under analysis. Taking the example in Figure 6-2, the great majority of subsystems display a 'good' complexity, a small proportion have an 'acceptable' complexity, and there are no subsystems that have been deemed 'too complex'.

The `Detail` column provides greater detail on the classification of subsystems. Class limits are given, as well as the exact number of subsystems assigned to each one. Furthermore, each class is portrayed with a qualitative name (good, acceptable, too complex). These names have been selected to provide a definitive association with a qualitative statement.

### 6.1.4 Contents section

Each report contains a table of contents listing all the subsections of the report. These subsections can be accessed directly by clicking on the links.

The global complexity of every model and every library is always given. If you look at the example in Figure 6-3, the top model displays a total complexity of 2453 and its main part (two lines below) displays the same complexity value.

'Most complex subsystems' details how many of the model's subsystems are included in the list of particularly complex subsystems. The top model in the example therefore consists of 59 subsystems and 23 of these display at local complexity compared to all subsystems.

The three libraries used in the example are listed next. The first library has a global complexity of 24 and is used 5 times in the whole model.

Use the 'Structural Overview: Main Part' link to jump directly to the representation of `fuelsys` model structure in which the model is shown as a tree structure (see Chapter 6.1.8).

#### Contents

[1. Model: C:\Programme\MATLAB\R2006b\toolbox\stateflow\sf\demos\fuelsys.mdl](#) (Global Complexity: 2453)  
[Most complex subsystems](#) (23 of 59 - 39%)  
[1.0 Main Part](#) (Global Complexity: 2453)  
[1.1 Library "simulink/Signal Routing/Manual Switch"](#) (Global Complexity: 24) (5 References)  
[1.2 Library "simulink\\_extras/Additional Linear/Transfer Fcn \(with initial outputs\)"](#) (Global Complexity: 9) (1 Reference)  
[1.3 Library "simulink\\_extras/Additional Discrete/Discrete Transfer Fcn \(with initial outputs\)"](#) (Global Complexity: 9) (2 References)  
[1.4 Structural overview: Main Part](#) (Global Complexity: 2453)  
[1.5 Structural overview: Library "simulink/Signal Routing/Manual Switch"](#) (Global Complexity: 24) (5 References)  
[1.6 Structural overview: Library "simulink\\_extras/Additional Linear/Transfer Fcn \(with initial outputs\)"](#) (Global Complexity: 9) (1 Reference)  
[1.7 Structural overview: Library "simulink\\_extras/Additional Discrete/Discrete Transfer Fcn \(with initial outputs\)"](#) (Global Complexity: 9) (2 References)

Figure 6-3: Contents of analysis report

### 6.1.5 Most complex subsystems section

If you call M-XRAY selecting the `TopList` feature (the default setting in the GUI), the analysis report will begin with an overview of the model's most complex subsystems. Each line shows a single subsystem (or a chart or state). All subsystems that display at least average local complexity are listed under the `Most complex subsystems` section. With larger models, this tends to be around 20-25% of all existing subsystems. The subsystems are listed according to their local complexity value, starting with the most complex. Their ranking is also given in the `Info` column. An additional link takes you straight to the respective subsystem within the model hierarchy.



Each line lists one subsystem (or a chart or state). A color is assigned to each to show the local complexity of each subsystem. The subsystem with the maximum local complexity value of all analyzed models is shown in bright red. Subsystems with an average local complexity are shown in yellow. All other subsystems with above-average complexity are shown in different shades of orange. All subsystems with less than average local complexity are shown in gray.



In order to better differentiate Stateflow objects from Simulink subsystems, charts and states are displayed in green lettering.

If multiple models are analyzed simultaneously during a single M-XRAY execution and the results are shown in a common file, then the selection criterion here is the average complexity of the subsystems of all analyzed models. If, however, the results are stored in separate files (*Separate* feature, cf. Chapter 5.2.2), then the average complexity of the subsystems of the respective model is used.

#### 1. Model: C:\Programme\MATLAB\R2006b\toolbox\stateflows\demos\fuelsys.mdl [\[contents\]](#)

##### Most complex subsystems [\[contents\]](#)

Path	Name	Level	Info	Local Complexity	Global Complexity
fuelsys/fuel_rate_controller	Airflow calculation	2	1	344	344
fuelsys/fuel_rate_controller/Fuel_Calculation	Switchable Compensation	3	2	200	238
fuelsys/engine_gas_dynamics/Throttle & Manifold	Throttle	3	3	151	151
fuelsys/fuel_rate_controller	Sensor correction and Fault Redundancy	2	4	134	332
.	fuelsys	0	5	96	2453
fuelsys/fuel_rate_controller/control_logic	Sens_Failure_Counter	3	6	85	195
fuelsys/fuel_rate_controller/control_logic/Sens_Failure_Counter	MultiFail	4	7	85	100
fuelsys/fuel_rate_controller/control_logic	Fueling_Mode	3	8	85	275
fuelsys/fuel_rate_controller/control_logic	Oxygen_Sensor_Mode	3	9	80	110
fuelsys/fuel_rate_controller/control_logic	Pressure_Sensor_Mode	3	10	80	100
fuelsys/fuel_rate_controller/control_logic	Throttle_Sensor_Mode	3	11	80	100
fuelsys/fuel_rate_controller	Fuel_Calculation	2	12	75	313
fuelsys/fuel_rate_controller/control_logic	Speed_Sensor_Mode	3	13	70	90
fuelsys/engine_gas_dynamics/Throttle & Manifold	Intake Manifold	3	14	67	67
fuelsys/fuel_rate_controller/Sensor correction and Fault Redundancy	MAP Estimate	3	15	66	66
fuelsys/fuel_rate_controller/Sensor correction and Fault Redundancy	Speed Estimate	3	16	66	66
fuelsys/fuel_rate_controller/Sensor correction and Fault Redundancy	Throttle Estimate	3	17	66	66
fuelsys/fuel_rate_controller/control_logic/Fueling_Mode	Running	4	18	65	120
fuelsys/fuel_rate_controller/control_logic/Fueling_Mode	Fuel_Disabled	4	19	60	70
fuelsys/engine_gas_dynamics	Mixing & Combustion	2	20	59	68
fuelsys	fuel_rate_controller	1	21	58	1928
fuelsys	throttle_command	1	22	44	44
fuelsys/engine_gas_dynamics	Throttle & Manifold	2	23	42	260

Figure 6-4: Most complex subsystem in analysis report

### 6.1.6 Main part section

The main part is the section that shows the hierarchy of the model's subsystems, starting with the top level of the model. This is followed by the subsystems that are directly or indirectly subordinate to this one. The respective place in the model hierarchy is given in the `Path` column. The `Level` column gives the depth of each system in the model hierarchy, starting with 0 for the top-level subsystem of the model.

1.0 Main Part [\[context\]](#)

Path	Name	Level	Info	Local Complexity	Global Complexity
-	fuelsys	0	(Root)	96	2588
fuelsys	EGO sensor	1	(Library)	33	33
fuelsys	MAP sensor	1	(Library)	33	33
fuelsys	engine speed	1	(Library)	33	33
fuelsys	engine gas dynamics	1	-	33	361
fuelsys/engine gas dynamics	Mixing & Combustion	2	-	59	68
fuelsys/engine gas dynamics/Mixing & Combustion	system lag	3	(Library)	9	9
fuelsys/engine gas dynamics	Throttle & Manifold	2	-	42	260
fuelsys/engine gas dynamics/Throttle & Manifold	Intake Manifold	3	-	67	67
fuelsys/engine gas dynamics/Throttle & Manifold	Throttle	3	-	151	151
fuelsys	fuel rate controller	1	-	58	1922
fuelsys/fuel rate controller	Airflow calculation	2	-	344	344
fuelsys/fuel rate controller	Fuel Calculation	2	-	75	313
fuelsys/fuel rate controller/Fuel Calculation	Switchable Compensation	3	-	200	238
fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation	LOW Mode	4	-	10	19
fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation/LOW Mode	Discrete Transfer Fcn (with initial outputs)	5	(Library)	9	9
fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation	RICH Mode	4	-	10	19
fuelsys/fuel rate controller/Fuel Calculation/Switchable Compensation/RICH Mode	Discrete Transfer Fcn (with initial outputs)	5	(Library)	9	9
fuelsys/fuel rate controller	Sensor correction and Fault Redundancy	2	-	134	332
fuelsys/fuel rate controller/Sensor correction and Fault Redundancy	MAP Estimate	3	-	66	66
fuelsys/fuel rate controller/Sensor correction and Fault Redundancy	Speed Estimate	3	-	66	66
fuelsys/fuel rate controller/Sensor correction and Fault Redundancy	Throttle Estimate	3	-	66	66
fuelsys/fuel rate controller	control logic	2	Chart	5	875
fuelsys/fuel rate controller/control logic	Oxygen_Sensor_Mode	3	State	80	110
fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mode	O2_fail	4	State	10	10
fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mode	O2_warmup	4	State	10	10
fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mode	O2_normal	4	State	10	10
fuelsys/fuel rate controller/control logic	Pressure_Sensor_Mode	3	State	80	100
fuelsys/fuel rate controller/control logic/Pressure_Sensor_Mode	press_norm	4	State	10	10
fuelsys/fuel rate controller/control logic/Pressure_Sensor_Mode	press_fail	4	State	10	10
fuelsys/fuel rate controller/control logic	Throttle_Sensor_Mode	3	State	80	100
fuelsys/fuel rate controller/control logic/Throttle_Sensor_Mode	throt_norm	4	State	10	10
fuelsys/fuel rate controller/control logic/Throttle_Sensor_Mode	throt_fail	4	State	10	10
fuelsys/fuel rate controller/control logic	Speed_Sensor_Mode	3	State	70	90
fuelsys/fuel rate controller/control logic/Speed_Sensor_Mode	speed_norm	4	State	10	10
fuelsys/fuel rate controller/control logic/Speed_Sensor_Mode	speed_fail	4	State	10	10
fuelsys/fuel rate controller/control logic	Sens_Failure_Counter	3	State	85	195
fuelsys/fuel rate controller/control logic/Sens_Failure_Counter	MultiFail	4	State	85	100
fuelsys/fuel rate controller/control logic/Sens_Failure_Counter/MultiFail	FL3	5	State	5	5
fuelsys/fuel rate controller/control logic/Sens_Failure_Counter/MultiFail	FL4	5	State	5	5
fuelsys/fuel rate controller/control logic/Sens_Failure_Counter/MultiFail	FL2	5	State	5	5
fuelsys/fuel rate controller/control logic/Sens_Failure_Counter	FL1	4	State	5	5

