InterCAX LLC

75 Fifth Street NW, Suite 213
Atlanta, GA 30308, USA
Voice: +1- 404-592-6897
Web: www.InterCAX.com
E-mail: info@intercax.com

ParaMagic™ v16.6 sp1
Users Guide

Table of Contents

1 About

This is the Users Guide for ParaMagic™ 16.6 sp1. The sections in this document and their purpose are stated below:

- Quick Start – speed up learning ParaMagic™ (run existing models, create new models)
- Installation– installation instructions for ParaMagic™
- User Documents – important documents for using ParaMagic™ (including tutorials and examples)
- SysML Model Requirements – requirements for creating SysML models executable in ParaMagic™.
- Program Features – learn about the ParaMagic™ plugin user interface
- Connections to External Solvers – learn about incorporating Mathematica functions, MATLAB functions/scripts & Simulink models, and Excel spreadsheets as part of SysML parameteric models, and using ParaMagic to execute parametric models that require multiple solvers.
- Copyright –important copyright information that users must read

The following conventions are followed in this document:

- ParaMagic™ implies this ParaMagic™ 16.6 sp1 plugin
- MagicDraw SysML implies MagicDraw UML + MagicDraw SysML plugin
- Text enclosed by < > implies that you need to provide your system-specific settings, such as installation directory path or IP address.
- Text written in this `font` implies that it is a computer keyword or an abbreviation for a computer keyword.
- MD implies MagicDraw
- `<MD_Root>` refers to the MagicDraw installation directory (e.g. `C:\Program Files\MagicDraw_UML_16.6` on a Windows machine).
- OM implies OpenModelica

MagicDraw[1] is a registered trademark of No Magic, Inc.
Mathematica[2] is a registered trademark of Wolfram Research, Inc.
MATLAB[3] and Simulink[4] are registered trademarks of The MathWorks, Inc.
OpenModelica[5] is a freely-available modeling and simulation tool as part of the OpenModelica Project supported by the Open Source Modelica Consortium (OSMC).

Note: If you are an existing ParaMagic™ user, refer to section 5.8. The section presents steps for upgrading your models from ParaMagic™ 16.5 to ParaMagic™ 16.6 and later.

2 Quick Start

This section recommends a starting point for users to quickly learn ParaMagic™.

2.1 First Pass – execute existing models

In this first pass you will review and execute parametrics in existing SysML models per the next steps.

[1] MagicDraw: http://www.magicdraw.com
[2] Mathematica (Wolfram Research): http://www.wolfram.com/
[3] MATLAB (The MathWorks) - http://www.mathworks.com/products/matlab/
[4] Simulink (The MathWorks) - http://www.mathworks.com/products/simulink/
[5] OpenModelica - http://www.ida.liu.se/~pelab/modelica/OpenModelica.html

- The first model is a basic problem from the mechanical domain called SpringSystems. It includes quick start instructions on how to execute parametrics using ParaMagic^{TM}. It exercises many SysML constructs and modeling concepts and is thus useful for learning parametrics.
- The second model is a UAV road scanning system problem. It is based on the LittleEye problem outlined by No Magic and is a systems engineering-oriented demonstration model. It is also a part of the step-by-step tutorial provided with ParaMagic^{TM}.

1) Install ParaMagic^{TM} (per Section 3 below).

2) Open and execute the SpringSystems model
   a) Run MagicDraw and open the SpringSystems model located here: `<MD_Root>\samples\ParaMagic\Other_Examples\SpringSystems.mdzip`
   b) Follow the instructions in the Model Overview diagram to execute this model.
   c) See Section 2.2 of the INCOSE paper (IS07 Part 1) located here: http://eislab.gatech.edu/pubs/conferences/2007-incose-is-1-peak-primer/ for a detailed description of this model

3) Open and execute the LittleEye model
   a) Run MagicDraw and open the LittleEye model located here: `<MD_Root>\samples\ParaMagic\Tutorials\LittleEye.mdzip`
   b) The model is saved in a state that corresponds to the end of the LittleEye tutorial Step VII (i.e., the instance state seen in Figure 4.9 in the Tutorial[6] document).
   c) Review the Objective section of the LittleEye tutorial to get a feel for what this model is designed to do. The tutorial document is located here: `<MD_Root>\samples\ParaMagic\Tutorials\`
   d) Proceed with Step VIII in the LittleEye tutorial to validate and execute this model using the same kind of user interface actions you learned in the SpringSystems model above.
   e) Continue through the end of the LittleEye Tutorial to change a value, re-solve the model, and so on.

2.2 Second Pass – create new models

This Second Pass takes you deeper, including learning how to create SysML models that contain parametrics that can be solved using ParaMagic.

It is useful to identify several types of users who work with SysML parametrics:

- Type 1: Someone who works with an existing model (including executing it and performing additional instance-oriented interactions such as solving, modifying values, changing causalities, and re-solving).
  - This type of user needs the least amount of SysML and parametrics know-how.
  - This a good place to start for the casual user, for someone wanting to do basic demos, or someone just beginning to explore SysML parametrics.
- Type 2: Someone who modifies the structure of an existing model and/or creates new instances.
  - This type of user requires more know-how.
  - They also need a fair amount of MagicDraw SysML tool-aided modification support (in some respects more than Type 3 users).
  - This is a good step towards becoming a Type 3 user.
- Type 3: Someone who creates their own model structures and instances from scratch (and/or from pre-existing building blocks from a library).
  - This type of user requires a fair amount of know-how and needs good MagicDraw SysML support.

[6] Tutorial document: `<MD_Root>\samples\ParaMagic\Tutorials\Tutorials.pdf`

- Type 4: Someone who creates building block libraries that Type 2 and Type 3 users can utilize.
  - This type of user requires similar skills as Type 3, but with a bent towards making their work reusable and modular, as well as providing good documentation and rigorous validation.

The First Pass in the section above provides a quick introduction for Type 1 users.

After completing the First Pass, you can work at the Type 1 level with all the pre-built tutorial models and examples provided in this release (see a listing of these models in Section 4 of this document).

After completing the First Pass, you also have a good "big picture" basis to now proceed with the step-by-step tutorials (see section 4.2). After completing the tutorials, you will have achieved a good foundation to work as a Type 3 user. There are other topics you may eventually need that may be addressed in future tutorials and/or courses.

3 INSTALLATION

3.1 Installation Requirements

3.1.1 System Requirements

1) Operating system: ParaMagic™ has been tested to work[7] with MagicDraw SysML (i.e. MagicDraw UML + SysML plugin) installed on the following operating systems:
   a) Windows XP 32-bit and 64-bit editions.
   b) Windows VISTA 32-bit edition.
   c) Mac OS X 10.6 (Snow Leopard) 64-bit edition.
   d) Linux (Ubuntu 9.04).
2) Java: ParaMagic™ requires Java 1.6 or higher. ParaMagic™ uses the same Java installation that is being used by MagicDraw. To check the Java version used by MagicDraw, check that the value of the JAVA_HOME variable in <MD_Root>\bin\mduml.properties file points to Java 1.6 installation on your computer.
3) Hard disk space: ParaMagic™ requires 30 MB of hard disk space for installation.
4) RAM: ParaMagic™ requires 500 MB of memory. Additional available RAM will improve the performance of the plugin.

3.1.2 MagicDraw Requirements

1) MagicDraw SysML 16.6 sp2 (or higher) should have been installed and configured on your system.
2) ParaMagic™ 16.6 sp1 has been tested to work with MagicDraw SysML 16.6 sp2 Enterprise edition. In principle, it should work with other editions of MagicDraw SysML but has not been tested for them.

3.1.3 Core Solver Requirements

ParaMagic™ 16.6 sp1 allows users to select Mathematica or OpenModelica[8] as the core solver—see section 3.2 (Step 3) for installation and configuration details.

[7] Unless otherwise specified, ParaMagic™ may work with other editions of these OS but it is not been rigorously tested for editions other than those mentioned here. For example, ParaMagic™ should work with Windows VISTA 64 bit edition but hasn't been tested with it.
[8] OpenModelica: http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html

If you plan to use Mathematica as the core solver:

1) ParaMagic™ has been tested with Mathematica version 7.0.
2) ParaMagic™ has been tested to work with local Mathematica installed directly on user's hard drive or on a networked hard drive.
3) ParaMagic™ has also been tested to work with Mathematica installed on a network, using our customized web-services product XaiTools Web Services™ (a.k.a. XWS). Please contact us (info@intercax.com) if you would like to setup XWS for your organization.

If you plan to use OpenModelica as the core solver:

1) ParaMagic™ has been tested to work with OpenModelica 1.5.0 (RC2 on Windows and RC1 on Mac) installed locally on a user's hard drive.

3.2 Installation Process

Before installing the ParaMagic™ plugin, make sure that you have administrative privileges to install new software on your computer. Windows VISTA users should ensure that the User Account Control (UAC) is turned off even if they have administrative privileges on the computer. To learn more about UAC settings in Windows VISTA and how to turn it off, click here.

Follow the steps below to install ParaMagic™ plugin.

Step 1. Start MagicDraw (MD)

If you are already running MD, close all open projects. You do not need to close MD itself.

Step 2. Install the ParaMagic™ plugin

a) *If you do not have the ParaMagic™ plugin (zip file),*
   i) In the main menu bar, select Help → Resource/Plugin Manager
   ii) Click on Check for Updates button
   iii) You will see ParaMagic™ plugin version 16.6 sp1 available for installation. Select this plugin.
   iv) Click on Download / Install button to install ParaMagic™ plugin.
   v) After installation, you will need to restart MD for the plugin to take effect. It is suggested that you finish all the steps below before restarting MD.

b) *If you have the ParaMagic™ plugin (zip file),*
   i) In the main menu bar, select Help → Resource/Plugin Manager
   ii) Click on Import button and specify the location of the plugin zip file.
   iii) After installation, you will need to restart MD for the plugin to take effect. It is suggested that you finish all the steps below before restarting MD.

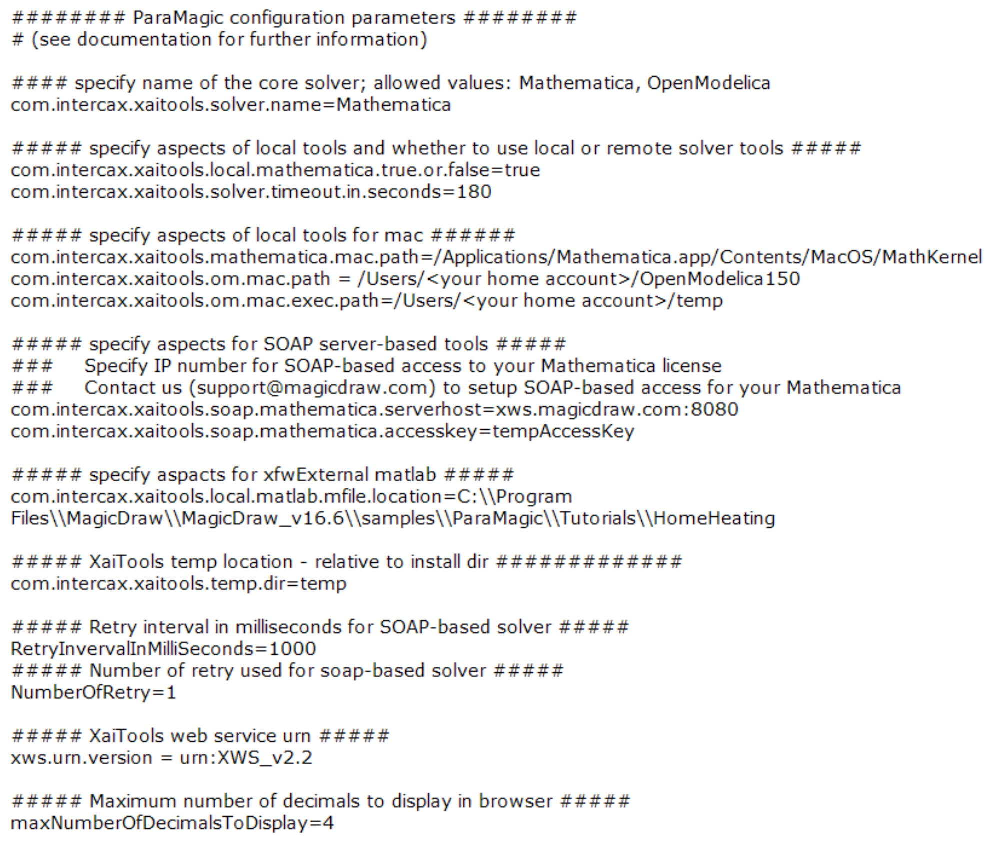
The ParaMagic™ download site (www.magicdraw.com/download/paramagic) provides a bundle (zip) that includes the ParaMagic™ plugin and OpenModelica installation for Windows and Mac. If you plan to use OpenModelica as the core solver, download and unzip the bundle. Then, install the ParaMagic™ plugin (zip file) per option b above.

Step 3. Select the core solver for ParaMagic™

ParaMagic™ 16.6 sp1 allows users to select Mathematica or OpenModelica[9] as the core solver. OpenModelica is a *free* Modelica-based modeling and simulation tool available as part of the OpenModelica Project that is managed by the Open Source Modelica Consortium (OSMC).

[9] OpenModelica: http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html

Specify the core solver in ParaMagic.ini file
a) Open <MD_Root>\plugins\com.intercax.paramagic\xfw\conf\ParaMagic.ini file in a text editor
b) You will see text as shown in Figure 1. Lines not starting with # character are configuration settings.
c) Set the value of the variable com.intercax.xaitools.solver.name equal to
   1. Mathematica to use Mathematica as the core solver
   2. OpenModelica to use OpenModelica as the core solver
d) Save and close the ParaMagic.ini file
e) Skip Step 5 if you have selected Mathematica as the core solver, or skip Step 4 if you have selected OpenModelica as the core solver.



```
######## ParaMagic configuration parameters ########
# (see documentation for further information)

#### specify name of the core solver; allowed values: Mathematica, OpenModelica
com.intercax.xaitools.solver.name=Mathematica

##### specify aspects of local tools and whether to use local or remote solver tools #####
com.intercax.xaitools.local.mathematica.true.or.false=true
com.intercax.xaitools.solver.timeout.in.seconds=180

##### specify aspects of local tools for mac ######
com.intercax.xaitools.mathematica.mac.path=/Applications/Mathematica.app/Contents/MacOS/MathKernel
com.intercax.xaitools.om.mac.path = /Users/<your home account>/OpenModelica150
com.intercax.xaitools.om.mac.exec.path=/Users/<your home account>/temp

##### specify aspects for SOAP server-based tools #####
###     Specify IP number for SOAP-based access to your Mathematica license
###     Contact us (support@magicdraw.com) to setup SOAP-based access for your Mathematica
com.intercax.xaitools.soap.mathematica.serverhost=xws.magicdraw.com:8080
com.intercax.xaitools.soap.mathematica.accesskey=tempAccessKey

##### specify aspacts for xfwExternal matlab #####
com.intercax.xaitools.local.matlab.mfile.location=C:\\Program
Files\\MagicDraw\\MagicDraw_v16.6\\samples\\ParaMagic\\Tutorials\\HomeHeating

##### XaiTools temp location - relative to install dir #############
com.intercax.xaitools.temp.dir=temp

##### Retry interval in milliseconds for SOAP-based solver #####
RetryInvervalInMilliSeconds=1000
##### Number of retry used for soap-based solver #####
NumberOfRetry=1

##### XaiTools web service urn #####
xws.urn.version = urn:XWS_v2.2

##### Maximum number of decimals to display in browser #####
maxNumberOfDecimalsToDisplay=4
```

Figure 1: ParaMagic.ini file

Step 4. Mathematica installation

If you have selected Mathematica as the core solver in Step 3, you have the following choices:

- For short-term (1 month) evaluation, use our test Mathematica server—see Option 1 below.
- For long-term or production usage,
  - use Mathematica locally-installed on your machine—see Option 2.
  - use Mathematica installed in your enterprise as a web-service via our XWS product—see Option 3.

