myM

Getting Started 6

website: www.my-m.eu
e-mail: info@my-m.eu

Contents

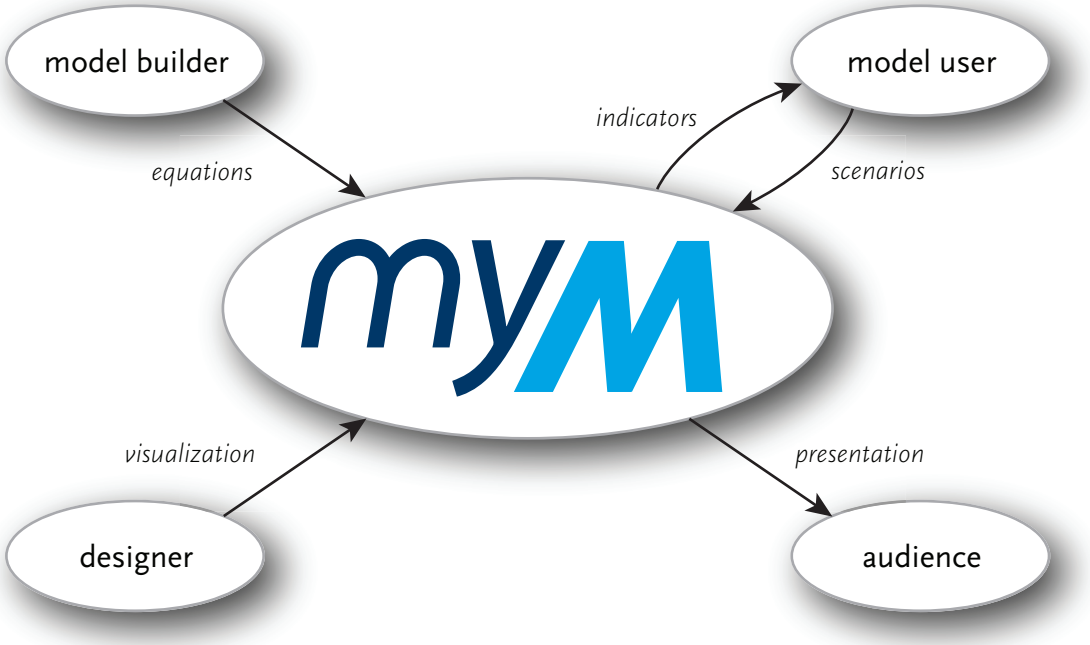

 Figure 1: Integrated model development

1. Introduction

This manual gives a brief and practical introduction to the MyM package. PDF versions of reference manuals of the language and of the GIM are included in the distribution.

The MyM package is used to define, visualize and run mathematical models of dynamic systems.

MyM integrates the development, application, design and presentation of these models into one software package.
We define four roles: the model builder, the model user, the interface designer and the audience. They all use the same MyM software (see figure 1).
It is possible that one person takes more than one role. E.g., the model builder will also explore and debug his model and might also be the person to design the interface for the target audience. MyM allows multiple interfaces on the same model, allowing the designer to create a interface focussed on different audiences or viewpoints.

MyM models can be distributed as stand-alone applications in which users cannot modify the mathematical specifications and can only change input variables selected by the developer.
Mathematical model can only be built and modified by anyone having a licensed version of the MyM package.

Stand-alone applications can be distributed freely by distributing the model with the MyM Player, however the MyM software must be purchased at the MyM webstore at http://www.my-m.eu.

On-line documentation (user and system) is available in HTML and PDF. Acrobat Reader is required to read the manuals in PDF format, which is available for download at http://www.adobe.com/nl/products/acrobat/readstep2.html