Figure 6-5: Main part of analysis report

### 6.1.7 Library section

If libraries which are included in the display of subsystem hierarchy are used in the model under analysis, then their top subsystem will be marked as a `Library` in the `Info` column of the main part section. The actual library itself including all its subordinate subsystems can be found in a dedicated section that follows the main part. The advantage of this separate means of display is that libraries that are referenced at multiple points in the model only have to be shown once in their complete entirety. A link to the library section is all that is required at every point at which they are referenced. The `Library` entry in the `Info` column is also a link that takes you straight to the respective library.

At the beginning of the library section, there is a list of all subsystems from which the library is referenced under the `Referenced from` heading. Links take you straight to these places. The actual description of the library itself follows and begins with its top subsystem, which is designated by the entry `Root` in the `Info` column. This is followed by the subsystems that lie below. These may link to further libraries that lie at an even deeper level.

### 6.1.8 Structural overview section

The structural overview displays the models as a tree. Each element contains its name, its complexity, and its subsystems as a subtree. This representation simplifies recognition of model structure.

The subsystems assigned to the classes ‘acceptable’ and ‘too complex’ are additionally color-coded. This makes the relationship between model structure and model quality simple to grasp and problematic model structures can be easily identified.

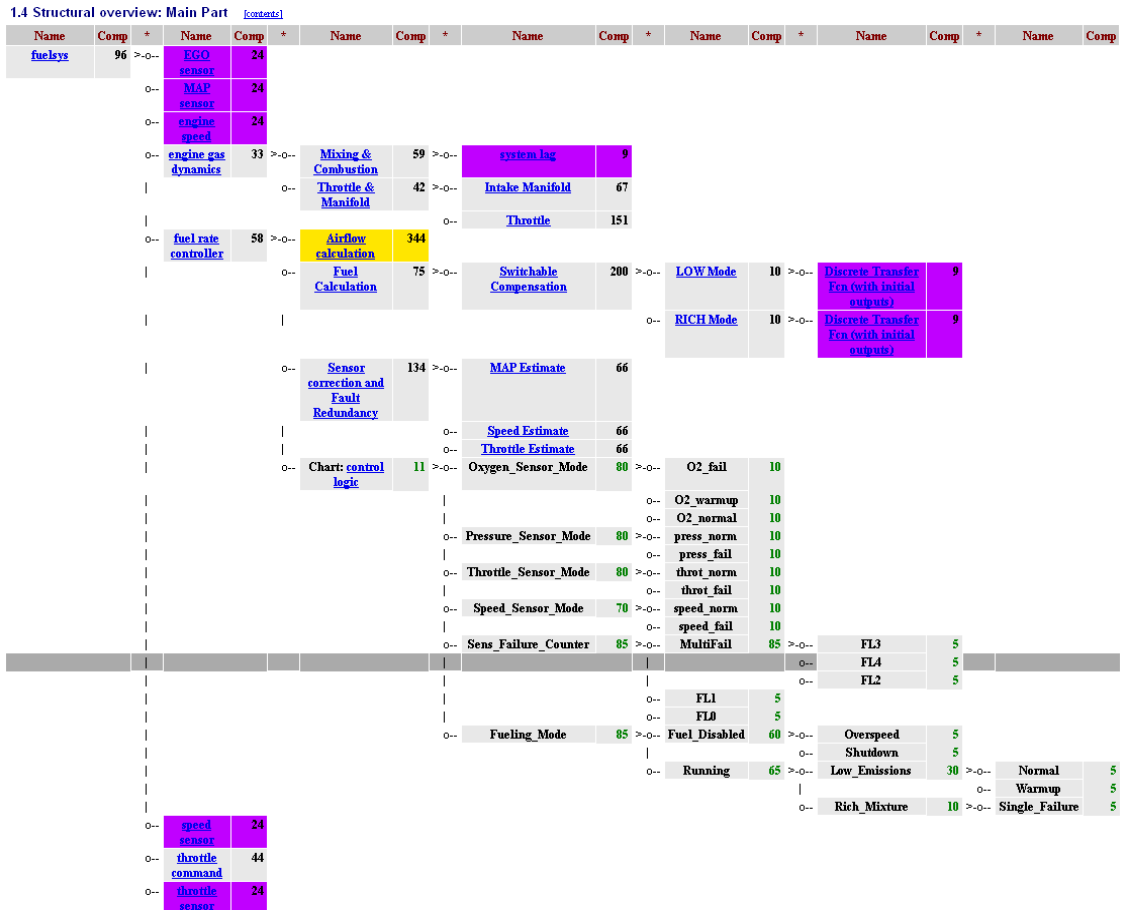


Figure 6-6: Structural overview of analysis report

## 6.2 Analysis Report Format (Excel)

### 6.2.1 Naming of the results file

When you save the results of analysis as an Excel file, they are always saved in a common file irrespective of the number of models. A dedicated table is created for every model within the file.

The name of the Excel file begins with `mxray_AnalysisResult_`, followed by the name of the first model to be analyzed, plus the date and time of analysis. Instead of using the name of the first model, the function can also be given a more suitable name when called.

Example of naming a results file for the `fuelsys` and `sldemo_braketest` models that were analyzed on 28.06.2010 at 13:34 am:

```
mxray_AnalysisReport_fuelsys_20100628T133400.xls
```

Example of naming a results file where multiple models were analyzed on 28.06.2010 at 13:34 am and M-XRAY was given the name 'Examples' to use:

```
mxray_AnalysisReport_Examples_20100628T133400.xls
```

### 6.2.2 Template file and post-processing using macros

A template file is available to ensure clear display of results. This file is called `MXRAY_AnalysisReport_Template.xls` and can be found in the M-XRAY installation directory (see Chapter 10.3). At the end of analysis, a copy of this file is automatically created in the target directory in which the results are saved.



If a file with the name of the template file is found in the current working directory, then this will be used in place of the file in the installation directory. This enables you to use a specially adapted template file for generating reports.



A macro that helps post-processing the data is also part of this file. It is executed automatically and ensures that entries are properly ordered, different links are set within the table, and color-coding is employed to improve the overall layout.

### 6.2.3 Structure of the tables

A dedicated table is created for each model under its name. The structure of these tables basically corresponds with the results display in HTML files as described in Chapters 6.1.6 and 6.1.7. Direct links to the models are not possible in Excel format (unlike in HTML).

## 6.3 Explanation of Columns in Results Display

When results of analysis are displayed, information concerning the subsystems is divided into different columns. The number of columns and thus the scope of information can be selected when calling M-XRAY. The columns can be divided into groups, which will be explained in the following. The order corresponds with that used in the results files, moving from left to right. It is possible to exclude all of these groups of columns from the analysis report except for the first one, which provides information on model structure.

### 6.3.1 Group 1: Columns 'Path', 'Name', 'Info', and 'Level'

		Name	Level	Info
1	M-XRay Version 1.4 MES - Evaluation			
2	-	fuelsys	0	(Root)
3	fuelsys	EGO sensor	1	<a href="#">(Library)</a>
4	fuelsys	MAP sensor	1	<a href="#">(Library)</a>
5	fuelsys	engine speed	1	<a href="#">(Library)</a>
6	fuelsys	engine gas dynamics	1	-
7	fuelsys/engine gas dynamics	Mixing & Combustion	2	-
8	fuelsys/engine gas dynamics/Mixing & Combustion	system lag	3	<a href="#">(Library)</a>
9	fuelsys/engine gas dynamics	Throttle & Manifold	2	-

Figure 6-7: Columns Path, Name, Info, and Level in the results report

These columns contain basic information on the structure of model hierarchy:

- The `Name` column provides the name of the subsystem. HTML reports also contain a link straight to the model, which opens the level of the subsystem involved.
- The `Path` column gives the path of the subsystem in the model hierarchy. HTML reports also contain a link straight to the model, which opens the level above the subsystem involved.