Option 1. Using remote Mathematica installation at No Magic for temporary evaluation

This installation comes with a temporary access to our test Mathematica server. You may use this access to evaluate ParaMagic™ with our test server. After that, you must have your own Mathematica license (Option 2) or have access to an XWS-based server (Option 3). Contact support@magicdraw.com for issues/questions regarding our test Mathematica server. Please complete the steps below to configure local / network access to your Mathematica license.

a) Open the ParaMagic.ini file.
b) Set the variable com.intercax.xaitools.local.mathematica.true.or.false to false. This is set to false by default (as shown above).
c) Set the variable com.intercax.xaitools.soap.mathematica.serverhost to the IP number provided by technical support, followed by a colon followed by the port number. By default, this value is set to the Mathematica server at No Magic (xws.magicdraw.com:8080).
d) Set the access key variable com.intercax.xaitools.soap.mathematica.accesskey to the access key provided by technical support. By default, a temporary access key (tempAccessKey) is already specified in the ParaMagic.ini file (see above) for you to try out this new release of ParaMagic^{TM}.
e) Save and close the ParaMagic.ini file.

Option 2. Using your local installation of Mathematica

If you have a local installation of Mathematica, follow the steps below. Here, local implies (a) either on your machine hard drive, or (b) a networked hard drive.

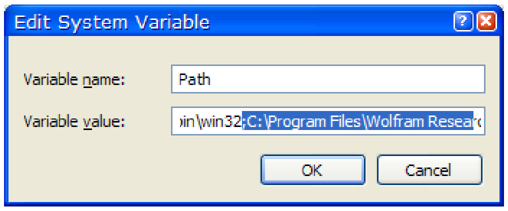
a) Open the ParaMagic.ini file.
b) Check that the com.intercax.xaitools.local.mathematica.true.or.false variable is set to true in the ParaMagic.ini file. By default, this variable is set to false (as shown above).
c) If you are using Windows OS, ensure that Mathematica is accessible from your machine. Type math at the command line on a console. You should see some output such as below (for a Windows machine):

```
C:\>math
Mathematica 6.0 for Microsoft Windows
Copyright 1988-2008 Wolfram Research, Inc.
```

If you do not see text similar to above, check that the environment variable "Path" has an entry pointing to the root folder where Mathematica is installed (such as C:\Program Files\Wolfram Research\Mathematica\7.0 in Windows for Mathematica 7.0 installation). If there is no such entry, add the Mathematica root folder location to the "Path" environment variable. Follow the steps below to add the Mathematica root folder to Path environment variable value on a Windows machine.

i) Right-click on My Computer, and then click Properties.
ii) Click the Advanced tab.
iii) Click Environment variables.
iv) Lookup the Path environment variable in the list of System variables
v) Select the Path environment variable and click the Edit button.
vi) Add the Mathematica root folder location at the end of the variable value field preceded by a semi-colon. If the existing entry in the variable value field is abc, then after adding the Mathematica root folder location, this field should appear as:

```
abc;C:\Program Files\Wolfram Research\Mathematica\7.0
```



Edit System Variable

Variable name: Path

Variable value:)in\win32;C:\Program Files\Wolfram Resear(

Figure 2: Adding Mathematica installation location to the Path environment variable

See the highlighted section of the Variable value field in the snapshot above.

Note: Do not replace the existing value of Variable value field in step above. Only add the Mathematica root folder location at the end of the existing value.

d) If you are using Mac OS X, ensure that the com.intercax.xaitools.soap.mathematica.mac.path variable in the ParaMagic.ini file points to the location of MathKernel on your Mac. For example: com.intercax.xaitools.soap.mathematica.mac.path=/Applications/Mathematica.app/Contents/MacOS/ MathKernel. Note that Mathematica.app is a package and you will have to explore the package contents to verify the location of MathKernel.

e) Save and close the ParaMagic.ini file.

Option 3. Using a Mathematica installation in your enterprise as a web-service

If you have a Mathematica installation in your organization but not accessible locally, and would like to access it remotely to use ParaMagic^{TM}, please contact us (info@intercax.com) to help you get setup with InterCAX's web-services product (XWS). With XWS, multiple users in your organization can access Mathematica solution services and you can manage the access keys for your organization. Once you have configured XWS, follow the steps below.

a) Open the ParaMagic.ini file.
b) Set the variable com.intercax.xaitools.local.mathematica.true.or.false to false.
c) Set the variable com.intercax.xaitools.soap.mathematica.serverhost to the IP number (of the machine where XWS is installed) followed by a colon followed by the port number. So, the variable value will look something like: 100.100.100.100:8080
d) The XWS administrator in your organization will also create an access key for you. Use this access key to set the value of the access key variable com.intercax.xaitools.soap.mathematica.accesskey.
e) Save and close the ParaMagic.ini file.

Note: If the ParaMagic.ini file is changed during regular usage, the ParaMagic browser must be closed and re-opened for the new settings to take effect. However, MagicDraw does not need to be re-started.

Step 5. OpenModelica (OM) installation

If you have selected OM as the core solver in Step 3, follow the instructions below.

Downloading OpenModelica

The ParaMagic™ download site (www.magicdraw.com/download/paramagic) provides a bundle (zip) that includes the ParaMagic™ plugin and OM installation for Windows and Mac OS X. Download and unzip the bundle to obtain the OM installation for Windows (.msi) or Mac OS X (.dmg).

Alternatively, you can also download the latest versions of OM from the following sites:

1. OpenModelica 1.5.0 RC2 for Windows
   http://www.ida.liu.se/~pelab/modelica/OpenModelica/releases/1.5.0/Windows/rc2/

2. OpenModelica 1.5.0 RC1 for Mac OS X
   http://www.ida.liu.se/~pelab/modelica/OpenModelica/releases/1.5.0/MacOS/

Installing and Configuring OpenModelica (OM) for Windows

Follow the steps below to install and configure OM for Windows.

a) Double click the OpenModelica-1.5.0-RC2.msi file. This will install OM on your machine. If you have an existing OM installation (older version), you will need to uninstall it first. We recommend using the latest version of OM (1.5.0 RC2).
b) Type omc at the command prompt and press enter. Check if OM responds with the version information and command options. If yes, this implies that OM is setup to use.
c) Ensure that OpenModelica is specified as the core solver in ParaMagic.ini file (Step 3 above).
d) After successfully completing Steps 6 and 7 below, open and solve the Addition example (see section 4.2). During solving, ParaMagic™ will indicate that it is using OpenModelica. Successful solution will indicate that OM is installed and configured to work with ParaMagic™.

Installing and Configuring OpenModelica (OM) for Mac OS X

Follow the steps below to install and configure OM for Mac.

a) Double click the OpenModelica150.dmg file.
   - OM on Mac OS X requires XCode[10]. If you do not have XCode installed, the OM installation process will stop and show a message. Download XCode from Apple's website (http://developer.apple.com/technologies/tools/xcode.html), install XCode, and retry installing OM.
b) By default, OM will be installed in your home folder, e.g. /Users/manas/OpenModelica150. It is preferable that you do not move the installed folder.
c) Locate the omc.command file provided by ParaMagic™. This file is located in your MD installation as follows: <your MD installation>/plugins/com.intercax.paramagic/xfw/conf/omc.command
d) Copy the omc.command file and paste it the bin folder of your OM installation, e.g. /Users/manas/OpenModelica150/bin.
e) The plugin installation process changes the read-write-execute permissions for the omc.command file. To resolve this, go to the OpenModelica150/bin folder on your Mac and manually change the permissions of the omc.command file (chmod 777 omc.command) as shown below. All commands are shown in blue.

```
$ cd /Users/manas/OpenModelica150/bin
$ ls -al
total 39664
…
-rw-r--r--   1 …Apr  2 11:49 omc.command
…
$ chmod 777 omc.command
$ ls -al
total 39664
…
-rwxrwxrwx   1 …Apr  2 11:49 omc.command
…
```

[10] XCode - http://developer.apple.com/technologies/tools/xcode.html

f) Open ParaMagic.ini file which is located in your MD installation as follows: <your MD installation>/plugins/com.intercax.paramagic/xfw/conf/ParaMagic.ini
g) Change the value of the com.intercax.xaitools.om.mac.path variable to point to the OpenModelica folder on your Mac. For example,
   com.intercax.xaitools.om.mac.path = /Users/manas/OpenModelica150
h) Change the value of the com.intercax.xaitools.om.mac.exec.path variable to point to a folder for OpenModelica to execute parametric models. It is necessary that (1) there should be no spaces in the path to this folder, and (2) the folder must exist before you use ParaMagic™ with OpenModelica. For example, com.intercax.xaitools.om.mac.exec.path=/Users/manas/temp
i) Save and close the ParaMagic.ini file.
j) After successfully completing Steps 6 and 7 below, open and solve the Addition example (see section 4.2). During solving, ParaMagic™ will indicate that it is using OpenModelica. Successful solution will indicate that OM is installed and configured to work with ParaMagic™.

Step 6. Restart MagicDraw
After restarting MD, your installation is configured to use the ParaMagic™ 16.6 sp1 plugin.

Note: After ParaMagic™ is installed, any further changes to the ParaMagic.ini file do not require you to re-start MagicDraw. Only the ParaMagic™ browser must be re-launched for the changes to take effect.

Step 7. Checking that ParaMagic™ is successfully installed
a) If you select Help → Resource/Plugin Manager, you should see ParaMagic™ Plugin version 16.6 sp1 in the list of installed plugins.
b) If you open any of the models that come with the installation (see Sections 4.2 and 4.3 below) and right click on a package, you should see ParaMagic menu item towards the bottom of the list.

4 User Documents

4.1 Users Guide

1) This document is the Users Guide for ParaMagic™ v16.6 sp1 plugin
2) This Users Guide is also located here after installation:
   <MD_Root>\manual\ParaMagic\Users_Guide.pdf

4.2 Tutorials

1) This installation comes with 6 tutorials. Each tutorial has step-by-step instructions to create a valid SysML model that can be solved using this plugin. All tutorials are described in one document that is located here: <MD_Root>\samples\ParaMagic\Tutorials\Tutorials.pdf
2) Each tutorial also has a pre-built SysML model with instances that are ready to explore, solve, and perform trade studies on instance values. These models are located here: <MD_Root>\samples\ParaMagic\Tutorials
   a) Addition is a SysML model for learning how to create basic parametric equations. Its tutorial includes step-by-step instructions and demonstrates an addition operation applied to system properties.
   b) CommNetwork is a SysML model of a communication network system.
   c) LittleEye is a SysML model of a UAV-based road scanning system named LittleEye.
   d) Satellite is a SysML model of a satellite system.
   e) Orbital is an early-stage orbital mechanics and spacecraft model in SysML that demonstrates the use of ParaMagic-Excel connection (section 7.1) and ParaMagic-Custom Mathematica connection (section 7.1).

f) HomeHeating is a SysML model of a home heating system that demonstrates the execution of Simulink models using the ParaMagic-MATLAB connection (section 7.3).

4.3 Examples

1) This installation comes with 13 example SysML models that are ready to explore, solve, and perform trade studies on instance values.
2) These example SysML models are located here: <MD_Root>\samples\ParaMagic\Other_Examples. A document describing these example models is located here: <MD_Root>\samples\ParaMagic\Other_Examples\Other_Examples.pdf
3) The following example models are provided with ParaMagic^{TM}
   a) Banking is a SysML model of a banking services system.
   b) Circuit is a SysML model of an electrical circuit and demonstrates the application of Ohm's Law.
   c) Distiller is a SysML model of a water distiller. This model is derived from the Distiller model (in MagicDraw) distributed with the SysML book (Friedenthal, Moore, Steiner)[11].
   d) Electronics is a SysML model of an electronics server system.
   e) Financial is a SysML model of financial projections for a small business.
   f) FlapLinkage is a SysML model of Flap Linkage—a part used in an airframe structure. It demonstrates the use of SysML for modeling design constraints as well as building block libraries and other modeling concepts. See item 4) below for a corresponding paper (Part 2) that overviews the theory behind SysML parametrics as well as applications to simulation templates.
   g) Insurance is a SysML model of a web-based insurance claim filing system.
   h) LittleEye_Trade_Study is a model for performing trade studies on the LittleEye example (under Tutorials).
   i) OpAmp is a SysML model of an operational amplifier. By using the same constructs as in the Circuit model, it demonstrates the reusability of concepts defined in SysML.
   j) OrganizationSchemes is a SysML model that describes the different types of packaging structures supported by ParaMagic^{TM} 16.6. It provides examples of how users may organize and share instances across packages.
   k) ProjectPlan is a SysML model of a project scheduling system and demonstrates the use of conditionals (if-then statements).
   l) SpringSystems is a SysML model of a system of two springs connected in series and exhibiting linear deformation behavior. See item 4) below for a corresponding paper (Part 2) that overviews the theory behind SysML parametrics as well as applications to simulation templates.
   m) Trade is a SysML model of a trade financing system.

   Note that the intent of the tutorial and example models is to present how SysML (Parametrics in particular) can be used for different types of problems in different domains. Some models are created for demonstration purposes only, and not intended to represent all aspects of the systems in the most accurate manner.