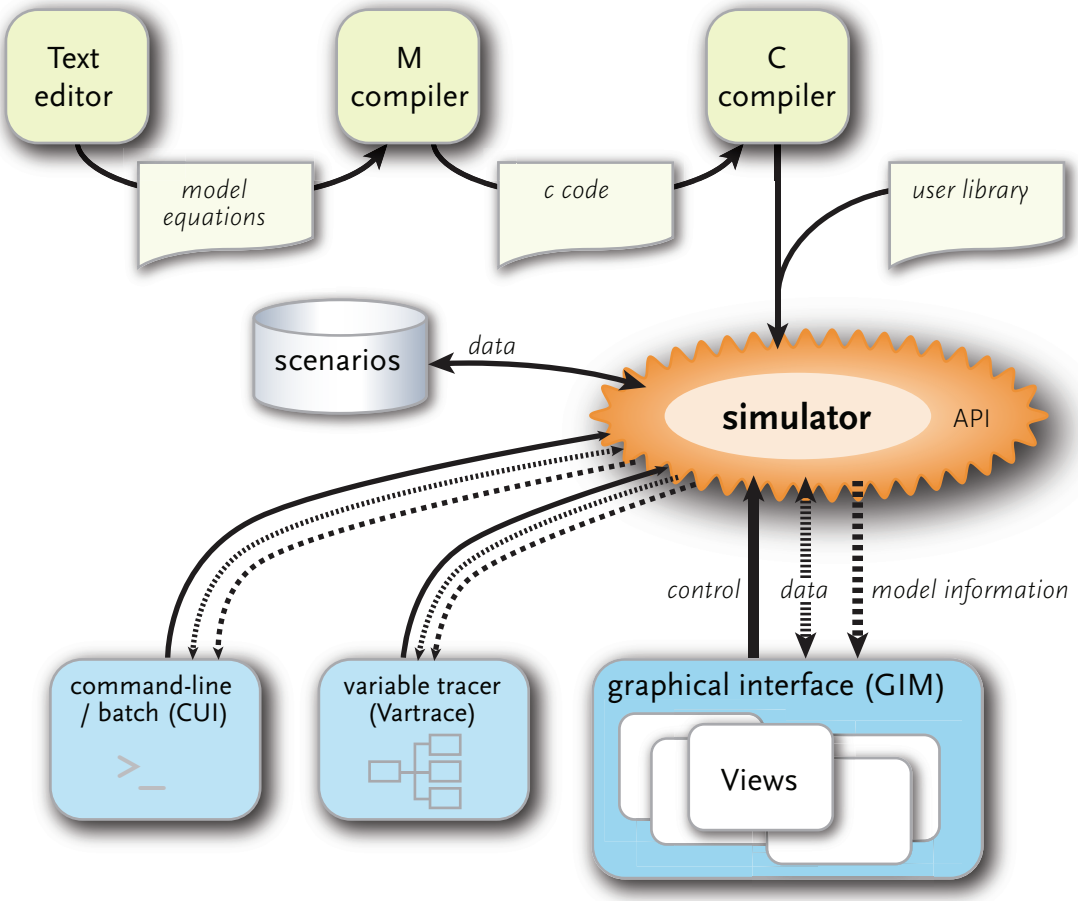

Figure 2: MyM architecture

Figure 2 shows the MyM architecture in more detail.

The basic steps in using MyM are:

- write mathematical models using any ASCII editor by writing equations in the MyM language
- MyM will turn this set of equations into an executable simulator (by creating intermediate c-code)
- run the model using one the user interfaces:
  - a *graphical interface* to design and use graphical interfaces to compiled MyM models, called MGIM (Graphical Interface Manager)
  - a *variable tracer* to trace the dependencies among the variables in a model, called MVartrace
  - a *text-based interpreter* to control a model from a command prompt, named MCUI (Command Line Interface), or the
  - *player*, which can be used to distribute your own models with a graphical user interface. The interface has to be designed with MGIM.

2. Requirements

To run a stand-alone MyM model, the MyM Viewer is required.
To create or edit models users need:

- a licensed version of the MyM Software,
- a properly installed ANSI-C compiler Visual C++ 8 (Visual Studio 2005 Express, Standard or Professional).

The C compiler is needed to generate an executable from the intermediate C code into which MyM translates the set of equations of the MyM model. No knowledge about the C compiler is needed to use MyM.
To use the MyM development package, users need a PC running Microsoft® Windows® XP with Service Pack 2 or higher. Versions for Linux and OSX will become available in the near future.
Hardware requirements depend mainly on the size of the models developed by the user. MyM requires 100 MB of available disk space and at least 512 MB of RAM.

3. Installation

The MyM package can be downloaded from our website, as a self-extracting executable.
Run the Setup executable and follow the instructions. Please review the README file included with the installation for environment variables that may need to be set by hand, such as M_PATH to the directory were MyM is being installed and C_PATH to the directory where the C Compiler can be found. The latter is needed if one wants to create new MyM models.

See the Environment Appendix in the GIM Reference Manual for some additional variables that can be changed, for instance to affect the way MyM responds to mouse actions.