In order to use these links to a model, the model must already be open and the report must be displayed in the MATLAB browser.

- The `Level` column gives the level of the subsystem in the model tree, i.e. the number of superordinate subsystems.
- The `Info` column contains further information on the subsystem: `Root` designates the top subsystem of a model part or a library; reference to a library is marked `Library`; Stateflow objects are designated with the entry `Chart` or `State`.

These columns are always part of the results every time you call M-XRAY. If you select the option `StructOnly` when calling M-XRAY, then these are the only results columns that will be displayed.

### 6.3.2 Group 2: Columns 'Local Complexity' and 'Global Complexity'

		Name	Local Complexity	Global Complexity
1	M-XRay Version 1.4 MES - Evaluation			
2	-	fuelsys	96	2588
3	fuelsys	EGO sensor	33	33
4	fuelsys	MAP sensor	33	33
5	fuelsys	engine speed	33	33
6	fuelsys	engine gas dynamics	33	361
7	fuelsys/engine gas dynamics	Mixing & Combustion	59	68
8	fuelsys/engine gas dynamics/Mixing & Combustion	system lag	9	9
9	fuelsys/engine gas dynamics	Throttle & Manifold	42	260

Figure 6-8: Columns Local Complexity and Global Complexity

These columns give the complexity of the subsystem.

- The 'Local Complexity' column contains the result of complexity metrics for the content of this subsystem.
- The 'Global Complexity' column also contains the accumulated local complexity of all subordinate subsystems.

**6.3.3 Group 3: Columns 'Blocks (local)' and 'Blocks (global)'**

Path	Name	Blocks (local)	Blocks (global)
-	fuelratecontroller	15	236
fuelratecontroller	Airflow calculation	34	34
fuelratecontroller	Fuel Calculation	18	38
fuelratecontroller/Fuel Calculation	Switchable Compensation	20	20
fuelratecontroller	Sensor correction and Fal	35	35
fuelratecontroller	control logic	47	111
fuelratecontroller/control logic	OxygenSensMode	7	7
fuelratecontroller/control logic/OxygenSensMode	O2_Warmup	-	-
fuelratecontroller/control logic/OxygenSensMode	O2_Fail	-	-
fuelratecontroller/control logic/OxygenSensMode	O2_Normal	-	-

Figure 6-9: Columns Blocks (local) and Blocks (global)

These columns contain the number of blocks in the subsystem.

- The 'Blocks (local)' column contains the number of blocks in this subsystem.
- The 'Blocks (global)' column also contains the accumulated number of blocks of all subordinate subsystems.

These columns only appear when you select the Blocks (or AllColumns) option when calling M-XRAY from the command line or when you select a more comprehensive option than Standard when calling via the GUI.

**6.3.4 Group 4: Columns 'Interface In' and 'Interface Out'**

	Name	Interface In	Interface Out
1 M-XRay Version 1.4 MES - Evaluation			
2 -	fuelsys	-	-
3 fuelsys	EGO sensor	2	1
4 fuelsys	MAP sensor	2	1
5 fuelsys	engine speed	2	1
6 fuelsys	engine gas dynamics	3	3
7 fuelsys/engine gas dynamics	Mixing & Combustion	2	2
8 fuelsys/engine gas dynamics/Mixing & Combustion	system lag	1	1
9 fuelsys/engine gas dynamics	Throttle & Manifold	2	2

Figure 6-10: Columns Interface In and Interface Out

These columns give the interface width of the subsystem, i.e. the number of incoming and outgoing lines. In the case of buses, the number of lines of which they consist is what counts. In the case of

bus inports, the tool determines how many of their lines are actually used in the subsystem and only these are counted.

- The `Interface In` column contains the subsystem's input interface.
- The `Interface Out` column contains the subsystem's output interface.

These columns only appear when you select the `Interface` (or `AllColumns`) option when calling M-XRAY from the command line or when you select a more comprehensive option than `Standard` when calling via the GUI.



In the current version of M-XRAY, only the number of inports is shown under 'Interface In' and not the number of lines contained as described here. This is set to change in an upcoming version.

### 6.3.5 Group 5: Columns with base values of Simulink metrics

		N1	N1	N2	N2	N2	N3	N3	N4	N4	N4	N4	
		(org)	(gw)	(org)	(gw)	(lg,gw)	(org)	(gw)	(org)	(gw)	(lg)	(lg,gw)	
1	M-XRay Version 1.4 MES - Evaluation												
2	-	fuelsys	16,0	12,0	19,0	17,4	12,6	3,0	1,8	15,0	13,0	14,0	12,0
3	fuelsys	EGO sensor	6,0	5,0	5,0	6,4	4,8	5,0	4,4	5,0	4,4	5,0	4,4
4	fuelsys	MAP sensor	6,0	5,0	5,0	6,4	4,8	5,0	4,4	5,0	4,4	5,0	4,4
5	fuelsys	engine speed	6,0	5,0	5,0	6,4	4,8	5,0	4,4	5,0	4,4	5,0	4,4
6	fuelsys	engine gas dynamics	8,0	5,6	7,0	5,8	5,0	3,0	2,2	7,0	5,8	6,2	5,0
7	fuelsys/engine gas dynamics	Mixing & Combustion	9,0	8,0	8,0	7,8	7,1	7,0	6,8	7,0	6,8	7,0	6,8
8	fuelsys/engine gas dynamics/Mixing & Combustion	system lag	3,0	2,2	2,0	1,6	1,6	3,0	2,2	2,0	1,6	2,0	1,6
9	fuelsys/engine gas dynamics	Throttle & Manifold	8,0	6,2	8,0	7,4	6,0	5,0	4,0	7,0	6,0	6,6	5,6

Figure 6-11: Columns with base values of Simulink metrics

These columns are not normally required for model analysis. They can, however, help you if you wish to understand how metric values are calculated. These are the metric base values for Simulink systems (here 'org' stands for original, 'gw' for weighted, and 'lg' for 'reduced with log2').

- The `N1 (org)` column represents the total number of blocks in the subsystem. This entry serves as general information only and is not part of the algorithm.
- The `N1 (gw)` column contains a base quantity of the Simulink metric algorithm. Contrary to `N1 (org)`, this number is not simply a summary of the number of blocks, instead a block type-specific weight is added per block.
- The `N2 (org)` column represents the total number of block inputs in the subsystem. This entry serves as general information only and is not part of the algorithm.
- The `N2 (gw)` column corresponds with `N2 (org)`, however the number of inputs per block is multiplied with a block type-specific weight. This entry serves as general information only and is not part of the algorithm.
- The `N2 (lg,gw)` column contains a base quantity of the Simulink metric algorithm. Contrary to `N2 (gw)`, log2 is applied to the number of block inputs before weighting and summation.
- The `N3 (org)` column represents the number of different block types in the subsystem. This entry serves as general information only and is not part of the algorithm.

- The  $N3_{(gw)}$  column contains a base quantity of the Simulink metrics algorithm. Contrary to  $N3_{(org)}$ , this is not simply a summary of the number of block types, instead a block type-specific weight is added per block.
- The  $N4_{(org)}$  column represents the total number of block outputs in the subsystem. This entry serves as general information only and is not part of the algorithm.
- The  $N4_{(gw)}$  column corresponds with  $N4_{(org)}$ , however the number of outputs per block is multiplied with a block type-specific weight. This entry serves as general information only and is not part of the algorithm.
- The  $N4_{(lg)}$  column corresponds with  $N4_{(org)}$ , however  $\log_2$  is applied to the number of block outputs before  $\log_2$  summation. This entry serves as general information only and is not part of the algorithm.
- The  $N4_{(lg,gw)}$  column contains a base quantity of the Simulink metric algorithm. The value corresponds with a combination of weight (as with  $N4_{(gw)}$ ) and logarithm (as with  $N4_{(lg)}$ ).

These columns only appear when you select the `AlgoSL` option when calling M-XRAY.

### 6.3.6 Group 6: Columns with base values of Stateflow metrics

		Name	Chart Comp	States	Cond Act	Tran Act	Events	Entry	During	Exit	On Event	Cond Tran	And	Or
1	M-XRay Version 1.4 MES - Evaluation													
26	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_fail	2	-	-	-	-	1	-	-	-	-	-	-
27	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_warmup	2	-	-	-	-	1	-	-	-	-	-	-
28	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_normal	2	-	-	-	-	1	-	-	-	-	-	-
29	fuelsys/fuel rate controller/control logic	Pressure_Sensor_Mode	16	-	-	2	-	-	-	-	-	2	1	1
30	fuelsys/fuel rate controller/control logic/Pressure_Sensor_Mo	press_norm	2	-	-	-	-	1	-	-	-	-	-	-
31	fuelsys/fuel rate controller/control logic/Pressure_Sensor_Mo	press_fail	2	-	-	-	-	1	-	-	-	-	-	-
32	fuelsys/fuel rate controller/control logic	Throttle_Sensor_Mode	16	-	-	2	-	-	-	-	-	2	1	1
33	fuelsys/fuel rate controller/control logic/Throttle_Sensor_Mod	throt_norm	2	-	-	-	-	1	-	-	-	-	-	-
34	fuelsys/fuel rate controller/control logic/Throttle_Sensor_Mod	throt_fail	2	-	-	-	-	1	-	-	-	-	-	-

Figure 6-12: Columns with base values of Stateflow metrics

These columns are not normally required for model analysis. They can, however, help you if you wish to understand how metric values are calculated. These are the metric base values for Stateflow charts and states.