4) Refer to the following publications to learn more about the SpringSystems and FlapLinkage models and our ongoing research work in SysML model execution in collaboration with Georgia Tech:
   a) Peak, R.S., Roger M. Burkhart, Sanford A. Friedenthal, Wilson, M.W., Bajaj, M. and Kim, I. (2007). *Simulation-Based Design Using SysML Part 1: A Parametrics Primer*. The Seventeenth International Symposium of the International Council on Systems Engineering, San Diego, California, USA June 24 -28, 2007. (available at http://eislab.gatech.edu/pubs/conferences/2007-incose-is-1-peak-primer/)
   b) Peak, R.S., Burkhart, R.M., Friedenthal, S.A., Wilson, M.W., Bajaj, M. and Kim, I. (2007). *Simulation-Based Design Using SysML Part 2: Celebrating Diversity by Example*. The Seventeenth International

[11] A Practical Guide to SysML (Friedenthal, Moore, Steiner): http://www.elsevierdirect.com/product.jsp?isbn=9780123786074
Companion site with link to the original Distiller model: http://www.elsevierdirect.com/companion.jsp?ISBN=9780123786074

Symposium of the International Council on Systems Engineering, San Diego, California, USA June 24 -28, 2007. (available at http://eislab.gatech.edu/pubs/conferences/2007-incose-is-2-peak-diversity/)

5 SysML Model Requirements

This section consists of a list of modeling requirements that need to be satisfied to use the solver capabilities of ParaMagic™ plugin. Some of these requirements are based on recommended practices for enhanced model interoperability; some are to make SysML models less ambiguous for the plugin and Mathematica/OpenModelica solvers; while others are limitations of this version of the plugin and Mathematica/OpenModelica solver. These guidelines are followed in the tutorial examples included with this installation. It is suggested that a user walks through the tutorials first and then review these requirements for better understanding.

Section 5.8 presents steps for upgrading your existing models—that worked with ParaMagic™ 16.5—to work with ParaMagic™ 16.6.

5.1 Structural requirements

These requirements deal with the model schema and instances created using SysML. In our terminology, a schema defines the structure of the model using SysML constructs such as blocks, properties, and constraint blocks; and an instance model conforms to this structure and has populated slots—completely or partially. There may be several instance models for a given schema.

5.1.1 Model schema requirements

1) ParaMagic™ requires that all SysML elements required for defining the schema must be in packages (say schema packages).
2) One of the schema packages is the main schema package. This package consists of the block corresponding to the top-level system in the hierarchy of systems being modeled. We refer to this block as the Root_Block. A Root_Block is a block that is not a part of any other block or is not referenced by any other block. All other blocks in the main schema package and other schema packages are directly or indirectly related to the properties of the Root_Block.
3) Schema packages (including the main schema package) should be of SysML Package type and not SysML Model type. A SysML Model type is a special type of SysML Package type and is visible with this icon in the MagicDraw containment tree. In contrast, SysML Package type has the same icon without the triangle.
4) A model schema may be defined using SysML constructs in several packages (SysML Package type). ParaMagic™ requires that all schema packages should be inside one SysML Model element (SysML Model type—visible with this icon in MD containment tree).
5) A Root_Block is identified using the CXS_heading block (as demonstrated in the tutorials). Once a CXS_heading block is defined in main schema package, a user may validate the schema. Validation is triggered from the main schema package.
6) A constraint block should have only one constraint specification. This version of the plugin does not support multiple constraint specifications defined in one constraint block.
7) If a constraint property (say c1) defined in a block has the same name as an inherited constraint property, ParaMagic™ will override the inherited constraint property with the constraint property c1. Modelers can use this feature to define abstract constraint properties for a block that are implemented by constraint properties of the block's subtypes.

5.1.2 Model instance requirements

1) ParaMagic™ can support several model organization schemes. Five example organization schemes are illustrated using a MD model (OrganizationSchemes.mdzip) located under <MD_Root>\samples\ParaMagic\Other_Examples. Refer to Section 7 in the Other_Examples.pdf file (same location as above) for details.
2) An instance package consists of block instances (a.k.a. InstanceSpecifications) that have populated slots and may refer to other block instances.
3) Instance slots that correspond to block properties participating in parametric relations must be populated unless the lower bound of the multiplicity for these properties is equal to 0 (zero).
4) Instance slots corresponding to value properties must be populated with plain text (Literal String) values and not other types (such as Opaque Expressions). For solving purposes, ParaMagic^{TM} checks if these values contain numbers.
5) Each instance package should have a CXI_heading block (as demonstrated in the tutorials). A CXI_heading block is used to version the instances.
6) Once a CXI_heading block is defined in an instance package, instances validation and browse operations can be triggered from the package.
7) If the value property corresponding to a slot is connected to the outputs of multiple ONEWAY relations, then the slot must have causality "given". This restriction is imposed to prevent instance models from being over-constrained.
8) *Causality Verification and Assignment*: The truth table below states valid causality assignments for slots in a SysML instance model. The validity of a causality assignment is based on whether or not a slot has value(s). In the table, TRUE implies valid assignments and FALSE implies invalid assignments. For example, if a slot has values, then its causality may be given, ancillary, or target but not undefined. Its causality may be ancillary or target only when the model is in a solved state. Similarly, if a slot has no values (empty), then its causality may be undefined or target but not given or ancillary.

Table 1: Truth table for valid causality assignments

	causality			
If a slot	**given**	**undefined**	**ancillary**	**target**
has values	TRUE	FALSE	TRUE	TRUE
does not have values (empty)[12]	FALSE	TRUE	FALSE	TRUE

ParaMagic^{TM} checks for validity of causality assignments when users attempt to browse SysML instance models in the ParaMagic^{TM} browser. The causality assignment utility in ParaMagic^{TM} that is invoked on instance models (ParaMagic→Util→ Add default causalities) assigns default causalities based on this table.

5.2 Naming requirements

1) All SysML block elements should have a unique name, even though they are in different packages.
2) All SysML blocks, constraint blocks, and properties (part / reference / shared / value) should have a name. It is recommended that connectors be also named for enhanced model understanding and debugging but it is not required for ParaMagic^{TM} to work. During validation, ParaMagic^{TM} will automatically assign a name to each connector.

[12] "Empty" denotes that the slot (or slot value) has been activated but does not contain any value (i.e. its value is an empty string and represented as a Literal String). Note that if the slot (or slot value) is not activated, then causality cannot be assigned to it anyway.

3) A block and its parent package cannot have the same name. For example, a block named A cannot be owned by a package named A.
4) A block and a package at the same level cannot have the same name. For example, a block named B and a package named B cannot be at the same level—owned directly by another package.
5) Block properties and constraint parameters should not have names that are reserved math keywords. In addition to names of math constants and functions listed in section 5.4, the following names are also reserved: binom, str.
6) All SysML model elements should have names that start with an alphabetical character (A-Z, a-z).
7) The allowable character classes that can be used in naming model elements are: A-Z, a-z, and 0-9. Underscore ("_") can be used for naming blocks and block properties but not constraint properties.
8) The period/dot operator (.) should not be used in naming model elements. Some of these limitations are due to math parsers and solvers.

5.3 Mathematical expression requirements

1) All value properties in the schema model must be of value type Real—included in the SysML profile in MD—or its subtypes. The values used for populating value properties should be within the bounds[13] specified by Java Double type.
2) If a constraint property is connected to a value property, then its type must be Real.
3) All numeric values in instances must begin with a numerical character 0-9. Values beginning with a space or decimal point may not be read correctly.
4) Mathematical functions (e.g., min, exp) should begin with a lower-case letter and use standard parentheses ( ) to enclose their arguments.
5) A valid mathematical equation for ParaMagic^{TM} processing purposes is one that has a single variable on the LHS. For example, a=b+c will be processed but users will face issues processing the same equation defined as b+c=a.
6) The list of math operators supported[14] in ParaMagic^{TM} are as follows:
   a) Addition (+), Subtraction (-), Multiplication (*), Division (/)
   b) Unary plus (+x), and Unary Minus (-x)
7) It is recommended that large expressions be broken into simpler expressions that are more readily solvable by Mathematica and OpenModelica. For example, the expression below

   ```
   k = (0.577*3.14*E*d) / ln(((1.15*t+D-d)*(D+d)) / ((1.15*t+D-d)*(D-d)))
   ```

   can be broken into two expressions:

   ```
   k = (0.577*3.14*E*d) / ln(dummyVariable)
   dummyVariable=((1.15*t+D-d)*(D+d)) / ((1.15*t+D-d)*(D-d))
   ```

   Since ParaMagic supports only one constraint expression per constraint block, this would require you to create two constraint blocks (one for each expression) and to create a dummy value property (corresponding to the dummyVariable) in the context block.

In general, the math expression syntax supported by ParaMagic^{TM} is based on JEP[15] (with extensions).

[13] http://java.sun.com/javase/6/docs/api/
[14] Note that ParaMagic^{TM} may not warn if other operators are used. Users should only use the math operators stated here.
[15] JEP: http://www.singularsys.com/jep/

5.4 Math constants and functions

ParaMagic™ currently supports the following math constants and functions that may be used in defining constraint specifications.

Some functions are not supported or have a limited support if OpenModelica (OM) is selected as the core solver. These functions have comments in blue.

A list of constants is provided below.

Name	Syntax	Example
Pi(**π**)	pi	y = pi + x

A list of functions is provided below. In principle, several of these functions may be executed in different causalities (directions)—computing LHS value(s) for a given set of RHS values, or computing RHS value(s) for a given set of RHS and LHS values. In the current version of this plugin, these functions and expressions containing these functions are verified to work reliably (i.e. give a correct answer) only in the natural causality (computing LHS value from a given set of RHS values). Hence, these functions are marked as ONEWAY by default when used in a MagicDraw model. If you would like to try solving a function (or the expression it is being used in) in the reverse direction, uncheck the ONEWAY mark for that function (or expression) in the Relationship Browser section of the ParaMagic browser (see Figure 18 in section 6.2) and press Solve. Note that this selection will not be stored in ParaMagic™ settings. This implies that when the browser is launched the next time, the ONEWAY marks for these functions need to be unchecked again. The two primary reasons for marking these functions as ONEWAY by default are:

1) For large expressions involving several functions, Mathematica may not always return an answer when these expressions are evaluated in non-natural directions.
2) The current version of ParaMagic™ does not support inequalities. Without an inequality constraint, some of the functions (esp. trigonometric functions) return a general solution instead of a specific solution. For example x=sin(pi/2) will return the general solution x = 2*n*pi + pi/2. This limitation will be addressed in future release(s) of ParaMagic™.

If OpenModelica is selected as the core solver, none of the functions are marked as ONEWAY. However, this may result in an unexpected (but correct) answer if some of these functions are solved in different directions (due to lack of support for inequalities). For example, solving 1=Sin(x) may not always return x=1.570 (or pi/2).

Name	Syntax	Example
Sine	sin(x)	y = sin(x)
Cosine	cos(x)	y = cos(x)
Tangent	tan(x)	y = tan(x)
Arc Sine	asin(x)	y = asin(x)
Arc Cosine	acos(x)	y = acos(x)
Arc Tangent	atan(x)	y = atan(x)
Arc Tangent (gives the arc tangent of y/x, taking into account which quadrant the point (x, y) is in)	atan2(x,y)	z = atan2(x,y)
Hyperbolic Sine	sinh(x)	y = sinh(x)
Hyperbolic Cosine	cosh(x)	y = cosh(x)
Hyperbolic Tangent	tanh(x)	y = tanh(x)
Inverse Hyperbolic Sine Not supported for OM	asinh(x)	y = asinh(x)
Inverse Hyperbolic Cosine	acosh(x)	y = acosh(x)

Not supported for OM		
Inverse Hyperbolic Tangent Not supported for OM	atanh(x)	y = atanh(x)
Natural Logarithm (ln(x)) (where x is a positive real number)	ln(x)	y = ln(x)
Logarithm ($\log_b x$) (where b and x are positive real numbers) For OM, only log (10,x) is supported	log(b,x)	y = log(b,x)
Exponential	exp(x)	y = exp(x)
Absolute Value	abs(x)	y = abs(x)
Random number (between 0 and 1)	rand()	y = rand()
Modulus	mod(x,y)	z = mod(x,y)
Square Root	sqrt(x)	y = sqrt(x)
Power x^y	pow(x,y)	z = pow(x,y)
Round (rounds argument to the closest integer, or the closest even integer for arguments equidistant from two integers)	round(x)	y = round(x)
Ceil (rounds argument to the smallest integer greater than or equal to the argument)	ceil(x)	y = ceil(x)
Floor (rounds argument to the greatest integer less than or equal to the argument)	floor(x)	y = floor(x)
Min (returns the argument with minimum value)	min(x_1,x_2,x_3,...) For Mathematica, x_1,x_2,...can be arrays or single-valued For OM, x_1,x_2,...can only be single-valued	y = min(x_1,x_2,x_3)
Max (returns the argument with maximum value)	max(x_1,x_2,x_3,...) For Mathematica, x_1,x_2,...can be arrays or single-valued For OM, x_1,x_2,...can only be single-valued	y = max(x_1,x_2,x_3)
Average (returns the arithmetic mean of members of the array passed as argument)	avg(x) where x is an array	y = avg(x)
Sum[16] (returns the sum of the members of the array passed as argument)	sum(x) where x is an array	y = sum(x)

5.5 Conditional Functions and Operators

ParaMagic™ currently supports conditional statements in the following format.

<Result> = if(<Condition>, <Result if Condition = true>, <Result if Condition = false>)

For example, in the conditional statement
X2 = if(X1 > 0, X1, -X1)

X2 is set equal to X1 when X1 is positive and –X1 when X1 is negative. Hence, this condition is the equivalent of X2 = abs(X1).

The following operators can be used as part of the condition term

[16] In ParaMagic 16.0, this function accepted multiple single-valued arguments but in ParaMagic 16.6, this function accepts only one multi-valued argument.

Operator	Syntax	Example
Equal to	==	y = if(x == 1, 2, 1)
Not Equal to	!=	y = if(x != 1, 2, 1)
Greater than	>	y = if(x > 1, 2, 1)
Less than	<	y = if(x < 1, 2, 1)
Greater than or Equal to	>=	y = if(x >= 1, 2, 1)
Less than or Equal to	<=	y = if(x <= 1, 2, 1)
AND	&&	y = if(x1 == 0 && x2 < 5, 2, 1)
OR	||	y = if(x1 == 0 || x2 < 5, 2, 1)
NOT	!	y = if(!(x>1), 2, 1)

5.6 Aggregate Properties and Functions

ParaMagic™ also supports aggregate value properties of type Real, that is, properties that contain a list of one or more values of type Real.

5.6.1 Multiplicity

The default multiplicity for value properties is 1. In order to hold more than one value, the multiplicity of the value property must be set to 1..* , 0..*, or * in the model schema.

5.6.2 Instance Slot Values

When setting up an instance for solution, values are assigned to slots for each variable—either real values for given values or empty values as placeholders for unknowns. To add multiple values to a slot, create the first value as normal in the Instance Specification window, then use the Plus icon at the bottom left of the window to add additional values one at a time. Multiple empty slot values can be created in the same way.