4. Documentation and support

The most recent information (and the latest version) is available at the MyM Web site. The following documentation is available:

- this Getting Started Manual
- a Language Tutorial
- a Language Reference Manual
- a Graphical Interface Reference Manual
- an Application Programmer's Interface Reference Manual

The distribution includes PDF versions of these documents that can be consulted and the GIM Manual can be consulted directly from the Help menu or directly using a HTML browser. Context-sensitive help about specific interface components is available by placing the mouse over the item of interest and pressing the F1 function key.

Please send suggestions, questions or comments to support@my-m.eu.

5. The basic steps

There are 2 ways to build MyM Models. The first method is to use the `mmake` command on the command-line. The second method is to use the Integrated Development Environment (IDE) named MStudio. We will discuss MStudio in the next section.

Let's take a brief look at the steps needed to create a simple model. We will use a simple model of water cascading through a couple of buckets as our example. This casc (cascade) model is included in the distribution, in the samples directory. Use any ASCII text editor to look at the casc.m file. (Always use the .m extension for files with MyM source code).
The MyM Language Tutorial takes you step by step through the mathematical formulation of this model. The MyM Language Reference Manual contains a full description of the MyM language.
To compile the model casc.m on command-line you first have to open a command prompt (type cmd at Windows Start->run) and change the current directory to the directory where the MyM model is located.

Then use the command

```
m2c casc.m
```

to generate casc.c, the file to be passed to the C compiler. The m2c command does syntax checking and accepts various flags to allow run-time error checking on array bounds and floating-point calculations. Type

```
m2c -help
```

for an overview of the available options.

Once a MyM model has been successfully translated into C, actual compilation is done by issuing the command:

```
mmake model.mdl
```

mmake casc.mdl generates a simulator that can be loaded dynamically during run-time into any of the three general interfaces included in the distribution, i.e.: MGIM, MCUI or MVARTRACE

- `mgim casc.mdl` runs the graphical interface for the casc model
- `mcui casc.mdl` runs the command-line interface for the casc model
- `mvartrace casc.mdl` runs the tracer of the variables in the casc model.

The CUI interface should be self-explanatory (type HELP at the prompt), as should Vartrace.

In the next section we discuss the Integrated Development Environment as a more user-friendly method to create MyM models.
In the remaining sections we will concentrate on the more complex and more powerful Graphical Interface Manager MGIM.

6. Using MyM Studio

MStudio was first introduced in version 6 of the MyM Software. It offers the user a graphical user interface to build MyM models.
Using an Integrated Development Environment saves you time in developing models, by managing editors, settings, and compilation of MyM models. By using MyM Studio the modeler does not need to use command line, but can just develop the models in the graphical IDE.
Source editing, compilation of the model, and generation of the simulation model are integrated in MStudio. Highlighted code and an error output window gives you the necessary feedback for writing MyM models.

Project

When you launch MStudio it opens with an empty project with no name.
Within MStudio, you work within the context of a project, which consists of a group of source files and associated metadata; project specific properties; and all the tools you will need to write and compile your source code into an executable MyM model.

First you have to give the project a name and save the project to disk.
Once you give MyM Studio the new project name, it updates the project tree.
From there you either add an existing MyM file (like casc.m) to the Project by using Project->Add existing item or use Project->Add new item from the menu bar. It will add a file to the project's source tree and it will open it a new editor. The first file added will also be made target of the Project by default.
The project's target is the main MyM file that contains the model and will be converted into a .mdl model in the compilation process.

The Project window displays the files that you want to be regularly edited, such as the source files of your model.

Double-click on a source file automatically opens the file in the workspace at the right-hand side in a new or existing source Editor.
It is also possible to exclude a file from the project, by selecting the file in the Project window and selecting Project->exclude item from the menu or right click the file in the source tree and select “exclude”.

Using the Source Editor

In the Source Editor the user can write MyM code for the model. When you write equations this way, you will notice that the editor automatically highlights the code in color.

Source Editor Options

You can change the font of the source editors by going to Options->Font
You can jump to a certain line by using Ctrl+G inside the editor’s window or Select Edit->Goto Line.

Setting project properties

The properties of a project determine how the model will be built.

Compilation of a MyM model is done in 2 stages. First the MyM file is translated into C code and then by using the Microsoft Visual Studio C Compiler it is translated into an executable MyM model.
In the project properties you can set the target model, additional flags for compilation in the m2c compilation process and compiler options.
MyM Studio will try set defaults for the options.
If the options are not set by default you would have to enter them.
The most important options are:

MyM directory

The path to the MyM directory you selected during installation.