- `ChartComp` shows complexity before scaling.
- `States` contains the number of states.
- `CondAct` contains the number of actions associated with conditions.
- `TranAct` contains the number of actions associated with transitions.
- `Events` contains the number of events in the transitions.
- `Entry` contains the number of entry actions in the state.
- `During` contains the number of during actions in the state.
- `Exit` contains the number of exit actions in the state.
- `OnEvent` contains the number of OnEvents in the state.
- `CondTran` contains the number of transitions supplied with a condition.
- `And` contains the number of '&' and '&&' in the conditions. Both of these indicate an additional sub-condition.

- `Or` contains the number of '[' and ']' in the conditions. Both of these indicate an additional sub-condition.

These columns only appear when you select the `AlgoSF` (or `AllColumns`) option when calling M-XRAY from the command line or the `All Details` option when calling via the GUI.

### 6.3.7 Group 7: Columns with local number of blocks of different types

		Chart (local)	Constant (local)	Inport (local)	Output (local)	Transition (local)
1	M-XRay Version 1.4 MES - Evaluation					
2	-	-	6	-	-	-
3	fuelsys	-	1	2	1	-
4	fuelsys	-	1	2	1	-
5	fuelsys	-	1	2	1	-
6	fuelsys	-	-	3	3	-
7	fuelsys/engine gas dynamics	-	1	2	2	-
8	fuelsys/engine gas dynamics/Mixing & Combustion	-	-	1	1	-
9	fuelsys/engine gas dynamics	-	1	2	2	-

Figure 6-13: Columns with the local number of block types displayed in two different ways: top, output of local number only; bottom, together with global number (each adjacent; masked here, see Chapter 6.3.8)

These columns show the number of blocks belonging to different block types in the subsystem. We will describe the column for the `Inport` block type to serve as an example; the description also applies to all other block types.

- `Inport` contains the number of blocks in the subsystem with the `Inport` block type.
- `Inport (local)` has the same content and replaces the `Inport` column if both local and global values are being displayed (see following section).

These columns only appear when you select one of the `AllColumns`, `AllBT`, or `BT` options (the latter will only display the block types you require) when calling M-XRAY from the command line, or when you select at least the 'local block count' option when calling via the GUI.



The block type used does not always correspond with the content of the `BlockType` field of the corresponding block column.

On one hand, certain characters are replaced in the string for processing ('-' by '\_', spaces are dropped).

On the other, in the case of `BlockType = 'SubSystem'`, the `MaskType` is used in place of the `BlockType` for better differentiation.

## 6.3.8 Group 8: Columns with global number of blocks of different types

		Name	Chart (global)	Constant (global)	Inport (global)	Outport (global)	Transition (global)
1	M-XRay Version 1.4 MES - Evaluation						
2	-	fuelsys	1	26	47	30	34
3	fuelsys	EGO sensor	-	1	2	1	-
4	fuelsys	MAP sensor	-	1	2	1	-
5	fuelsys	engine speed	-	1	2	1	-
6	fuelsys	engine gas dynamics	-	3	13	11	-
7	fuelsys/engine gas dynamics	Mixing & Combustion	-	1	3	3	-
8	fuelsys/engine gas dynamics/Mixing & Combustion	system lag	-	-	1	1	-
9	fuelsys/engine gas dynamics	Throttle & Manifold	-	2	7	5	-

Figure 6-14: Columns with the global number of block types (local number is always adjacent; masked here)

These columns show the accumulated number of blocks of different block types in the subsystem and all their subordinate subsystems. We will describe the column for the `Inport` block type to serve as an example: the description also applies to all other block types.

- `Inport (global)` contains the number of blocks in this subsystem and all underlying subsystems with the `Inport` block type.

These columns only appear when you select the `AllColumns` option (or the `Global` option in addition to `AllBT` or `BT`) when calling M-XRAY from the command line, or when you select at least the 'global block count' option when calling via the GUI.

## 7 M-XRAY Review Report

Having an overview of the structure of the models you are investigating can be very helpful indeed when conducting model reviews. M-XRAY gives you the option of compiling all relevant information about model structure in an Excel file. The file also contains additional sheets, which can be used when conducting your model review.

### 7.1 Creating a Model Review File

#### 7.1.1 Review file template

---

To create a review file you will require a template file called `MXRAY_ReviewReport_Template.xls`. M-XRAY automatically uses the file from the M-XRAY installation directory (see Chapter 10.3).



If a file with the name of the template file is found in the current working directory, then this is used in place of the file in the installation directory. This enables you to use a specially adapted template file for generating reports.

#### 7.1.2 Naming the review file

---

The name of the review file starts with `mxray_ReviewReport_`, followed by the name of the first model to be analyzed, plus details of the date and time of analysis. Instead of using the name of the first model, the function can also be given a more suitable name when called.

Example of the name of a review file where the models 'fuelsys' and 'sldemo\_braketest' were analyzed on 28.06.2010 at 13:34 am:

```
mxray_ReviewReport_fuelsys_20100628T133400.xls
```

Example of the name of a review file where the models were analyzed on 28.06.2010 at 11:40 am and M-XRAY was given the name 'Examples' to use:

```
mxray_ReviewReport_Examples_20100628T133400.xls
```

### 7.2 Tables in the Review File

This section will explain the function of the tables in the review file.

7.2.1 Summary table

Findings						
	trivial	minor	major	critical	blocker	total
Occurrences	1	2	0	0	0	3
Fraction rejected or duplicate	0	0	0	0	0	0
Remaining	1	2	0	0	0	3

Figure 7-1: Summary table in the review report

MXRAY\_FuelsysReviewReport The entries in the Summary table contain information on the analyzed models and a summary of the results of the model review. This includes:

- Object under investigation and date of review
- Information on MATLAB and TargetLink versions used plus revision number
- Information on review objectives
- Information on modules
- Overview of the number of issues of differing severity detected in the review. This section is generated automatically using the entries in the Issue Tracking table.

7.2.2 Issue tracking table

The Issue Tracking table describes the issues detected during review. Issues can be entered into the rows of the table. The columns are used to enter various pieces of information during the review and subsequent correction phases.

This includes:

- **Product:** Name of model group
- **Component:** Name of module in which the issue was detected
- **Type:** defect, enhancement, task, documentation, testcase
- **Status:** new, assigned, fixed, reviewed, reopened, postponed, duplicated, rejected
- **Priority:** trivial, minor, major, critical, blocker
- **Path:** Path of block or subsystem where the issue appears
- **Keywords:** Consistency, Understandability, Editorial Remark, Formal Aspect, Correctness, Optimization, Model Review
- **Object Text:** Verbal description of issue
- **SVN Revision:** Revision number of the model reviewed
- **reported by:** Name of reviewer
- **Date found:** Date of entry
- **fix comment:** Comment on correction of issue
- **responsible:** Corrector
- **SVN Revision fixed:** Revision number of the model where the issue was fixed
- **Date fixed:** Date of correction
- **Resolution:** fixed, no\_need\_for\_action, duplicate, postponed

	B	C	D	E	F	G	H	I	K	L
	Product	Component	Type	Status	Priority	Path	Keywords	Object Text	reported by	Date found
1										
2	fuelsys	fuel rate controller	defect	new	minor		Correctness, Modell-Review	Description ...	MES	24.09.2010
3	fuelsys	engine gas dynamics	enhancement	new	trivial		Understandability, Modell-Review	Description ...	MES	24.09.2010
4	fuelsys	engine gas dynamics	documentation	new	minor		Editorial remark, Modell-Review	Description ...	MES	24.09.2010
5										
6										
7										
8										
9										

Figure 7-2: Issue tracking table in the review report

### 7.2.3 Tables providing an overview of model structure

A dedicated table is saved for each model under its name. This contains an overview of the structure with the information subset of default analysis with M-XRAY. These are the columns Path, Name, Info, Local Complexity, and Global Complexity, the significance of which is explained in Chapters 6.3.1 and 6.3.2.