5.6.3 Aggregate Functions and Operators

The following function and operators are supported by ParaMagic™ for aggregate values.

Some functions are not supported or have a limited support if OpenModelica (OM) is selected as the core solver. These functions have comments in blue.

Function	Explanation
y = a[2]	a is an aggregate of Real numbers and y is a single Real number; y = second item in aggregate a; parameterized indices not supported (a[i] not supported)
y = sum(a)	a is an aggregate of Real numbers and y is a single Real number; y = a1 + a2 + a3 +…
y = avg(a)	a is an aggregate of Real numbers and y is a single Real number; y = mean of a1, a2, a3,…
y = standarddeviation(a) Not supported for OM	a is an aggregate of Real numbers and y is a single Real number; y = standard deviation of a1, a2, a3,…
y = variance(a) Not supported for OM	a is an aggregate of Real numbers and y is a single Real number; y = variance of a1, a2, a3,…
y = min(a)	a is an aggregate of Real numbers and y is a single Real number; y = minimum of a1, a2, a3,…
y = max(a)	a is an aggregate of Real numbers and y is a single Real number y = maximum of a1, a2, a3,…

x = a	x and a are aggregates of Real numbers; {x1, x2, x3,…} = {a1, a2, a3 …}. Note that the sizes of x and a must be same.
x = a + b	x, a, and b are aggregates of Real numbers; {x1, x2, x3…} = {a1+b1, a2+b2, a3+b3 …}; similar syntax for other math operators and functions. Note that the sizes of x, a, and b must be same.
x = sin(a)	x and a are aggregates of Real numbers; {x1, x2, x3…} = {sin(a1), sin(a2), sin(a3) …}; similar syntax for other trigonometric, exponential, logarithmic, and hyperbolic functions. Note that the sizes of x and a must be same.
x = n Not supported for OM	x is an aggregate of Real numbers and n is a single Real number; {x1, x2, x3…} = {n,n,n…}.
x = a + n Not supported for OM	x and a are aggregates of Real numbers, and n is a single Real number; {x1, x2, x3…} = {a1+n, a2+n, a3+n…}; similar syntax for other math operators and functions. Note that the sizes of x and a must be same.
x = pow(n,a) Not supported for OM	x and a are aggregates of Real numbers, and n is a single Real number; {x1, x2, x3…} = {n^a1,n ^a2,n^ a3 …}. Note that the sizes of x and a must be same.

Equations involving both single-valued variables/constants and multi-valued (array) variables solve in Mathematica 7.0 but may not solve in Mathematica 6.0 or 5.2. For example, the following equations may not solve in Mathematica 5.2 or 6.0 when one of the variables (x or a) is given and the other is computed.

1) x = a, where x is an aggregate and a is a single-valued variable
2) x = a + n, where x and a are aggregates and n is a single-valued variable
3) x = a + pi(), where x and a are aggregates
4) x = a + rand(), where x and a are aggregates

For OpenModelica as the core solver, math expressions combining aggregates and single-valued variables are not supported. For example, x = a + n, where x and a are aggregates and n is a single-valued variable is not supported.

5.7 Complex Aggregate Relationships

ParaMagic^{TM} 16.6 sp1 has enhanced support for complex aggregate relationships. First, a short description of complex aggregates and complex aggregate relationships is presented to better characterize this feature of ParaMagic^{TM}.

A *primitive aggregate* is a collection of primitive elements, such as real numbers, strings, integers, etc. For example, in the relation a = avg(b), where a is a single-valued real number and b is an array of real numbers, b is a primitive aggregate. Primitive aggregates are manifested in the SysML model when value properties have multiplicity > 1. ParaMagic currently supports several types of relations involving primitive aggregates—see section 5.6 above.

A *complex aggregate* is a collection of blocks (in SysML context). Complex aggregates are manifested in a SysML model when part/reference/shared properties of a block have multiplicity > 1. For example, Figure 4 below illustrates a car system composed of several sub-systems, including the chassis sub-system. The Chassis is composed of several components. The parametric diagram in Figure 5 illustrates the relationship between the mass of the Chassis block and the masses of its components.

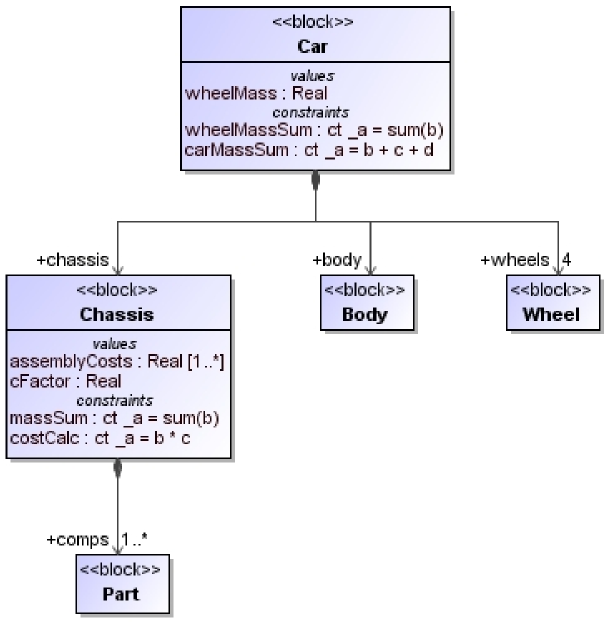


Figure 3: Car example to illustrate complex aggregates

The constraint relationship embodied in the massSum constraint property is an example of a complex aggregate relationship. This is because one of the parameters in the relationship is tied to a value property that is owned directly/indirectly by a complex aggregate (e.g. parameter b is tied to value property mass owned by complex aggregate comps).

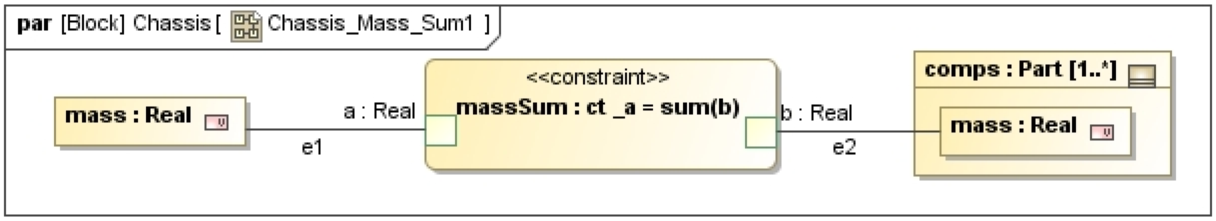


Figure 4: Parametric diagram illustrating the relationships between properties of the Chassis block (Figure 4)

In ParaMagic^{TM}, the depth of a complex aggregate relationship is defined as the maximum level of nesting of the properties (owned by complex aggregates) constrained by that relationship. For example, the depth of the complex aggregate relationship shown in Figure 5 above is equal to 1. The properties involved in the relationship are mass and comps[i].mass. The level of nesting of mass is 0 while that of comps[i].mass is 1. The level of nesting also corresponds to the number of complex aggregate part boundaries crossed by the binding connector that ties the parameters of the relationship with the block properties. For example, the connector e2 in Figure 5 crosses only 1 complex aggregate boundary (of comps). Similarly, Figure 6 below illustrates a complex aggregate relationship (loadSum) with depth = 2. As evident, the connector e2 crosses two complex aggregate part boundaries (floors and parkedCars). In Figure 6 below, if the multiplicity of floors was changed to 1, then the connector e2 would cross only 1 complex aggregate part boundary (parkedCars). Hence, the depth of the complex aggregate relationship would change from 2 to 1.

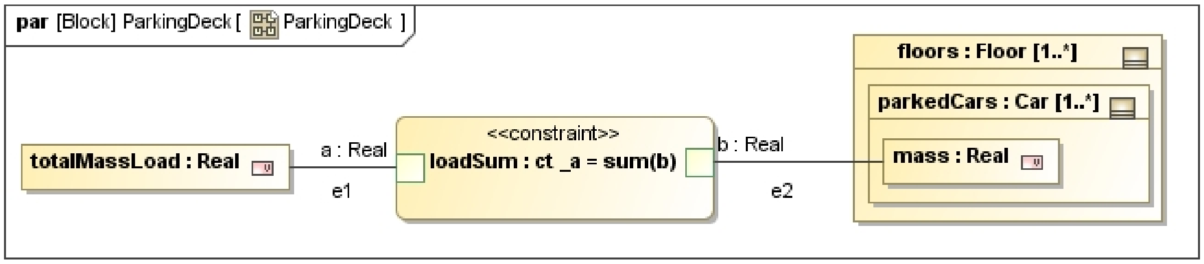


Figure 5: Parametric diagram illustrating a complex aggregate relationship of depth = 2.

ParaMagic^{TM} 16.6 sp1 only supports execution of complex aggregate relationships with depth = 1. In general, the parameters of the complex aggregate relationship (and the properties they are connected to) can be single-valued or multi-valued (e.g. array). However, properties owned by complex aggregates and the relationship parameters they are tied to should be single-valued. For example, in the complex aggregate relationship illustrated in Figure 5, the property comps[i].mass is owned by the complex aggregate comps. Hence, comps[i].mass and the parameter b to which it is connected should be single-valued (as shown) for ParaMagic to solve this relationship. Figure 7 below illustrates a complex aggregate relationship (costCalc) with depth = 1. ParaMagic^{TM} 16.6 sp1 can solve this relationship because the property comps[i].mass owned by the complex aggregate comps is single-valued.

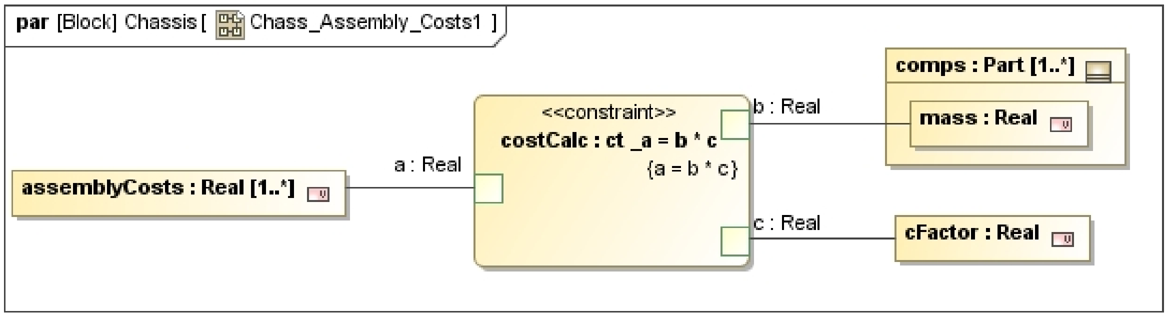


Figure 6: Parametric diagram illustrating a complex aggregate relationship with depth = 1.

ParaMagic^{TM} 16.6 sp1 does not directly support complex aggregate relationships that wrap MATLAB M-files (section 7.3) or custom Mathematica functions (section 7.1). This is because it is required that the parameters of the constraint properties wrapping MATLAB M-files (or custom Mathematica functions) be bound to block (value) properties that are primitives—single-valued or aggregates. As a workaround to this limitation, users can define an equality relation that transforms the single-valued primitive properties owned by a complex aggregate to a property that is a primitive aggregate. Then, this property can be tied to the parameters of the constraint property wrapping MATLAB M-files (or custom Mathematica functions). Figure 8 below illustrates this workaround. If the properties ppA[i].vp1 and ppB[i].vp2 were directly connected to the parameters input1 and output of the constraint property cp1 (that wraps MATLAB M-file function), then the MATLAB relationship in cp1 could not have been executed. To avoid this, two extra primitive aggregate properties vp1 and vp2 are created. Then, the equality relations cp2 and cp3 are used to connect these primitive aggregates to the properties ppA[i].vp1 and ppB[i].vp2. With this workaround, the MATLAB relationship in cp1 can be executed.

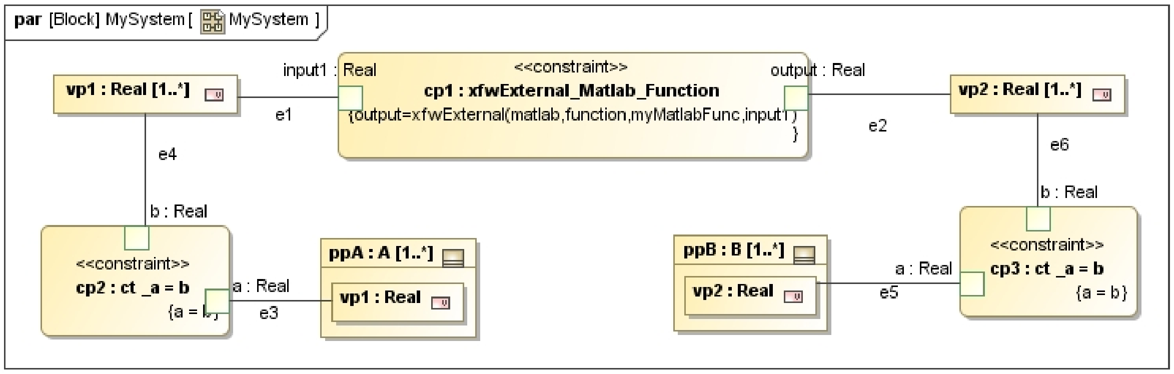


Figure 7: Workaround for complex aggregate relationships that wrap M-files or Mathematica functions.

In ParaMagic™ 16.6 sp1, complex aggregate relationships are not supported for OpenModelica as the core solver.

5.8 Upgrading your existing models for ParaMagic™ 16.6

Follow the steps below to upgrade your existing models—that worked with ParaMagic™ 16.5—to work with ParaMagic™ 16.6. Note that update 2 and 3 may not apply to your models. It is recommended that you try browsing and solving your model after update 1. If your model does not validate and the reasons are due to aspects mentioned in update 2 and 3 below, then ParaMagic™ will point to the model elements that need to be updated.

1) **Using SysML String value type instead of UML String datatype for value properties**
   Per SysML, all value properties must be typed by a value type. However, SysML 1.1 specifications did not provide a value type for representing strings. If your model was created in MagicDraw SysML 16.5 or earlier, it may be using the String datatype (part of UML specs) for some value properties. MagicDraw SysML 16.6 implements[17] SysML 1.2 RTF submission that provides the value type String as part of its model library, and also flags model elements that use the UML String datatype. If you open your model in MagicDraw SysML 16.6, you may see warnings as below (Figure 9).