Type C++ compiler

Currently MyM only supports the Visual Studio Compiler 2005, selected by the name “VC8”. Currently you cannot change this value, because VC8 is the only supported compiler.

Path C++ Compiler

The path to where the Microsoft Visual C++ 2005 executables are installed, typically C:/Program Files/Microsoft Visual Studio 8/VC/bin. Do not change this value if MyM Studio set it for you.

Building MyM Models

Make sure you have the Visual Studio 2005 C++ Compiler installed before you try to build a MyM model.
Before you can compile a model you also need to have one MyM source file in the project browser to be selected as the target. The target source file is indicated by a bold font. If none of the files in the project file browser is selected, select a source file and then select Project->Set as project target in the menu bar.

There are two stages in the model build process. First a compilation and then the actual building of the simulator.

You can see which source file is selected as the target model and will be compiled or built by selecting the General tab of the Project properties dialog.

To compile a model and check for syntax errors select Build->Compile form the menu bar.
If there are no errors in the output window, you can start building the model. It might be that the compilation process generated warnings. Please read them carefully and doublecheck that these warnings will not cause future problems.

From that moment you can build a model by going to Build->Build model.

7. Designing a graphical interface

Start the graphical interface of MyM, by double-clicking in the File Manager or typing MGIM at a command prompt.
MGIM will bring up an empty window with a menu and tool bar along the top.
In the general MGIM (when no model is linked yet), the File->Load Model menu option should be used first to load a model. This should be the compiled version (e.g. casc.mdl), not the source code version (e.g. casc.m).
In our example we will use the GIM in the samples/casc directory, where the model is already compiled to an executable simulator.

To see how an interface of the cascade model could look like, select the View Manager in the Tools menu. In MyM any number of views on a model can be defined and kept around. Initially only an (empty) default view is available.

Load a predefined view by clicking on View->Load... and use the file browser to load casc.vdf from the samples/casc directory.
A VDF or View Definition File is a data file containing the definition of a particular view: which variables are included and how they are presented and (hierarchically) organized. Do not edit these files outside the graphical interface, as they can easily become corrupt and unusable.

8. Exploring a view

If you load a view a couple of (closed) containers will appear in the window.

To get a complete overview, put your mouse anywhere outside any container and press the right button. A pop-up menu will appear, select *Open All*. The sliders and graphs shown visualize all main variables in casc.
Click Right on some box and select *Zoom In* or *Zoom Out* from the context menu to explore the various components in more detail.

9. Editing model inputs

Input variables are variables that do not depend on other variables, i.e. they have no arrows pointing at them[1]. Input variables can be edited directly, for instance by placing the mouse over a slider, pressing left and dragging.

While sliders represent constants. i.e. variables that do not change over time, line graphs representing time-dependent functions can also be edited directly.
This is done by redrawing the graph using a left-drag, i.e. by dragging the mouse while pressing the left button. Doing so resets the function for specific domain values[2].

1. Another way to find out whether a variable is an input variable is by pointing at it and pressing Ctrl-E. For input variables nothing will happen, for dependent variables this will bring up the equation for that variable.
2. Which domain values is determined by the values used in the initialization statement in the M model: a(t) = [1, 0, 10, 0] means that only the value for t = 1 and t = 10 can be changed. To allow a higher resolution, add more values to this initial set, e.g.a(t) = [1, 0, 2, 0, 3, 0, 5, 0, 10, 0].

Various options are available, including a Table Editor in which the numbers can be typed in directly. Mouse-right over the graph displays the options.

The Load (and Save) options are one way of using previously defined data stored in MyM format. See the section on Scenarios in the MyM Graphical User Interface Reference Manual for further details.

10. Changing appearances

Any MyM variable in MGIM is characterized by its range, its domain (unless it is a constant) and its dimensionality:

- *range* – the minimum and maximum values shown, the unit, sub-ranges etc.
- *domain* – the minimum and maximum time value, unit etc.
- *dimensionality* – as MyM variables can have any number of indices or dimensions, of any size, a mechanism is needed to keep track of the subset selected for visualization; depending on the graphical type used (line graph, grid map, barchart) up to two dimensions can be selected, and from these one or more values (classes).