	A	B	C	E	F	G	I	J	K	L
	M-XRAY Version 1.4 MES - Evaluation	Subsystem Name	Level	Info	Local Complexity	Global Complexity	Subsystem Review	Quality of Documentation	Requirements fulfilled in functionality	REQ-Ids
1										
2	-	fuelsys	0	(Root)	96	2588	not started	not reviewed		
3	fuelsys	EGO sensor	1	(Library)	33	33	not started	not reviewed		
4	fuelsys	MAP sensor	1	(Library)	33	33	not started	not reviewed		
5	fuelsys	engine speed	1	(Library)	33	33	not started	not reviewed		
6	fuelsys	engine gas dynamics	1	-	33	361	not started	not reviewed		
7	fuelsys/engine gas dynamics	Mixing & Combustion	2	-	59	68	not started	not reviewed		
8	fuelsys/engine gas dynamics/Mixing & Combustion	system lag	3	(Library)	9	9	not started	not reviewed		
9	fuelsys/engine gas dynamics	Throttle & Manifold	2	-	42	260	not started	not reviewed		
10	fuelsys/engine gas dynamics/Throttle & Manifold	Intake Manifold	3	-	67	67	not started	not reviewed		
11	fuelsys/engine gas dynamics/Throttle & Manifold	Throttle	3	-	151	151	not started	not reviewed		
12	fuelsys	fuel rate controller	1	-	58	1922	not started	not reviewed		
13	fuelsys/fuel rate controller	Airflow calculation	2	-	344	344	not started	not reviewed		
14	fuelsys/fuel rate controller	Fuel Calculation	2	-	75	313	not started	not reviewed		
15	fuelsys/fuel rate controller/Fuel Calculation	Switchable Compensation	3	-	200	238	not started	not reviewed		
16	fuelsys/fuel rate controller/Fuel Calculation/Switchable Com	LOW Mode	4	-	10	19	not started	not reviewed		
17	fuelsys/fuel rate controller/Fuel Calculation/Switchable Com	Discrete Transfer Fcn (with	5	(Library)	9	9	not started	not reviewed		
18	fuelsys/fuel rate controller/Fuel Calculation/Switchable Com	RICH Mode	4	-	10	19	not started	not reviewed		
19	fuelsys/fuel rate controller/Fuel Calculation/Switchable Com	Discrete Transfer Fcn (with	5	(Library)	9	9	not started	not reviewed		
20	fuelsys/fuel rate controller	Sensor correction and Fau	2	-	134	332	not started	not reviewed		
21	fuelsys/fuel rate controller/Sensor correction and Fault Redu	MAP Estimate	3	-	66	66	not started	not reviewed		
22	fuelsys/fuel rate controller/Sensor correction and Fault Redu	Speed Estimate	3	-	66	66	not started	not reviewed		
23	fuelsys/fuel rate controller/Sensor correction and Fault Redu	Throttle Estimate	3	-	66	66	not started	not reviewed		
24	fuelsys/fuel rate controller	control logic	2	Chart	5	875	not started	not reviewed		
25	fuelsys/fuel rate controller/control logic	Oxygen_Sensor_Mode	3	State	80	110	not started	not reviewed		
26	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_fail	4	State	10	10	not started	not reviewed		
27	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_warmup	4	State	10	10	not started	not reviewed		
28	fuelsys/fuel rate controller/control logic/Oxygen_Sensor_Mod	O2_normal	4	State	10	10	not started	not reviewed		
29	fuelsys/fuel rate controller/control logic	Pressure_Sensor_Mode	3	State	80	100	not started	not reviewed		
30	fuelsys/fuel rate controller/control logic/Pressure_Sensor_M	press_norm	4	State	10	10	not started	not reviewed		
31	fuelsys/fuel rate controller/control logic/Pressure_Sensor_M	press_fail	4	State	10	10	not started	not reviewed		
32	fuelsys/fuel rate controller/control logic	Throttle_Sensor_Mode	3	State	80	100	not started	not reviewed		
33	fuelsys/fuel rate controller/control logic/Throttle_Sensor_M	throt_norm	4	State	10	10	not started	not reviewed		
34	fuelsys/fuel rate controller/control logic/Throttle_Sensor_M	throt_fail	4	State	10	10	not started	not reviewed		
35	fuelsys/fuel rate controller/control logic	Speed_Sensor_Mode	3	State	70	90	not started	not reviewed		
36	fuelsys/fuel rate controller/control logic/Speed_Sensor_Mod	speed_norm	4	State	10	10	not started	not reviewed		
37	fuelsys/fuel rate controller/control logic/Speed_Sensor_Mod	speed_fail	4	State	10	10	not started	not reviewed		

Figure 7-3: Table with model structure (the model here is fuelsys) in the review report

The following additional columns provide comments on the review:

- **Subsystem Review:** not started, in progress, finished, aborted, ignore
- **Requirements fulfilled in functionality:** n.a., unclear, partly, complete, unsettled
- **DOORS REQ-Ids:** Here you can enter the requirements IDs that are implemented in the respective subsystems. Non-fulfilled requirements can be highlighted here. This provides an overview of where issues can be found in the model.
- **Review Time:** Here you can enter the time taken to conduct the review of this subsystem.

#### 7.2.4 Maintenance table

---

This table serves to configure the entries in the selection lists of the `Issue Tracking` table and model structure tables. It is usually hidden. A right mouse-click on a table tab at the bottom left (e.g. on `Summary`) and subsequent selection of `Unhide` or `Hide` lets you edit the table.

#### 7.2.5 Template table

---

This table is used by the macro and subsequently deleted.



## 8 Measuring Model Complexity

One of the features of M-XRAY is calculating metrics to measure the complexity of individual subsystems, model parts, and complete models. The given value for the total complexity of a model can vary from between 1,000 and 4,000 for smaller models (e.g. the `fuelsys` model used in Chapter 3), between 15,000 and 40,000 for medium-sized models, up to values of 80,000 and more for very complex models.

This chapter provides a brief introduction to the topic of metrics and describes the algorithms used by M-XRAY.



A detailed understanding of these algorithms is not necessary to work with M-XRAY. However it is helpful knowledge should you wish to configure the weighting used in these algorithms.

### 8.1 Goals of Complexity Measurement

A software metric reproduces a software unit (such as a function or module) as a numerical value. Depending on the metric you use, this numerical value can, for example, evaluate the complexity of the software unit. This results in a simple way of comparing the complexity of different software systems or different parts of a software. You can, for example, check whether the total complexity is evenly distributed across the modules or whether it is overly concentrated in certain parts. It therefore becomes possible to estimate how time-consuming it will be to conduct a review before you begin, based on experience with software systems of a similar complexity. It is even possible to track the development of complexity through multiple versions of the same software.

### 8.2 Software Metrics Model

A whole range of metrics exist for calculating the complexity of source code (e.g. C code). The most well-known of these include:

- Lines of Code
- Cyclomatic complexity (after McCabe)
- Halstead metrics



No generalized metric for calculating the complexity of Simulink/TargetLink models exists so far.

M-XRAY uses the Halstead metric as the basis upon which to calculate the complexity of Simulink/TargetLink models. This is a tried and tested procedure for calculating the complexity of source code (such as C or Java). Different versions of the Halstead metric exist with different emphases.

M-XRAY is based on the Halstead volume metric. Its formula is as follows:

$$(N1 + N2) * \log_2 (N3 + N4)$$

This formula contains the following base quantities:

- **N1** : number of operators (commands, functions, etc.)
- **N2** : number of operands (parameters, return values, etc.)
- **N3** : number of different operators
- **N4** : number of different operands

The values refer to the elements of the software unit for which the metric is calculated.

## 8.3 Calculating Model Complexity (Simulink)

### 8.3.1 Application to SL/TL systems

Simulink and TargetLink systems are treated equally in analysis with M-XRAY.

Applying the Halstead metric to Simulink/TargetLink models requires a few adaptations to be made.

Ausgehend davon, dass Operatoren Anweisungen sind, die Daten durch Berechnungsvorschriften ändern, werden Simulink/TargetLink blocks als operators verwendet. Damit lassen sich die Parameter N1 sowie N3 bestimmt werden. Operanden stellen in der Regel Variablen bzw. Konstanten also Daten dar. Simulink/Targetlink Blöcke erhalten ihre Daten durch Eingänge und geben ihre Berechnungen an ihre Ausgänge weiter. Deshalb werden die Ein- bzw. Ausgänge Als Operanden verwendet. Um den Parameter N2 zu ermitteln wird die Anzahl der Eingänge als Operanden verwendet. Da für N4 die Anzahl unterschiedlicher Operanden bestimmt werden muss, werden hier nur die Blockausgänge verwendet.

As a result, the base values of the metric for calculating the complexity of Simulink subsystems are as follows:

- **N1** : number of blocks
- **N2** : number of block inputs
- **N3** : number of different block types
- **N4** : number of block outputs

The values all refer to the subsystem under analysis.

The formula is as follows:  $(N1 + N2) * \log_2 (N3 + N4 + 1)$ .

In variance to the original Halstead formula, M-XRAY uses additional weights for the different block types. These weights are incorporated into all components of the formula (**N1**, **N2**, **N3**, **N4**). This addition takes the differing complexity of the individual block types into account. Each block type has its own weight value in M-XRAY. These values can also be modified, if required, or expanded

to include weights for additional block types. Unknown block types that have not been assigned a weight are evaluated using a default weight.