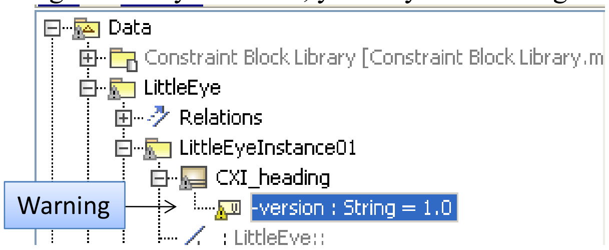


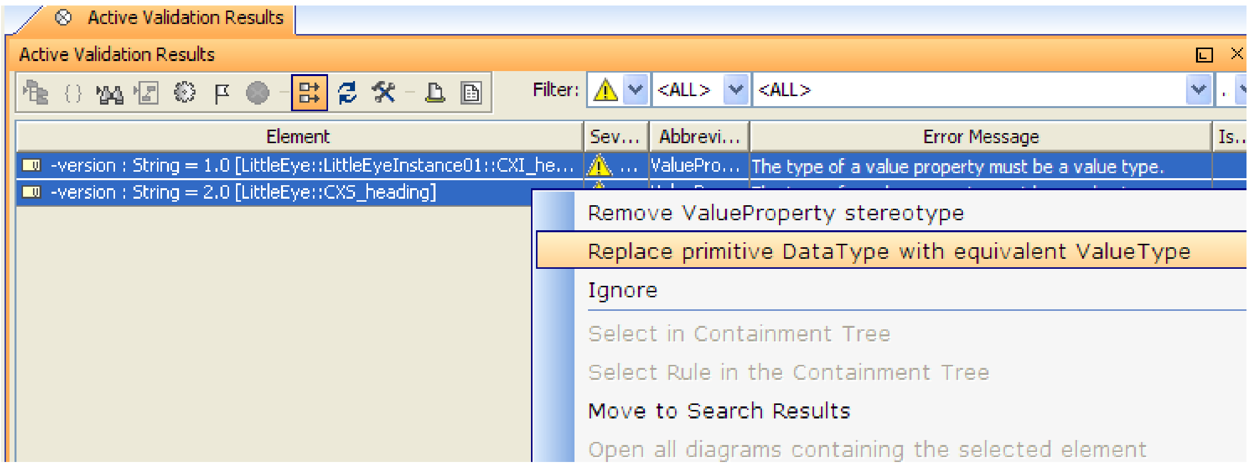
Figure 8: Warnings shown in MD SysML 16.6 containment tree if value properties do not use a value type.

To use String value type for the subject model elements, follow the steps below:

a) Click on the warnings icon on the bottom right hand corner of the MagicDraw window.
b) It will bring up the validation results, as below.
c) Select validation warnings that are due to value properties using UML String datatype instead of SysML String valuetype. In Figure 10 below, both the warnings are due to this problem.
d) After selecting the relevant validation warnings, right click and select Replace primitive DataType with equivalent ValueType, as shown in Figure 10 below.

[17] for most purposes relevant to ParaMagic™

e) This action will change the type of the subject value properties from UML String datatype to SysML String value type.


Active Validation Results
Active Validation Results
Filter:
<ALL>
<ALL>
Remove ValueProperty stereotype
Replace primitive DataType with equivalent ValueType
Ignore
Select in Containment Tree
Select Rule in the Containment Tree
Move to Search Results
Open all diagrams containing the selected element

Element	Sev...	Abbrevi...	Error Message	Is..
-version : String = 1.0 [LittleEye::LittleEyeInstance01::CXI_he...	...	ValuePro...	The type of a value property must be a value type.	
-version : String = 2.0 [LittleEye::CXS_heading]				

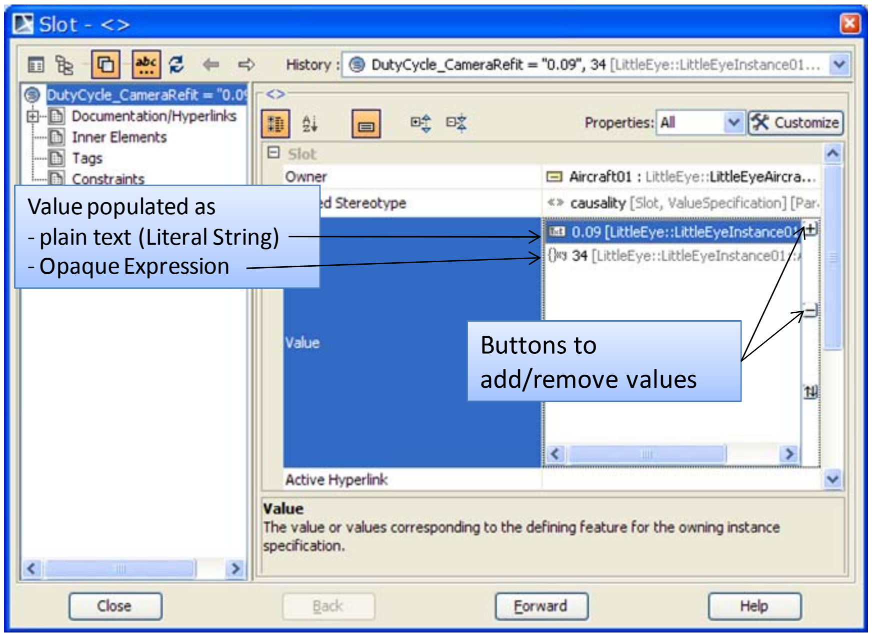
Figure 9: Automatically updating value properties to use String value type instead of String datatype

2) Instance slots corresponding to value properties must be populated as plain text (Literal String)
This problem may surface only if your model was created in much earlier versions of MagicDraw (~15.5 or earlier). Recent versions of MagicDraw (16.0 and later) populate slot values corresponding to value properties as plain text (Literal String) by default when values are added manually to the slot.

ParaMagic™ requires that instance slots corresponding to value properties must be populated as plain text (Literal String) and not in other formats (such as Opaque Expressions). Refer to section 5.1.2, item 4. ParaMagic™ 16.6 sp1 enforces this and will catch such cases in SysML instance models during validation/browsing. Figure 11 below shows the specification view for a slot. Note that the slot has 2 values. The first value is populated in plain text (Literal String)— indicated by the icon Txt, and the second value is populated as an Opaque Expression—indicated by the icon {}xy.
If ParaMagic™ 16.6 sp1 complains about this problem when attempting to browse your existing models, you will need to re-enter the value(s) for the subject slots as Literal String. To do this, follow the steps below:

a) Select the subject value and press the Remove value button (as shown in Figure 11).
b) Click on the add value button (as shown in Figure 11) and select Literal String.
c) Renter the value.

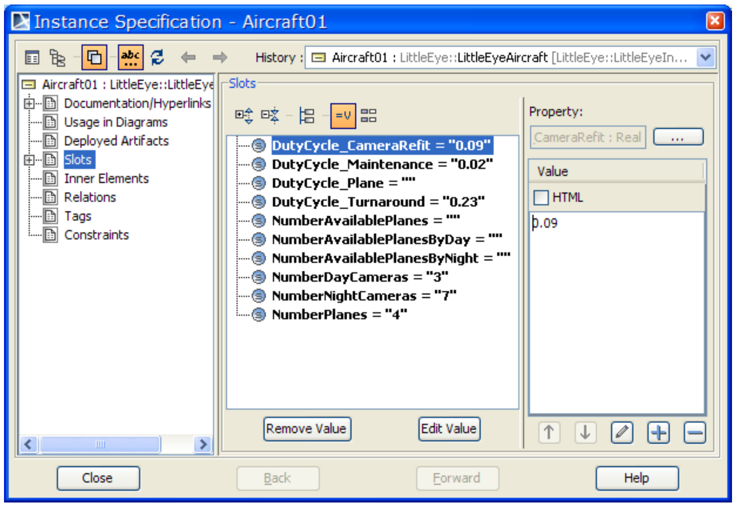
If the subject slot has multiple values that have the same problem, you can delete all values, configure the slot and read values from an Excel spreadsheet.


Slot - <>
History : DutyCycle_CameraRefit = "0.09", 34 [LittleEye::LittleEyeInstance01...
DutyCycle_CameraRefit = "0.09"
Documentation/Hyperlinks
Inner Elements
Tags
Constraints
Properties: All
Customize
Slot
Owner
Aircraft01 : LittleEye::LittleEyeAircra...
Applied Stereotype
causality [Slot, ValueSpecification] [Par...
0.09 [LittleEye::LittleEyeInstance01
34 [LittleEye::LittleEyeInstance01
Value
Active Hyperlink
Value populated as
- plain text (Literal String)
- Opaque Expression
Buttons to add/remove values
Close
Back
Forward
Help

Value
The value or values corresponding to the defining feature for the owning instance specification.

Figure 10: Instance slots corresponding to value properties must be populated as plain text (Literal String)

Note that if you are populating slots from the Instance specification window (Figure 12 below), you may not see an option for selecting different formats (such as Literal String and Opaque Expression). By default, MagicDraw adds slot values as Literal String.



DutyCycle_CameraRefit = "0.09"
DutyCycle_Maintenance = "0.02"
DutyCycle_Plane = ""
DutyCycle_Turnaround = "0.23"
NumberAvailablePlanes = ""
NumberAvailablePlanesByDay = ""
NumberAvailablePlanesByNight = ""
NumberDayCameras = "3"
NumberNightCameras = "7"
NumberPlanes = "4"

0.09

Figure 11: Slot values as seen in the Instance Specification window

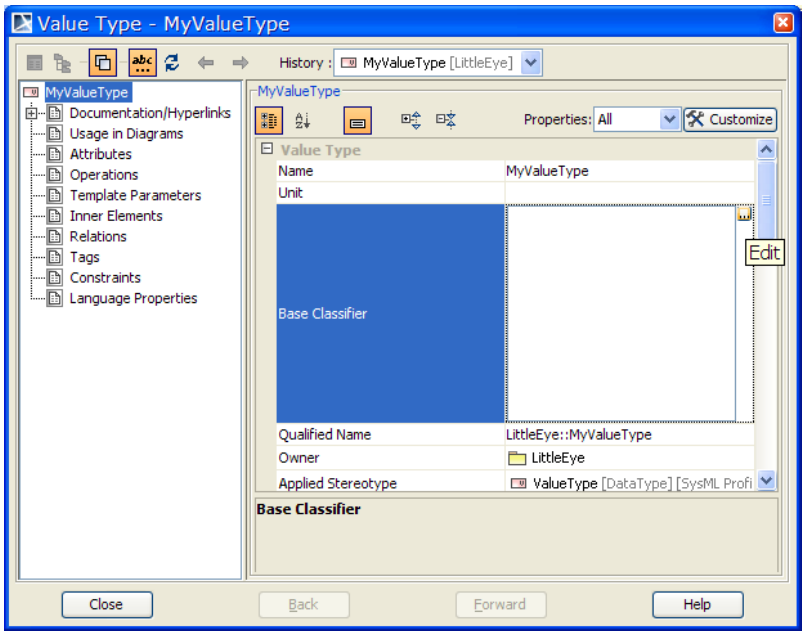
3) **Value properties participating in parametric relations must be typed by the SysML Real value type or its subtypes**

For solving purposes, ParaMagic^{TM} requires that all value properties that are participating in the parametric models should be typed by the SysML Real value type or its subtypes. This has been the case since earlier versions of ParaMagic^{TM}. Refer to section 5.3, item 1. However, ParaMagic^{TM} 16.6 sp1 enforces this and will catch such cases in SysML models during validation/browsing. If you have defined custom value types, make sure that they are subtypes of the SysML Real value type. To do this, follow the steps below:

a) Double click the custom value type in the MD containment tree. This will open the specification window as shown in Figure 13 below.
b) Edit the Base Classifier field by clicking on the Edit icon, as shown in Figure 13 below.

Copyright © InterCAX LLC. All Rights Reserved.

c) This will bring up the selection window. Select Real under SysML Profile::Blocks package, as shown in Figure 14 below.
d) Press the OK button in the selection window. Then, press the Close button in the specification window.



Value Type	
Name	MyValueType
Unit	
Base Classifier	
Qualified Name	LittleEye::MyValueType
Owner	LittleEye
Applied Stereotype	ValueType [DataType] [SysML Profi

Base Classifier

Figure 12: Specifying custom value types as subtype of SysML Real value type

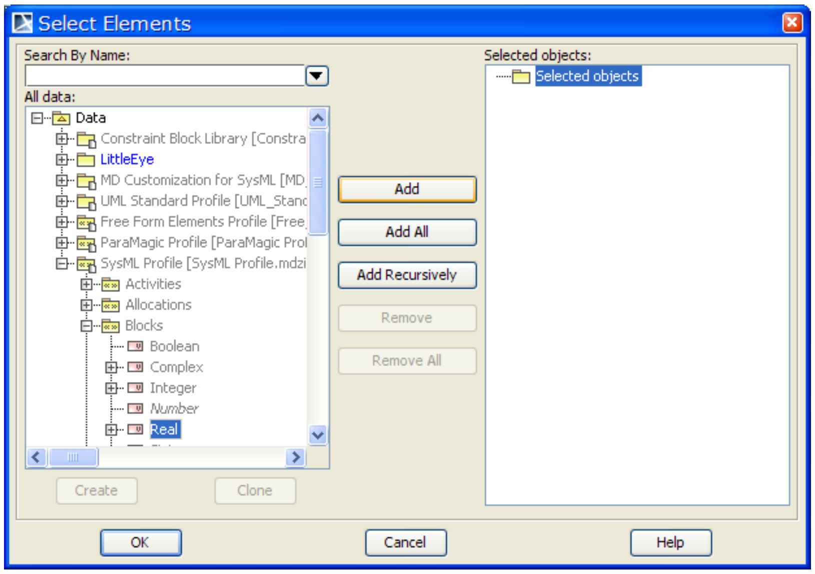


Figure 13: Select SysML value type Real

Alternatively, you could create a block definition diagram, show the custom value type and the SysML Real value type in the diagram, and then create a Generalization relationship from the custom value type to the SysML Real value type.

5.9 Limitations

ParaMagic™ 16.6 sp1 has the following limitations with respect to SysML models.

1) OpenModelica (OM) as the core solver – If OM is selected as the core solver, the following features are not supported:
   a) ParaMagic – Custom Mathematica Connection (section 7.1)
   b) ParaMagic – MATLAB Connection (section 7.3)
   c) Complex aggregate relationships (section 5.7)
   d) Some math functions are not supported or have a limited support if OM is selected as the core solver. These functions are commented in blue in sections 5.4, 5.5, and 5.6.

2) Constraint block-related limitations - This version of ParaMagic^{TM} does not support:
   a) multiple constraint specifications in a constraint block.
   b) constraint blocks composed of other constraint blocks.
   c) binding connectors between constraint parameters belonging to two different constraint blocks. A user must create an additional value property to achieve this.
   d) inequality constraints.