To edit these aspects, the appropriate dialog can be selected from the *Edit* menu.

The Widget option under *Edit* is used to select the type of graph and to select the dimensions, the Dimension dialog is used to select values within those dimensions.
The Layout dialog is used to set the appearance attributes of the widget, like colors, font, etc.

11. Modes

The Layout dialog is only available in *Design* mode (select *Design* in the *Mode* menu).

This mode is also needed to add items (the *New* menu), to cut-and-paste items (in the *Edit* menu) or for access to the advanced options in the various Edit dialogs that are activated by pressing the button with the >> label.

Press this button to select or edit templates for ranges, domains or dimensions.

12. Templates

The specifications for ranges, domains and dimensions are stored in named templates that can be used for multiple variables. Doing so ensures that these variables use the same scales, units, labels and color codings.

By editing a template, e.g. selecting a different set of values on a dimension or changing a color coding, all variables using that template will be updated automatically.

This also happens when a range is edited directly by scaling or dragging with the mouse.

13. Adding variables

To add a variable to a view, make sure you are in *Design* Mode and select the *Variable* option in the *New* menu.
A variable browser will appear showing all variables not yet contained in the currently active view. If you do this in the view included, an empty list will appear, as all variables have been used.
Use the View manager (in the *View* menu) to go back to the empty (Default) view and try again.
Select a variable with mouse-left, put your mouse somewhere in the window and drag-left to determine the size of the new box.
To add several variables, keep the Shift pressed to prevent the variable browser from disappearing.

14. Grouping, moving and resizing items

To group items, select Container under New and draw a box over the items (i.e. variables or containers), by drag-left.

Components can also be added or removed by moving them around.

Items are resized by dragging their borders with the left mouse key, and moved by pressing the Alt- key during a drag-left.

15. Looking at different views

Using the *View Manager*, one can load existing views and switch between views

It is also possible to split the window (using the *Split* menu in the menu *Window*) and to select the same or different views for each pane.

When looking at different instances of the same view, zooming can be synchronized (in the View manager).

16. Saving and loading data

There are several ways to handle data input/output:

- direct manipulation of sliders and graphs
- the Table Editor (from the Tools menu or the right-mouse pop-up menu)
- use of the Load and Save options in the right-mouse pop-up menu
- use of the FILE construct in the MyM source code for a model (or of the PROG construct to execute a shell command to read or write data). For more information about the FILE contruct consult section "Variable Declarations" in the MyM Language Reference Manual.
- use of the Scenario Manager in the Tools menu (see the next section).

Although a Load or Save on a container (3rd option) will set or store all variables within that container, options 1 to 4 are basically intended to deal with one variable at a time.

17. Scenarios

To read and save sets of variables and provide a systematic handling of data storage, use of the scenario manager is recommended. The term scenario is used loosely in this context, it referes to any set of variables that are handled as a set. A scenario can be restricted to all or some inputs, to all or some outputs or to combinations of inputs, outputs and intermediate variables. (Obviously, all non-input variables will be recomputed when the model is run).

Scenarios are read and stored using the *Scenario Manager* (under Tools). Up to 10 scenarios can be kept in memory for easy comparison.
The Scenario Manager is used to load and unload scenarios from hard disk, to select a new scenario for the current view or to save a scenario, giving it a name, a label and a color for use in graphs comparing variables from different scenarios, an option available by selecting the scenario dimension in the Widget dialog.

Another option to compare scenarios (and the only one available when dealing with maps) is to split the window into several panes showing the same graph for different scenarios.

The variables to be stored or read, and the files to be used for this purpose, are specified in a template that is stored in a file named modelname.sce in the same directory where the model is located. This template can be parameterized.

The Scenario Manager uses the scenario name to instantiate the first parameter in the template. Depending on the use of the parameter in the template, scenarios can thus either be stored in different directories or with different filenames or extensions. See Appendix C of the MyM GIM Reference Manual for full details.

18. File formats

MyM uses several file types, each of which is characterized by its own extension. Some of these files should never be changed by the user (.VDF, .MCM, .MDL), others will have to be created and maintained by the developer of a MyM model:

- a *model specification* in a file model.m (see Language Reference),
- a *scenario template* in model.sce (see Appendix C of the GIM Reference Manual),
- *raster* or *vector maps* in .VSF or .RSF files (see Appendix D of the GIM Reference Manual).

For more information about data formats consult the MDIO document and the Data Specification information on the MyM website.