Another deviation from the original Halstead metric is the transformation of the number of inputs and outputs of a block with  $\log_2$ . As a result, the contribution made by the inputs does not increase linearly with their increasing number.

Example: block with 7 inputs  $\rightarrow \log_2(7 + 1) = 3$

Trigger/enable ports play a special role. They are not included in the transformation with  $\log_2$ .

Example: block with 7 inputs and 1 trigger port  $\rightarrow \log_2(7 + 1) + 1 = 4$

### 8.3.2 Base quantities of the metric

Explanation of base quantities of the metric:

- **N1** = sum of weights of the individual blocks  
Complexity increases with the number of blocks in a subsystem.
- **N2** = sum of block inputs, each reduced with  $\log_2$  and subsequently weighted according to block type  
This value takes into account any links between the blocks. Blocks with many inputs increase complexity more strongly, though the use of  $\log_2$  means this is not a linear increase.
- **N3** = sum of weights of block types (once per block type)  
Subsystems with many different block types are more complex.
- **N4** = sum of block outputs, each reduced with  $\log_2$  and subsequently weighted according to block type  
Block outputs are evaluated separately from block inputs as their number can differ (branching lines).

### 8.3.3 Example of a Simulink metric calculation

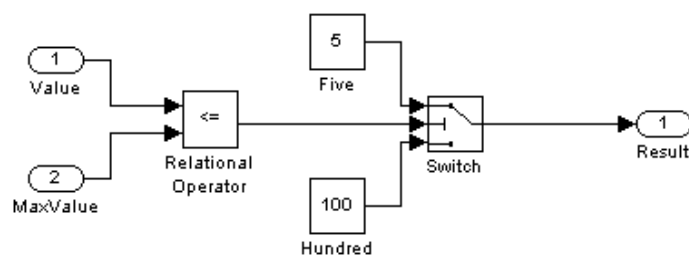


Figure 8-1: Example subsystem

Figure 8-1 shows an example subsystem consisting of seven blocks (2 inputs, 1 relational operator, 2 constants, 1 switch, 1 output). Table 8-1 shows the results for the associated base values, initially unweighted and without logarithmic scaling:

Table 8-1: Base values for the example subsystem without weighting and log2 scaling

Base value	Inport value	Inport MaxValue	Relational operator	Constant five	Constant hundred	Switch	Output result	Result
<b>N1</b>	1	1	1	1	1	1	1	<b>7</b>
<b>N2</b>	0	0	2	0	0	3	1	<b>6</b>
<b>N3</b>	1		1	1		1	1	<b>5</b>
<b>N4</b>	1	1	1	1	1	1	0	<b>6</b>

If the aforementioned logarithmic scaling is then applied to the base values N2 and N4 to adapt larger values, the only thing to change (in this example) are the values of two cells (those shown in bold). The results in Table 8-2 are as follows:

Table 8-2: Base values for the example subsystem with log2 scaling of certain values

Base value	Inport value	Inport MaxValue	Relational operator	Constant five	Constant hundred	Switch	Output result	Result
<b>N1</b>	1	1	1	1	1	1	1	<b>7</b>
<b>N2</b>	0	0	<b>1.6</b>	0	0	<b>2</b>	1	<b>4.6</b>
<b>N3</b>	1		1	1		1	1	<b>5</b>
<b>N4</b>	1	1	1	1	1	1	0	<b>6</b>

Table 8-3 shows the base values plus (fictitious) weight values:

Table 8-3: Base values for the example subsystem with weighting and log2 scaling

Weight	<b>0.8</b>	<b>0.8</b>	<b>1.5</b>	<b>0.5</b>	<b>0.5</b>	<b>2</b>	<b>0.8</b>	
Base value	Inport value	Inport MaxValue	Relational operator	Constant five	Constant hundred	Switch	Output result	Result
<b>N1</b>	0.8	0.8	1.5	0.5	0.5	2	0.8	<b>6.9</b>
<b>N2</b>	0	0	2.4	0	0	4	0.8	<b>7.2</b>
<b>N3</b>	0.8		1.5	0.5		2	0.8	<b>5.6</b>
<b>N4</b>	0.8	0.8	1.5	0.5	0.5	2	0	<b>6.1</b>

Complexity is calculated for the subsystem in Figure 8-1 using the figures from Table 8-3 as follows:  
 $(N1 + N2) * \log_2(N3 + N4 + 1) = (6.9 + 7.2) * \log_2(5.6 + 6.1 + 1) = 51.7$

## 8.4 Calculating the Complexity of Loops

Loops (feedback loop, closed loop) inside a model increase the complexity of the modeled function. Thus, M-XRAY considers loop complexity too.

Loop complexity is added to model complexity as follows:

$$(\text{Local}) \text{ complexity} = (\text{local}) \text{ model complexity} + (\text{local}) \text{ loop complexity.}$$



Considering loop complexity separately changes the local and global complexity of subsystems.

M-XRAY considers loops belonging to one subsystem (local loops). If a subsystem is located within a loop, it is viewed as a single block. The internal structure of this subsystem is not considered in the calculation.

Figure 8-2 shows an example with three loops, which describes how to determine the complexity of a loop.

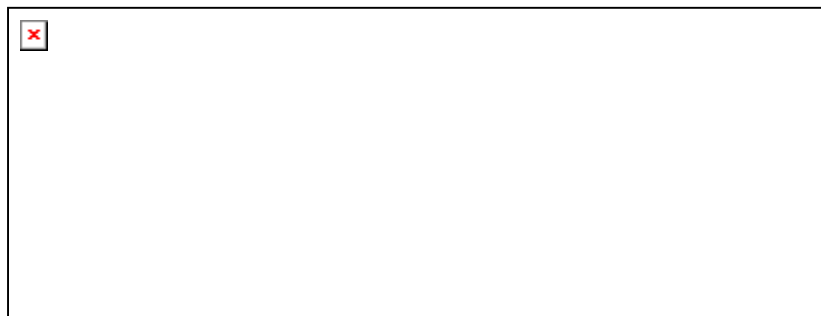


Figure 8-2: Example of a loop complexity calculation

The blocks contained in each loop are determined. The blocks are then weighted and added together. The three loops shown in Figure 8-2 produce the following values:

Loop	Number of blocks	Complexity
Loop 1	2	1.8
Loop 2	6	5.4
Loop 3	4	3.6

The total complexity of all loops in this example is therefore 10.8.

## 8.5 Calculating Model Complexity (Stateflow)

Due to the different structure of the objects contained in Stateflow, M-XRAY employs a different metric for calculating complexity values for Stateflow charts than the one used for Simulink subsystems.

The Stateflow metric algorithm is based on the number of states, actions, and events, as well as the number and scope of conditions. Other Stateflow objects such as SF functions will also be included in the complexity evaluation in a future version of M-XRAY.

### 8.5.1 Complexity of states

---

The complexity of a state results from the sum of the base weights:

- **+1** for the state itself
- **+1** per entry action
- **+1** per during action
- **+1** per exit action
- **+1** per OnEvent
- **+** complexity of contained transitions (see following Chapter 8.5.2)

The weighting of these base quantities can be individually adapted as required.

### 8.5.2 Complexity of transitions

---

Transitions increase the complexity of the state that contains them. The following Stateflow objects are included: conditions, condition actions, transition actions, and events.

The complexity of a transition results from the sum of the following base weights:

- **+0** for transitions that are not associated with an action or condition (these do not increase complexity but can make structuring clearer)
- **+4** for each transition that is associated with a condition
- **+2** (additional) per individual part condition - separated by '&' or '|' (stronger weighting of composite conditions)
- **+1** per action associated with a condition
- **+1.5** per action associated with a transition
- **+4** per event

### 8.5.3 Complexity of truth tables

---

Stateflow can be used to model truth tables as well as state diagrams. Conditions are used to execute corresponding actions, in most cases calculations. The description of a truth table is transferred into a Simulink model prior to simulation. The complexity of this model is determined as detailed in Chapter 8.3.

### 8.5.4 Example of a Stateflow metric calculation

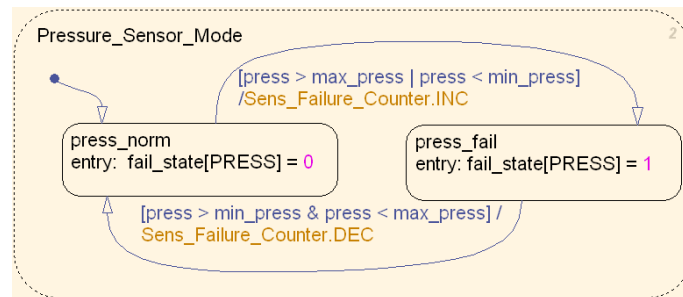


Figure 8-3: Excerpt from a chart

The local complexity of the two states `press_norm` and `press_fail` is 2 respectively: one point for the state itself and another for its entry action in each case.

On the superordinate `Pressure_Sensor_Mode` level, the local complexity for the state itself (+1) is increased by the two transitions, which combine the two sub-states: each +4 for the link of the transition with a condition and an additional +2 for the additional part condition (linked by '|' at the top and '&' at the bottom) and each +1.5 for the transition action (top `INC`, bottom `DEC`).