3) Cyclic references-related limitations - This version of ParaMagic^{TM} does not support some types of cyclic references among model instance elements. Specifically, the following scenarios are not supported:
   a) A block instance has a slot that it is populated with the instance itself, i.e. the instance points to itself.
   b) Given two block instances, each has a slot that is populated with the other instance. For e.g. instance A points to instance B, and instance B points to instance A.

   Note that a block may have a property typed by the block itself. This looping structure at the block level is supported if there is no looping at the instance level.

4) Units - This version of ParaMagic^{TM} does not support units and automated unit conversions in math expressions. Thus, instance validation and solution does not check for the consistency of units. Users must provide instance values with consistent units. During model creation, MagicDraw automatically checks if the value properties being connected in a parametric model have the same dimensions (e.g. mass) but it will not check if two value properties with same dimensions but different units (e.g. grams and kilograms) are connected together.

5) Complex numbers – This version of ParaMagic^{TM} does not support complex numbers. If solution results return a complex number for a variable, a string No Value is displayed for that variable in the ParaMagic^{TM} browser.

6) Complex Aggregate Relationships – The following limitations exist for complex aggregate relationships in ParaMagic^{TM}. See section 5.7 for details.
   a) Only complex aggregate relationships with depth = 1 are supported.
   b) Constraint blocks that wrap a MATLAB M-file or a custom Mathematica function cannot be used directly with complex aggregates for some cases. See section 5.7 for details on a workaround for this limitation.

7) If the value property corresponding to a slot is connected to the outputs of multiple ONEWAY relations, then the slot must have causality "given". This restriction is imposed to prevent instance models from being over-constrained—conflicting values of such slots may be computed from the different ONEWAY relations.

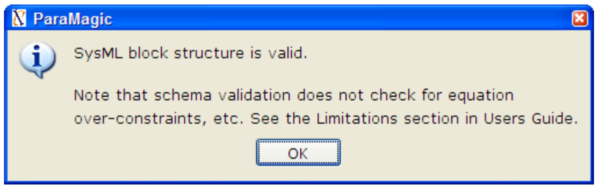
8) Unless otherwise specified, all slot values in MagicDraw SysML models are internally stored as Literal String. ParaMagic™ only supports values of this default type and converts them to real numbers when solving parametric constraints.

9) This version of ParaMagic does not support scientific notation for numbers.

10) If you change the causality of a solved target variable to given in MagicDraw and launch the ParaMagic™ browser, you may see an over-constrained set of values. ParaMagic™ does not check for over-constrained instance values during validation. To resolve such states, users should identify a new target variable and re-solve the model.

11) When you successfully validate the schema structure of a model (see Validate in section 6.1), you will receive the message in Figure 15.
    a) The warning is to inform you that this validation function effectively checks the syntax and connectivity of the structure of your SysML model schema as far as parametrics solving goes.
    b) At this stage ParaMagic™ does not check for over-constrained equations and similar non-syntactic issues. In general the math solver that you can later invoke via the ParaMagic™ Browser will uncover such issues (usually by returning a "No Value" result or an over-constraint warning.

12) When you successfully validate the instance structure of a model (see Validate in section 6.1), you will receive the message in Figure 16.
    a) The warning is to inform you that this validation function effectively checks the syntax and connectivity of the structure of your SysML model instance as far as parametrics solving goes.
    b) ParaMagic™ does not check for over-constrained values or inconsistent causalities and similar non-syntactic issues at the instance validation stage. Such issues will be uncovered during the instance solution stage, usually by returning a No Value result (in the ParaMagic™ browser) or an over-constraint warning.

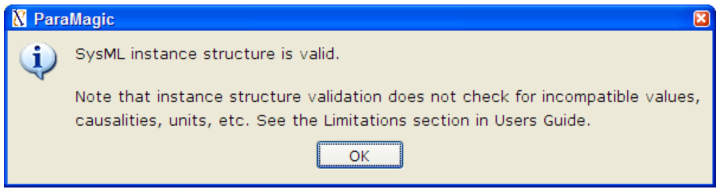

ParaMagic

SysML block structure is valid.

Note that schema validation does not check for equation over-constraints, etc. See the Limitations section in Users Guide.

OK

Figure 14: Structure validation notice


ParaMagic

SysML instance structure is valid.

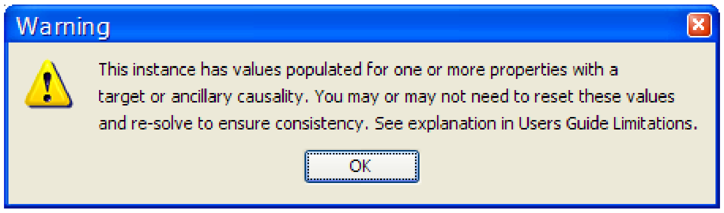
Note that instance structure validation does not check for incompatible values, causalities, units, etc. See the Limitations section in Users Guide.

OK

Figure 15: Instance validation notice

13) When you open an existing instance from MagicDraw for browsing (see Browse in section 6.1) you will see the warning message in Figure 17 if that instance has values for any attributes with causality.
    a) Usually this situation exists because you have stored previously solved values in the SysML model. Usually it is not a problem and you can browse the instance to review its results without concern (if you fully control the model and know its change history), *but in general we recommend re-setting the instance and re-solving it to be safe.*
    b) There are several ways that previously solved instance values can become invalid (because the instance in MagicDraw is not continuously connected to the instance in the ParaMagic™ browser), including:
        i) Someone changed the corresponding schema structure in MagicDraw after that instance was solved (e.g., they added a new equation or changed an existing equation).
        ii) Someone changed an attribute value or an attribute causality in the instance in MagicDraw. In general we recommend not making such changes in MagicDraw for a previously solved instance. Instead, open the ParaMagic™ browser and then make such changes.

In other words, for example if you make structural changes to your SysML model schema, such as redrawing the connectors in a parametric diagram, you can still browse the old solved instance (conforming to the schema before changes) in ParaMagic^{TM}. With the changes in the model, the values in old instances may be non-conformant to the model schema. This version of ParaMagic^{TM} does not automatically re-solve (update) old instances to conform with the new schema. It is recommended that after structural changes to the model schema, users should (a) re-validate the schema, (b) open the ParaMagic^{TM} browser, (c) Reset all non-given values (automatically by pressing the Reset button), and (d) solve the instance again.



This instance has values populated for one or more properties with a target or ancillary causality. You may or may not need to reset these values and re-solve to ensure consistency. See explanation in Users Guide Limitations.

Figure 16: Warning regarding previously solved instance values.

14) Mathematica solving issues:
    a) If a system of equations is under-constrained, ParaMagic^{TM} browser shows that no value is available for one or more target variables after solving. It does not explicitly warn the user that the system of equations was under-constrained.
    b) In some cases, while solving a large system of equations in the reverse direction (non-natural direction), Mathematica returns an over-constrained set of values (without exceptions or warnings). Contact us (*support@intercax.com*) for further information.
15) Teamwork server issues: Before updating SysML instance values, ParaMagic^{TM} does not check if the instance is locked by other users sharing that SysML model via MagicDraw Teamwork server.

6 PROGRAM FEATURES

6.1 Command Menus

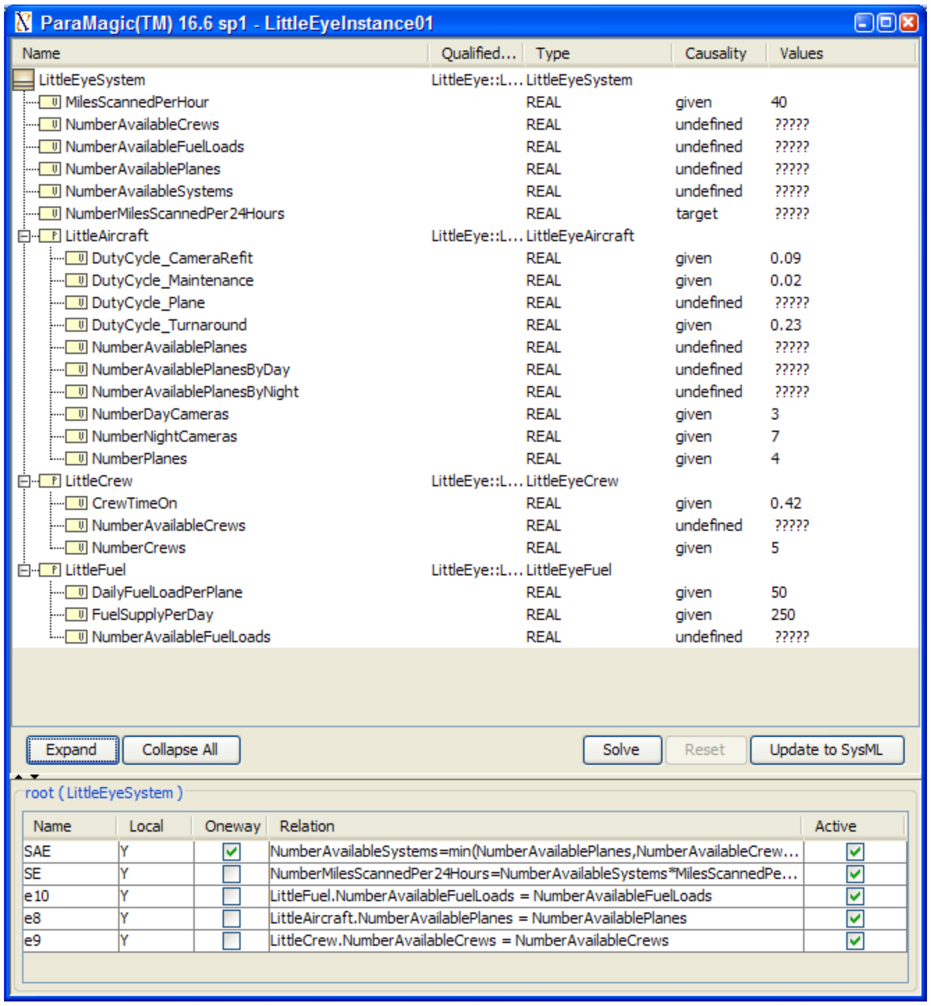
ParaMagic^{TM} commands are applied by right-mouse-clicking on the appropriate block or package in the MagicDraw containment tree and selecting the menu item ParaMagic.

Command	**Description**
Validate	• Identifies and versions the Root_Block of the model, creates the block CXS_heading, and validates the SysML model, when applied to the package containing the block. • Versions and validates a model instance before it is solved, when applied to a model instance package.
Browse	Opens a browser for the parametric model (see section 6.2), when applied to an instance of the model
Util → Add default causalities	Assigns default causality to each variable in the parametric model. Variables that have been assigned a value are tagged as given, and variables without values are tagged as undefined. At least one variable must be assigned manually as target to solve the model.
Util → Create CXS_heading	Creates a CXS_heading block when applied to the Root_Block. User will be asked for a string value to identify the version of the model schema
Util → Create CXI_heading	Creates a CXI_heading block when applied to an instance of the model. User will be asked for a string value to identify the version of the instance.

Excel → Setup	Launches the Excel setup utility to connect block instances to Excel spreadsheets.
Excel → Read from Excel	Reads values from Excel spreadsheet(s) into the SysML model in MagicDraw.
Excel → Write to Excel	Writes values from SysML model in MagicDraw to Excel spreadsheet(s).
Trade Study → Setup	Launches the trade study setup utility.
Trade Study → Run	Runs a trade study.

6.2 Browser

The ParaMagic^{TM} plugin browser displays the parametric model variables and controls for solving and displaying the variable values. An example of the browser window is shown in Figure 18 below. The browser has been updated in ParaMagic™ 16.6 sp1 to display properties in the following order—value, part, shared, and reference.



Name	Qualified...	Type	Causality	Values
LittleEyeSystem	LittleEye::L...	LittleEyeSystem		
MilesScannedPerHour		REAL	given	40
NumberAvailableCrews		REAL	undefined	?????
NumberAvailableFuelLoads		REAL	undefined	?????
NumberAvailablePlanes		REAL	undefined	?????
NumberAvailableSystems		REAL	undefined	?????
NumberMilesScannedPer24Hours		REAL	target	?????
LittleAircraft	LittleEye::L...	LittleEyeAircraft		
DutyCycle_CameraRefit		REAL	given	0.09
DutyCycle_Maintenance		REAL	given	0.02
DutyCycle_Plane		REAL	undefined	?????
DutyCycle_Turnaround		REAL	given	0.23
NumberAvailablePlanes		REAL	undefined	?????
NumberAvailablePlanesByDay		REAL	undefined	?????
NumberAvailablePlanesByNight		REAL	undefined	?????
NumberDayCameras		REAL	given	3
NumberNightCameras		REAL	given	7
NumberPlanes		REAL	given	4
LittleCrew	LittleEye::L...	LittleEyeCrew		
CrewTimeOn		REAL	given	0.42
NumberAvailableCrews		REAL	undefined	?????
NumberCrews		REAL	given	5
LittleFuel	LittleEye::L...	LittleEyeFuel		
DailyFuelLoadPerPlane		REAL	given	50
FuelSupplyPerDay		REAL	given	250
NumberAvailableFuelLoads		REAL	undefined	?????

Name	Local	Oneway	Relation	Active
SAE	Y	✓	NumberAvailableSystems=min(NumberAvailablePlanes,NumberAvailableCrew...	✓
SE	Y	☐	NumberMilesScannedPer24Hours=NumberAvailableSystems*MilesScannedPe...	✓
e10	Y	☐	LittleFuel.NumberAvailableFuelLoads = NumberAvailableFuelLoads	✓
e8	Y	☐	LittleAircraft.NumberAvailablePlanes = NumberAvailablePlanes	✓
e9	Y	☐	LittleCrew.NumberAvailableCrews = NumberAvailableCrews	✓

Variable Browser (shown in expanded form)

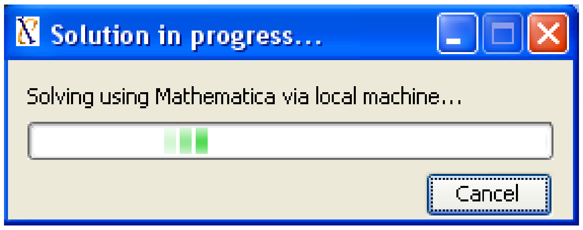
Toolbar

Relationship Browser

Figure 17: Browser window

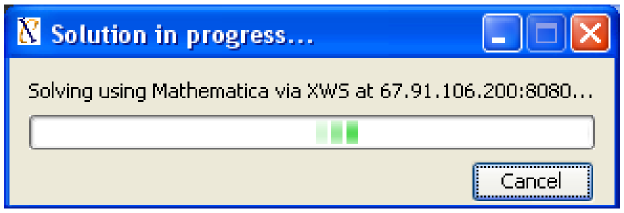
6.2.1 "Solution in progress" Window

When the Solve button on the browser is clicked, the network of parametric equations is exported to Mathematica and ParaMagic^{TM} waits for the results to be returned. During this interval, a Solution in progress window is displayed, as shown in Figure 19 and Figure 20 below.