## 8.6 Scaling of Metrics

Since the complexity values for Simulink systems and Stateflow charts are calculated using different metrics, a scaling is required in order that it become possible to compare the results. The preset scaling value can also be freely configured if required (see Chapter 9.3).

The total complexity of a model is determined from the results of the metrics for both Simulink and Stateflow:

$$\text{Complexity}_{\text{Global}} = \text{complexity}_{\text{Simulink}} + (\text{complexity}_{\text{Stateflow}} * \text{scaling})$$

## 8.7 Displaying Model Structure

Before you can determine the complexity of a model, you must first analyze its structure. The result of this analysis is a model hierarchy that contains all relevant subsystems starting with the top level of the model. The results of structural analysis are displayed by M-XRAY in the results report. This shows the individual subsystems with their name, path, and depth of the subsystem in the model tree.

Libraries play a special role here. They encapsulate model parts that are used repeatedly and are only listed once in the M-XRAY report in the display of model hierarchy. They are only counted once when calculating total complexity in every subsystem that contains them.

The content of subsystems is also shown in the overview of model structure. The total number of contained blocks per subsystem is shown, as well as their block types and width of interfaces of the

subsystem concerned (the number of incoming/outgoing lines). In addition, there is the complexity value of the subsystem, which is calculated on the basis of its content using specific algorithms. In addition to the display of these 'local' values that refer to the direct content of the respective subsystem, 'global' values can also be shown. These include the values of the subsystem including the accumulated values of all those located below this subsystem in the hierarchy. This helps you determine the total number of a particular type of block in the model and where exactly they lie.

## 8.8 Weighting of Blocks

Block weighting measures the required effort for checking this block. The weight thus represents a measure of the local complexity of a block. When blocks are analyzed, the complexity of the interface and the function of the block are taken into account. Moreover, blocks with similar properties are collated into groups and allotted the same weighting factor. Figure 8-4 shows blocks with increasing complexity as an example.

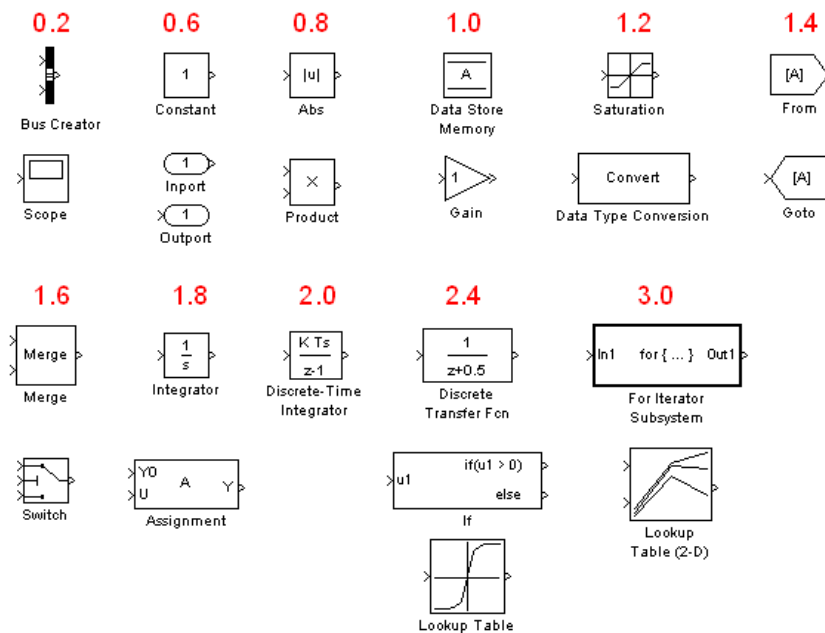


Figure 8-4: Example block weights

In the example shown in Figure 8-5, the inports, the constants, and the outport are all given the same weight of 0.6. The relational operator  $\leq$  is weighted with 1 and the switch block with 1.6.

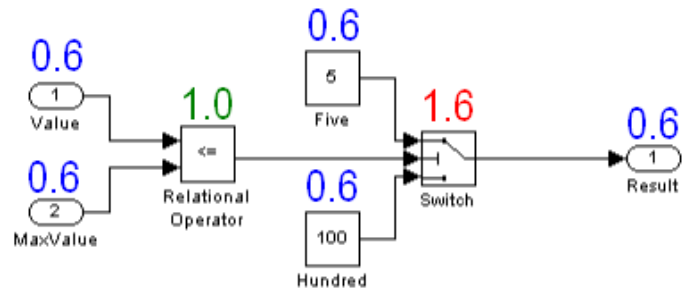


Figure 8-5: Example subsystem with block weights

All the determined standard weightings of Simulink, TargetLink, and Stateflow blocks can be taken from the appendix in Chapter 11. These weights are implemented in the files `mxray_blockweights.m` for Simulink and TargetLink blocks as well as `mxray_SFweights.m` for Stateflow blocks.



## 9 How to Configure M-XRAY

In addition to the different ways of calling M-XRAY described in the preceding chapter, execution of analysis can be influenced by the content of certain files. It is not normally necessary to make changes to these files. They merely serve to configure the weighting of SL/TL blocks and Stateflow objects or enable changes to be made to the list of subsystem mask types to be included in analysis. These configuration options will be discussed in more detail in this chapter.

### 9.1 Configuration of Mask Types

One component of M-XRAY execution is an automatic structural analysis of the analyzed models. This serves to capture the model structure and to identify those subsystems that are relevant to displaying model hierarchy. The first step is to search for all the model blocks that display the `SubSystem` block type. This can lead to a large number of subsystems, which impede a clear overview of structural analysis. The following blocks should therefore be hidden.

<code>Empty</code>	- Masked subsystem without mask type
<code>Stateflow</code>	- Stateflow blocks
<code>FXPCG_SIMFRAME,</code> <code>TL_SimFrame</code>	- TargetLink simulation frame
<code>FXPCG_ENABLE, TL_ENABLE</code>	- TargetLink enabled blocks
<code>Module, cluster, Safety</code>	-
<code>FUMO_TOP_OF_FUNCTION,</code> <code>FUMO_SIMFRAME</code>	- Fumo mask type

The use of a blacklist makes it possible to exclude specific mask types and not include them in analysis. This reduces the selection list in such a way that trivial masked subsystems such as TargetLink blocks do not unnecessarily confuse the overview.

However, the blacklist restricts the subsystems that are looked at in such a way as to exclude some subsystems that might be of interest in the analysis. To avoid this, the respective types can be added again by means of a whitelist and thus included in analysis.



If you wish to take additional mask types into consideration, add these to the `mxray_whitelist.m` file in the `/mxray/configuration` directory of your M-XRAY installation.

#### 9.1.1 Configuring the subsystem under analysis

Subsystems can be treated in two different ways when analyzing model structure:

- As blocks with content that does not need further investigation (this is useful for the TargetLink equivalents of Simulink blocks).
- As full subsystems with content that should be incorporated in the investigation (here subordinate subsystems are also included in analysis and displayed separately).

In which of these two ways a concrete subsystem is treated depends entirely on its mask type. The `mxray_whitelist.m` and `mxray_blacklist.m` files contain a list of mask types that signalize including or excluding full subsystems for analysis purposes. If a subsystem's mask type is not on the whitelist or the blacklist, the subsystem is treated as a block as described above. Execution of analysis can naturally be influenced by changing or adding entries to this file.

All subsystems with an empty mask type (`MaskType = ''`) are included in analysis by default.



If after model analysis, a model part is not displayed as desired, this is usually due to the fact that its mask type is not listed in the whitelist or excluded by the blacklist. This behavior is easily remedied by adding the mask type to the `mxray_whitelist.m` or removing it from the black list.

If you wish to add the mask type 'XYZ' to the white/blacklist, then simply add the line

```
rMasktypes{end+1}= 'XYZ';
```

to the bottom of the `mxray_whitelist.m` file.



Inclusion of the `mxray_whitelist.m` or `mxray_blacklist.m` into analysis can be determined by setting the `bUseWhiteList` or `bUseBlackList` flag respectively. Setting the flag to 1 enables the list, otherwise the list will be ignored.

## 9.2 Configuration of Weights of SL/TL Blocks

As described in Chapter 8, the algorithm metric for Simulink subsystems uses different weights for individual block types. For Simulink and TargetLink blocks, the assignment of weights to individual block types occurs in the `mxray_blockweights` file. This file is configurable. By adapting its content, you can influence the level of complexity values for the subsystems.

The file is repeatedly called during execution of M-XRAY to determine the weights of the blocks detected in the model. The file itself is divided into two parts:

- The first part defines specific default weights that can later be assigned to the individual block types.
- The second part lists the different block types and assigns one of the default weights to each block type.



The block type used in calculating weights does not always correspond with the content of the `BlockType` field of the associated block object. On one hand, certain characters are replaced for processing ('-' by '\_', spaces are dropped).

On the other, in the case of `BlockType = 'SubSystem'`, the `MaskType` is used in place of the `BlockType` for better differentiation.