Solving using Mathematica via local machine...

Figure 18: Solution in progress window when using local Mathematica



Solving using Mathematica via XWS at 67.91.106.200:8080...

Figure 19: Solution in progress window when using remote Mathematica

The Cancel button causes ParaMagic™ to ignore any intermediate results returned from Mathematica, to reset the model values (same as pushing Reset), and to return to the ready-to-solve condition. It does not necessarily affect Mathematica directly (i.e., the Mathematica job may continue to execute, in which case ParaMagic™ will ignore any results it obtains).

If the Mathematica job itself hangs, it may require manual intervention before you can continue successfully solving models via ParaMagic™. To cancel a Mathematica job, consult your Mathematica Users Guide (for local solver installation) or contact your server administrator (for an XWS/web services-based solver installation).

6.2.2 Variable Browser

Each variable is shown with Name, Type, Causality and Value. Variable causality can be changed in the browser window by clicking on the causality type and selecting from the list. Four possibilities are available for Causality. A user can only specify causality to be Given, Undefined, or Target. If a variable with initial causality Undefined is used to compute other variables with during solution, its causality will automatically change to Ancillary after solving.

Command	**Description**
Given	Value is assigned before solving
Undefined	Value may be calculated during solving if it is needed to determine a Target, directly or indirectly.
Target	Value is calculated during solving (if Mathematica can find a valid solution). At least one unknown must be assigned as a target variable to initiate the solution process.
Ancillary	A variable whose value is calculated during solving and used to calculate the value of another variable. The value of this variable is not available before solving.

The value of a Given variable can be changed by clicking on its value and editing it. All other variables must be changed to Given before they can be edited (see 6.2.5 for more about this).

The size of the Variable Browser window may be expanded by 1) dragging down the lower border of the Browser window, followed by 2) dragging down the horizontal black line between the Toolbar and Relationship Browser.

6.2.3 Toolbar

Command	**Description**
Expand	Expands all blocks in the variable browser one tree level per button click
Collapse All	Fully collapses the tree structure of the variable browser
Reset	Resets the values of all target and ancillary variables in a solved instance, and changes the causality of ancillary variables to undefined

Solve	Exports the model to Mathematica for execution and displays the results in the Browser window after solving is complete
Update to SysML	Causes the results in the Browser window to be exported to the SysML instance in MagicDraw

6.2.4 Relationship Browser

The Relationship Browser displays the constraint equations present in the parametric model and shows their current status during solution. By selecting one of the blocks in the Variable Browser (e.g. LittleAircraft in Figure 18), the equations specific to that part of the model are displayed in the Relationship Browser.

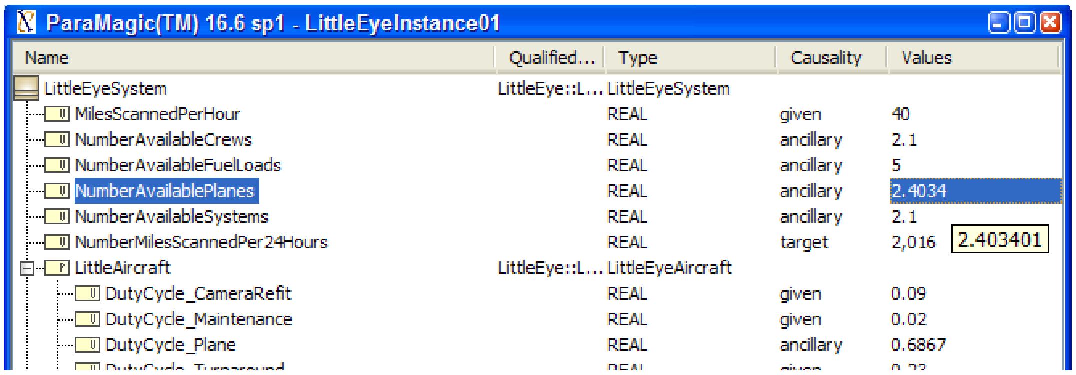
The checkbox in the Active column allows individual equations to be "turned off" during solving, i.e. not exported to Mathematica. This may prevent other parts of the parametric model from being solved.

The columns Local and Oneway refer to local vs. inherited characteristics and static causality characteristics of the equations. These are not user-controlled via this browser—they are determined by the structure of the model, which the user can change in MagicDraw and then re-open in the ParaMagic^{TM} browser to see the updated Local or Oneway properties.

6.2.5 Editing an Instance in the Browser Window

Attribute values and attribute causalities can be modified within the Browser window, as well as in the SysML instance diagram before launching the browser. Note that editing the SysML instance after the Browser is launched will not update the Browser. Changing either a value or causality in the Browser after a solution has been calculated will return the model to an unsolved state, which can then be solved with the changed parameters.

In ParaMagic™ 16.6 sp1, users can select the max number of decimal places to display in the browser. This can be done by specifying the value of maxNumberOfDecimalsToDisplay variable in the ParaMagic.ini file. See section 6.3 below for details. When a user hovers the mouse pointer over the displayed value, the browser will indicate the true value of the variable. If a user double-clicks on value with causality= "given", the true value will be displayed in the edit box.



Name	Qualified...	Type	Causality	Values
LittleEyeSystem	LittleEye::L...	LittleEyeSystem		
MilesScannedPerHour		REAL	given	40
NumberAvailableCrews		REAL	ancillary	2.1
NumberAvailableFuelLoads		REAL	ancillary	5
NumberAvailablePlanes		REAL	ancillary	2.4034
NumberAvailableSystems		REAL	ancillary	2.1
NumberMilesScannedPer24Hours		REAL	target	2,016 2.403401
LittleAircraft	LittleEye::L...	LittleEyeAircraft		
DutyCycle_CameraRefit		REAL	given	0.09
DutyCycle_Maintenance		REAL	given	0.02
DutyCycle_Plane		REAL	ancillary	0.6867

Figure 20: Hover the mouse pointer over the displayed value to see the true value.

6.3 ParaMagic.ini file

The ParaMagic.ini file provides settings to control ParaMagic™ behavior. The role of each variable and the format for specifying its values are described below. If there are restrictions on the values allowed for a variable, these are specified in the following format: Variable name = allowable value 1 / value 2 / …

1) com.intercax.xaitools.solver.name = Mathematica / OpenModelica

   This variable is used to specify if ParaMagic™ should use Mathematica or OpenModelica as the core solver. Allowable values for this variable are Mathematica or OpenModelica.

2) com.intercax.xaitools.local.mathematica.true.or.false=true / false

   This variable is used to specify if ParaMagic™ should use local or remote Mathematica. Set value = true for local Mathematica, or false for remote Mathematica (via XWS). If the value is set to false, ensure that you specify the location of remote Mathematica (com.intercax.xaitools.soap.mathematica.serverhost) and the access key (com.intercax.xaitools.soap.mathematica.accesskey). Contact your XWS administrator for access to remote Mathematica. For information on remote Mathematica (via XWS), refer to Steps 3d and 3e in section 3.2 above.

3) com.intercax.xaitools.solver.timeout.in.seconds=180

   This variable is used to specify the time interval in seconds—measured after pressing the Solve button in the ParaMagic™ browser—after which ParaMagic™ will disconnect from the external solver (e.g. Mathematica or MATLAB). Note that this will not kill the solver runtime process. However, ParaMagic™ will not wait for the solution results. By default, the value of this variable is set to 180 seconds (3 mins). If you estimate your problem will take longer to solve, increase the value of this variable accordingly. If you do not want ParaMagic™ to timeout, specify a value less than 0. For example, setting the value as -1 will ensure that the solution process is not timed out.

4) com.intercax.xaitools.soap.mathematica.mac.path=/Applications/Mathematica.app/Contents/MacOS/MathKernel

   This variable is used to specify the location of Mathematica installation (MathKernel) on Mac OS X. This value is needed when using local Mathematica on a Mac for ParaMagic™. By default, Mathematica installation places the MathKernel file at the location specified above.

5) com.intercax.xaitools.om.mac.path = /Users/<your home account>/OpenModelica150

   This variable is used to specify the location of OpenModelica folder on Mac OS X. This value is needed when using local OpenModelica on a Mac for ParaMagic™. By default, OpenModelica installs in the home folder.

6) com.intercax.xaitools.om.mac.exec.path=/Users/<your home account>/temp

   This variable is used to specify a working folder for OpenModelica to execute parametric models. It is necessary that there should be no spaces in the path to this folder, and the folder must exist before you use ParaMagic™ with OpenModelica. For example,
   com.intercax.xaitools.om.mac.exec.path=/Users/manas/temp

7) com.intercax.xaitools.soap.mathematica.serverhost=xws.magicdraw.com:8080

   This variable is used to specify the location of remote Mathematica installation. Values are specified in the following format: IP number (or alias): port number. For information on remote Mathematica (via XWS), refer to Option 3 in Step 4.

8) com.intercax.xaitools.soap.mathematica.accesskey=tempAccessKey

   This variable is used to specify the access key for the remote Mathematica installation. For information on remote Mathematica (via XWS), refer to Option 3 in Step 4.

9) com.intercax.xaitools.local.matlab.mfile.location=
   C:\\Program Files\\MagicDraw UML 16.6\\samples\\ParaMagic\\Tutorials\\HomeHeating

   This variable is used to specify the default location of MATLAB M-files (functions or scripts). Though the location of MATLAB M-files can be specified for each constraint block wrapping the file, it is sometimes useful to specify a default location. ParaMagic checks if the location of the M-file is specified in the constraint block used in the MagicDraw model. If not specified in the constraint block, ParaMagic will check for the M-file in the default location specified here.

10) com.intercax.xaitools.temp.dir= temp

    This variable is used to specify the location of temp folder where the Mathematica job is created and interim results are stored during the solution process. The location is specified relative to the xfw folder — <MD_Installation>\plugins\com.intercax.paramagic\xfw.

11) RetryInvervalInMilliSeconds=1000

    When using remote Mathematica (via XWS) network issues or other problems may hinder ParaMagic from connecting to Mathematica. This variable is used to specify the time interval after which ParaMagic^{TM} will retry connecting to Mathematica.

12) NumberOfRetry=1
    When using remote Mathematica (via XWS) network issues or other problems may hinder ParaMagic from connecting to Mathematica. This variable is used to specify the number of times ParaMagic^{TM} should retry connecting to Mathematica.

13) xws.urn.version = urn:XWS_v2.2

    This variable is used to specify the XWS service used for connecting to remote Mathematica. For information on remote Mathematica (via XWS), refer to Steps 3d and 3e in section 3.2 above.

14) maxNumberOfDecimalsToDisplay=-1

    This variable is used to specify the max number of decimal places to display for values in the ParaMagic™ browser. If the default value (-1) is specified, location-specific default settings will be used, for e.g. displaying 3 decimal places in US.

7 CONNECTIONS TO EXTERNAL TOOLS

7.1 ParaMagic - Custom Mathematica Connection

The ParaMagic - Custom Mathematica connection (a.k.a cMathematica) exposes full power of Mathematica to ParaMagic users through the cMathematica function. This allows "wrapping" of both pre-defined and user-written Mathematica functions—saved as .m files in the Mathematica Autoload folder—as constraint blocks. Nine pre-defined graphing functions and three statistical functions are included with ParaMagic™. The user can create as many custom functions as desired if he or she has familiarity with Mathematica function programming and access to the Mathematica directory structure.

The canonical form of the constraint expression is:

output parameter = cMathematica(function name, input arguments)

and would be used in a parametric diagram as shown in Figure 21.

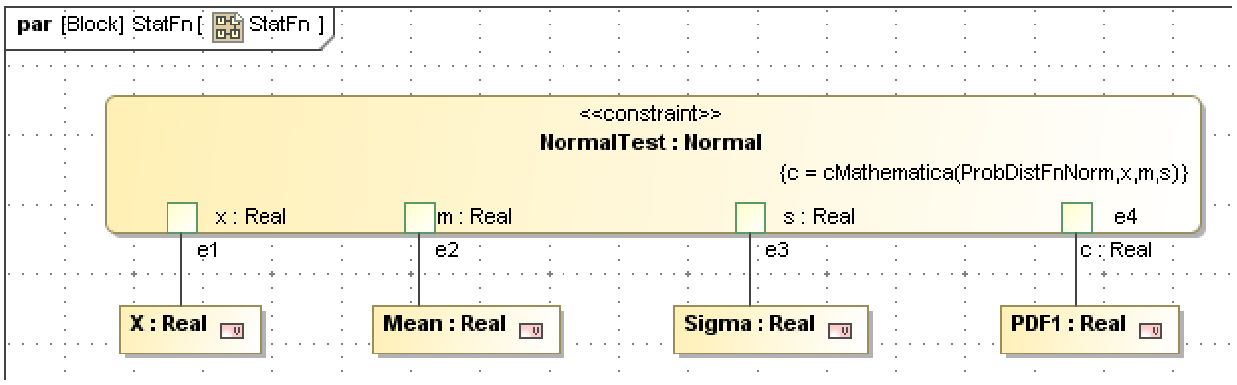


Figure 21: Constraint Expression using cMathematica Function in Parametric Diagram

The Orbital tutorial provided with ParaMagic™ demonstrates an example usage of a graphing function provided by cMathematica feature. It is located under <MD_Root>\samples\ParaMagic\Tutorials\Orbital.

This feature is not supported if OpenModelica is selected as the core solver.

7.1.1 Installation

A library of pre-defined Mathematica graphing and statistical functions is provided with ParaMagic 16.6. After ParaMagic™ 16.6 sp1 installation, this library is located here: <MD_Root>\plugins\com.intercax.paramagic\xfw\conf\ICAX.zip. If you are using local Mathematica, right click on the ICAX.zip file and extract it to <Your Mathematica Installation>\SystemFiles\Autoload folder. For example, if you have local Mathematica version 7.0 on your Windows machine, extract ICAX.zip to C:\Program Files\Wolfram Research\Mathematica\7.0\SystemFiles\Autoload. If you are using XWS-based Mathematica, ask your system administrator to extract ICAX.zip to your Mathematica server installation. If you are evaluating ParaMagic™ 16.6 sp1 and using our test server xws.magicdraw.com, then you do not need to follow the steps above. The library is already extracted in the Mathematica installation at xws.magicdraw.com and ready for you to evaluate.

Note: After extracting ICAX.zip file to C:\Program Files\Wolfram Research\Mathematica\7.0\SystemFiles\Autoload folder, verify that the Autoload folder has an ICAX folder and Mathematica .m files inside the ICAX folder. There should not be an ICAX folder inside the ICAX folder.