The weighting of TargetLink blocks usually conforms with their Simulink counterparts, although TargetLink blocks are additionally multiplied by the value of `tlFactor`. This process allows for the somewhat higher complexity of TargetLink blocks. This applies to all types of blocks with a mask type beginning with 'TL\_' and which are not listed in the middle section of the file. The value of `tlFactor` can be varied in the top section of the file. By applying the value '1', the functionality can also be switched off entirely.

The assignment of weights to the block types occurs in the second part of the file using the default weights that are defined in the first part, because this serves to differentiate groups of block types with the same complexity. The weight value of all block types in a group can therefore be easily changed all at the same time by assigning a different value to the default weight in the first part of the file.

The default weights in the second part of the file are laid out using indentations in such a way as to enable the same weights to be found directly under one another. This simply serves for a clearer overview.

Configuration procedure:

- By changing the value of a default weight in the first part of the file, the weights of all block types assigned to this default weight can be changed.
- In the second part of the file, alternative default weights can be assigned to individual block types or concrete values can be entered, if required.
- Additional block types can also be entered and assigned suitable weights. Please use the `Unknown` parameter (cf. Chapter 5.2.2) if you wish to find out whether there are block types in a model that are not included in the configuration file.

The concrete application of these weights in calculating the complexity of subsystems is discussed in greater detail in Chapter 8.



Block types that are not included in the file are automatically assigned the default weight of 1.

### 9.2.1 Example of changing the weights in the `mxray_blockweights` file

Excerpt from file contents (before):

```
% **** Part one ****
(...)
tlFactor = 1.3;
(...)
d_BB = 0.6;
d_CC = 0.8;
(...)
% **** Part two ****
'Constant', d_BB;
```

```
'Inport',    d_BB;
'MinMax',    d_CC;
(...)
```

Excerpt from file contents (after):

```
% **** Part one ****
(...)
tlFactor = 1.5; % Changed: new weight for TL-versions of SL-blocks
(...)
d_BB = 0.4; % Changed: new value for this standard weight
d_CC = 0.8;
(...)
% **** Part two ****
'Constant', d_BB;
'Inport',   d_CC; % Changed: new standard weight for block type
'MinMax',   0.5; % Changed: new direct value for block type
(...)
```

### 9.3 Configuration of Weights of Stateflow Objects

Different weights are equally used for particular objects in the metric for Stateflow charts and states (cf. Chapter 8.5). Stateflow objects are assigned individual weights in the `mxray_SFweights` file. This file is repeatedly called during execution of M-XRAY to define the weights for the Stateflow metric algorithm. The file is configurable and you can influence the calculation of complexity values for charts and states by adapting its content.

The following weights can be configured in the file:

- Weights in connection with transitions:
  - And: weight for '&' and '&&' in transition conditions
  - Or: weight for '|' and '||' in transition conditions
  - CondTran: weight for transitions with conditions
  - CondAct: weight for condition actions
  - TranAct: weight for transition actions
  - Events: weight for events of transitions
- Weights in connection with states:
  - Entry: weight for entry actions in state
  - During: weight for during actions in state
  - Exit: weight for exit actions in state
  - OnEvent: weight for events in state
  - State: weight for the state itself
- Weight for scaling of complexity values of Simulink and Stateflow metrics:

- **Scaling**: The accumulated Stateflow complexity for a chart or a state (as shown in the 'ChartComp' column) is multiplied by this factor before summation with Simulink complexity values.

The use of these weights in calculating the complexity of charts and states is described in greater detail in Chapter 8.5.

### 9.3.1 Example of changing the weights in the `mxray_SFweights` file

Excerpt from file contents (before):

```
(...)  
switch s_object  
(...)  
    case 'Entry',    dWeight = 1;  
(...)  
    case 'Scaling', dWeight = 5;  
(...)
```

Excerpt from file contents (after):

```
(...)  
switch s_object  
(...)  
    case 'Entry',    dWeight = 2; % Changed: new weight for entry  
                        % actions  
(...)  
    case 'Scaling', dWeight = 8; % Changed: new scaling factor  
(...)
```



## 10 Directory Structure of M-XRAY

The M-XRAY application consists of a number of individual functions, most of which are used indirectly by other functions. This means the user only needs to be aware of the following:

In normal GUI use:

- `mxray\mxray.m`

Or alternatively:

- `mxray\path_mxray.m`
- `mxray\functions\mxray_createReport.m`

For configuration purposes:

- `mxray\configuration\mxray_whitelist.m`
- `mxray\configuration\mxray_blacklist.m`
- `mxray\configuration\mxray_blockweights.m`
- `mxray\configuration\mxray_getFilterDefinition.m`
- `mxray\configuration\mxray_SFweights.m`

Information concerning directory structure, important files, and file content in the individual directories will be discussed in this chapter.

### 10.1 Files in the Main Directory

The `...\mxray\` main directory contains all the necessary directories and files for M-XRAY functionality (with the exception of models).

It includes the following directories:

- `...\mxray\configuration\` - (see Chapter 10.2)
- `...\mxray\templates\` - (see Chapter 10.3)
- `...\mxray\functions\` - (see Chapter 10.4)
- `...\mxray\utils\` - (see Chapter 10.5)

It also includes the following files:

- `...\mxray\path_mxray.m` - (M-File to set MATLAB paths, see Chapter 5.2)
- `...\mxray\mxray.m` - (GUI for comfortable operation of M-XRAY functions, see Chapter 3)

## 10.2 Files in the Configuration Directory

The `...\mxray\configuration\` directory contains files whose content can be used to configure the processing of M-XRAY:

- `mxray_whitelist.m`
- `mxray_blacklist.m`
- `mxray_blockweights.m`
- `mxray_SFweights.m`
- `mxray_getFilterDefinition.m`

Their configuration is discussed in Chapter 9.

## 10.3 Files in the Templates Directory

The `...\mxray\templates\` directory includes the two template files for creating reports in Excel format:

- `MXRAY_AnalysisReport_Template.xls`
- `MXRAY_ReviewReport_Template.xls`

Both templates contain ready-made tables and macros. The first of the two templates shows the results of structural and complexity analysis with M-XRAY. The second is for creating reports for model review purposes. M-XRAY uses these templates to create the Excel file with the results of model analysis.

## 10.4 Files in the Functions Directory

The `...\mxray\functions\` directory contains the functions of M-XRAY in M-files. It can be called from the GUI `mxray_GUI.m` or via the main function `mxray_createReport.m`. Their functionality is explained in the respective MATLAB help block (`help nameoffunction`).

## 10.5 Files in the Utils Directory

The `...\mxray\utils\` directory contains various auxiliary functions of M-XRAY in M-files. These are called by the M-files in the `...\mxray\functions\` directory. Their functionality is explained in the respective MATLAB help block (`help nameoffunction`).

## 11 Tables Appendix

### 11.1 Block Weights for Simulink Blocks

The following table gives the currently preconfigured Simulink blocks with their standard weights, ordered into groups.

Name	Weight	Name	Weight
S_Function_in_Chart	0	Logic_N	1
ActionPort	0.2	RelationalOperator	1
BusCreator	0.2	SubSystem	1
BusSelector	0.2	DataTypeConversion	1.2
Demux	0.2	Saturate	1.2
EnablePort	0.2	From	1.4
Function_CallGenerator	0.2	Goto	1.4
Ground	0.2	Merge	1.6
Mux	0.2	MultiPortSwitch	1.6
Scope	0.2	Switch	1.6
Terminator	0.2	Assignment	1.8
TriggerPort	0.2	Integrator	1.8
Constant	0.6	Math	1.8
Inport	0.6	Selector	1.8
Logic	0.6	Signum	1.8
Output	0.6	UnitDelay	1.8
ToWorkspace	0.6	ZeroOrderHold	1.8
Abs	0.8	DiscreteIntegrator	2
MinMax	0.8	DiscreteTransferFcn	2.4
Product	0.8	Fcn	2.4
Sum	0.8	If	2.4
DataStoreMemory	1	Lookup	2.4
DataStoreRead	1	Lookup1D	2.4
DataStoreWrite	1	ForIterator	3
Gain	1	Lookup2D	3

### 11.2 Block Weights for TargetLink Blocks

The following table gives the currently preconfigured TargetLink blocks with their standard weights.

Name	Weight
TL_AddFile	0.0
TL_DummyTrigger	0.0
TL_Function	0.0
TL_Inport	0.2
TL_Outport	0.2
TL_LogicalOperator	0.5
TL_Enable	0.6
TL_MainDialog	0.6
TL_MilHandler	0.6
TL_SimFrame	0.6
TL_ToolSelector	0.6
TL_LogicalOperator_N	0.8
TL_BusInport	1.8
TL_UnitDelayRE	2.3
TL_SRFlop	2.4

### 11.3 Weights for Stateflow Elements

The following table gives the currently preconfigured Stateflow blocks with their standard weights.

Name	Weight
CondAct	1
Entry	1
During	1
Exit	1
OnEvent	1
State	1
TranAct	1.5
And	2
Or	2
CondTran	4
Events	4