7.1.2 Usage

Tutorial 6 (Orbital) in the ParaMagic tutorials describes the process of using the pre-defined graphical and statistical Mathematica functions (sections 7.1.3 and 7.1.4 below). See section 6.2, Step IV, in ParaMagic Tutorials document for details. Note that the ParaMagic Constraint Block Library package—loaded with the ParaMagic Profile—has a generic cMathematica_Function constraint block that can be copied (in your package) and used in the manner described in Tutorial 6.

In contrast to ParaMagic^{TM} 16.0, ParaMagic^{TM} 16.6 sp1 allows users to specify the location of the plots created by the Mathematica graphing functions. The location can be specified for each constraint block that wraps a Mathematica function. The tag working_dir (owned by the stereotype External_Model) is populated for the constraint specification in the constraint block to specify the location—also described in Tutorial 6. The behavior of ParaMagic with respect to the working_dir tag is as below:

1) If the tag working_dir is populated and
   a) the folder location exists, ParaMagic^{TM} will pass the location as the first argument to the Mathematica function.
   b) the folder location does not exist, ParaMagic^{TM} will show an error message to the user.

2) If the tag working_dir is not populated, ParaMagic^{TM} will not pass the location to the Mathematica function. Therefore, the invoked Mathematica function should not expect its first argument to be the folder location.

Note that the working_dir tag is used to pass the location of a folder to a Mathematica function. This folder can be used for exporting plots or other forms of outputs (e.g. .txt or .csv files with the results).

Both input and output parameters of the constraint block wrapping the cMathematica function can be single-valued or multi-valued (e.g. an array of real number) but not a complex data structure (e.g. multi-dimensional arrays).

7.1.3 Graphing Functions

Nine pre-defined graphing functions have been created to provide quick access to some of the two-dimensional plotting functions in Mathematica. In the interests of simplicity, the flexibility of these pre-defined functions is limited. Users familiar with Mathematica function programming may want to create their own routines.

These nine functions have several common features:

- When executed, each function creates and saves a plot file with a fixed name to a location specified by the user. Calling the function a second time will over-write the first file. The location of the plot file is specified in the constraint block used in the model.
- Each function returns a single parameter (single value real) to ParaMagic^{TM} on completion of the function, with a value of 1.
- The input arguments can include numeric data, expressed as real single value or aggregate data, and strings which appear as labels on the plot. The strings cannot be entered as variables (constraint parameters); they must be explicitly fixed in the constraint specification.
- Blank spaces are not allowed in the names of the output and the input parameters of the cMathematica function.
- In the following table, Title will be displayed as the title of the graph, XAxis will be displayed as the horizontal axis label and YAxis will be displayed as vertical axis label. Strings must be enclosed in

quotes, e.g. "Power_versus_Time". Since plot and axes labels are also input arguments to the cMathematica function, blank spaces are not allowed. For example, if the plot title is "Power versus Time", ParaMagic^{TM} will throw an instance parsing error.

- It is frequently convenient to create a hyperlink in the MagicDraw model pointing to the output file created. For model portability, it is recommended to define the hyperlinked location relative to the MagicDraw installation. You may use <install.root> to refer to your MagicDraw installation. For example, <install.root>\plugins\com.intercax.paramagic\xfw\conf\PlotXY.jpg will automatically resolve to the plot file location.

Function	**Output**	**Description**
ICAXPlotX	PlotX.jpg	Plots a line graph of the x values vs. {1,2,3,…} z =cMathematica(ICAXPlotX,x,"Title", "XAxis", "YAxis")
ICAXPlotXT	LinePlotXT.jpg	Plots a line graph of x vs. t values Z =cMathematica(ICAXPlotXT,t,x,"Title", "XAxis", "YAxis")
ICAXPlotXY	LinePlotXY.jpg	Plots a two line graph of x and y values vs. {1,2,3,…} Z =cMathematica(ICAXPlotXY,x,y,"Title", "XAxis","YAxis")
ICAXPlotXYT	LinePlotXYT.jpg	Plots a two line graph of x and y values vs. t values z =cMathematica(ICAXPlotXYT,t,x,y,"Title", "XAxis", "YAxis")
ICAXPlotXYScatter	ScatterPlotXY.jpg	Plots a scatter plot of y vs. x values z =cMathematica(ICAXPlotXYScatter,x,y,"Title", "XAxis", "YAxis")
ICAXBarChartX	BarChartX.jpg	Plots a bar chart of the x values vs. {1,2,3,…} z =cMathematica(ICAXBarChartX,x,"Title", "XAxis", "YAxis")
ICAXBarChartXY	BarChartXY.jpg	Plots a double bar chart of x and y values vs. {1,2,3,…} z =cMathematica(ICAXBarChartXY,x,y,"Title", "XAxis", "YAxis")
ICAXPieChartX	PieChartX.jpg	Plots a pie chart of the x values vs. {1,2,3,…} z =cMathematica(ICAXPieChartX,x,"Title")
ICAXHistogramX	HistogramChartX.jpg	Plots a histogram of the x values vs. {1,2,3,…} z =cMathematica(ICAXHistogramX,x,"Title","XAxis", "YAxis")

7.1.4 Statistical Functions

Three pre-defined statistical functions have been created to provide quick access to some of the statistical power of Mathematica.

Function	**Description**
ProbDistFnBinom	Returns the probability distribution function for outcome k in a binomial distribution of n trials and success probability p (k and n integers) c =cMathematica(ProbDistFnBinom,n,p,k)
ProbDistFnNorm	Returns the probability distribution function for value x in a normal distribution with mean m and standard deviation s c =cMathematica(ProbDistFnNorm,x,m,s)
ProbDistFnPois	Returns the probability distribution function for value k in a Poisson distribution with mean m (k integer) c =cMathematica(ProbDistFnPois,m,k)

7.1.5 User-Defined Mathematical Functions

A user familiar with Mathematica function programming can easily create and use custom Mathematica functions within SysML parametric diagrams with the cMathematica capability. There are two ways to creating custom functions, using one of the five preset UserfnN.m functions (where N runs from 1 to 5) or creating a new function using one of the existing pre-defined functions as a template for compatibility with ParaMagic.

7.1.6 UserfnN.m

Within the ICAX library with the pre-defined graphing and statistical functions, we have provided five "empty" functions. Each one can be edited using Mathematica or any standard text editor. Replace the comment lines

```
(* :Add Mathematica code calculating output b from inputs t and m *)
(* :To save graph, Export["GraphFileName", GraphFunction[arguments]] *)
```

with valid Mathematica code, which will be executed when the function is called.

Note that only two input arguments are defined in the function definition. This number can be reduced or increased with appropriate modification of the template. Output and input parameters can be single-valued or multi-valued (e.g. array).

7.1.7 Custom Functions

The user can write his/her own Mathematica functions using the cMathematica capability. We recommend using one of the pre-defined functions as a template to insure compatibility with ParaMagic.

In order for a function to be recognized by Mathematica, it must be autoloaded on start-up. See the Mathematica user documentation for discussion on declaring and loading functions. One easy way to accomplish this is to

- Save the new .m file in <Mathematica installation directory>\SystemFiles\Autoload\ICAX
- Edit the Master.m and Kernel\init.m files in the ICAX folder to declare the new function. Add the lines

  ```
  DeclarePackage["ICAX`NewFunctionName`", {"NewFunctionName"}]
  ```

  to each of these files and save, where NewFunctionName is the name of the new function, without the .m extension.

7.2 ParaMagic - Excel Connection

The ParaMagic - Excel Connection (PM-EC) allows users to read/write slot values from/to Microsoft Excel files. This enables users to populate instances values from Excel workbooks[18] before solving the model in ParaMagic, and export solved instance values to Excel workbooks. Hence, Excel spreadsheet-based templates can be used with ParaMagic™ in a sequential manner. For example, users may compute some parameters using spreadsheets, read these values in ParaMagic™ to populate some SysML instance slot values, solve the SysML model using ParaMagic™, and export solved values to spreadsheets.

The Orbital tutorial provided with ParaMagic™ is an example usage of the PM-EC feature. It is located under <MD_Root>\samples\ParaMagic\Tutorials\Orbital.

[18] The terms Excel file and Excel workbook mean the same.

7.2.1 Operation

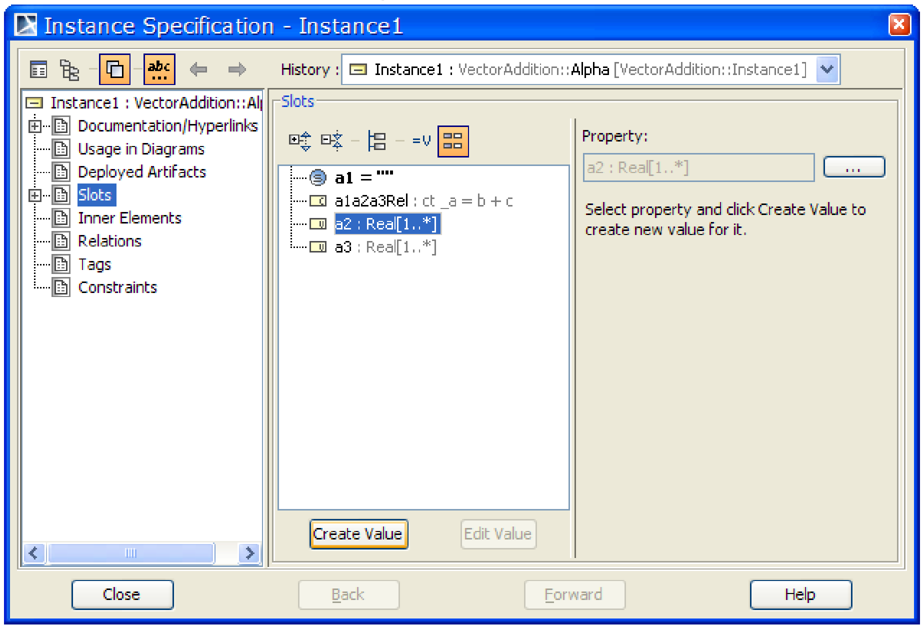
Follow the steps below to use PM-EC to connect your SysML models to Excel spreadsheets.

Step 1. *Initialize slot*

If you have been using previous versions of ParaMagic™ and have already initialized slots for your instance model—that you now intend to connect to Excel spreadsheets—you may skip to step 2.

Each slot that needs to interact (read/write) with Excel spreadsheets must be initialized. Initialization ensures that the slots are visible in the MagicDraw containment tree from where PM-EC operations can be invoked. For initializing a slot:

a) Double click an instance in the MD containment tree. A specification window opens as shown below in Figure 22.
b) Select a slot that needs to interact with Excel spreadsheets and click on Create Value button. For example, slot a2 is selected in Figure 22.
c) After initialization, a slot will appear with one empty value, as shown for slot a1 in Figure 22. Initialized slots will also be visible in the containment tree under the parent instance, as shown for slots a1, a2, and a3 of Instance1 in Figure 23.



Select property and click Create Value to create new value for it.

Figure 22: Initialize slots

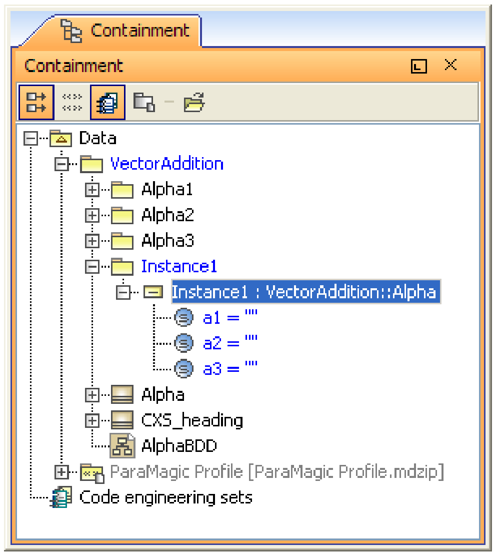
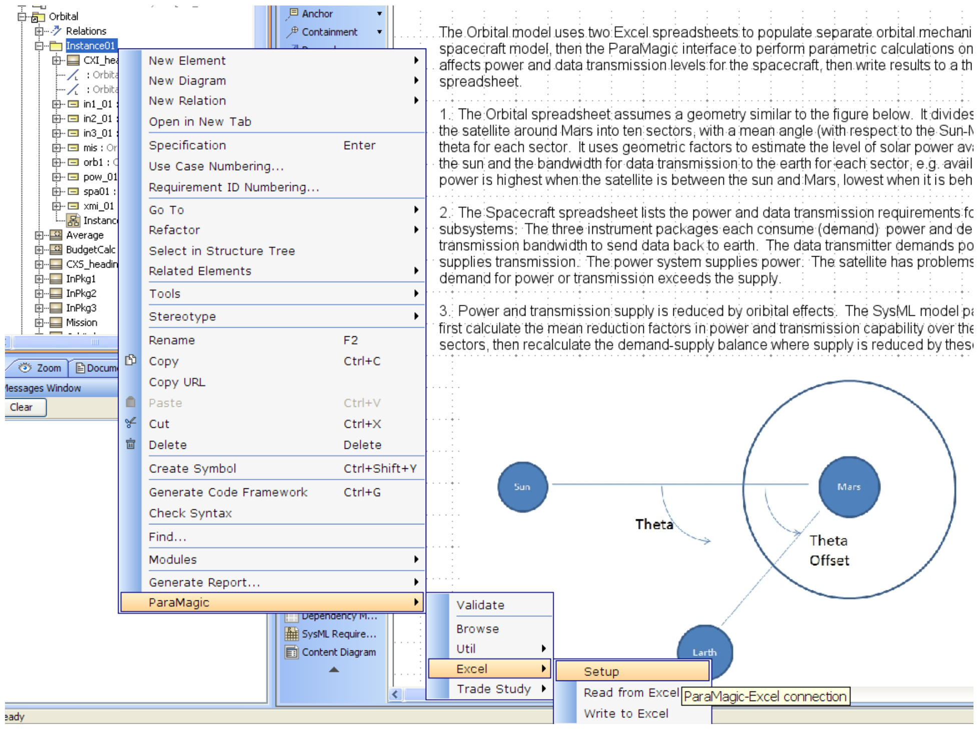


Figure 23: MagicDraw Containment Tree after initialization of slots.

For multi-valued slots (e.g. slots that store an array of numbers), the initialization process is the same as above. For initializing multi-valued slots that are not inputs (givens) for ParaMagic^{TM} simulations, see step 9 below.

Step 2. *Setup Excel slots to interact with Excel worksheet*

To setup slots to interact with Excel, right click on the slot (or its parent instance) in MD containment tree and select ParaMagic→Excel→Setup, as shown in Figure 24 below. Once the Setup menu is selected, ParaMagic Excel Setup utility appears, as shown in Figure 25 below.



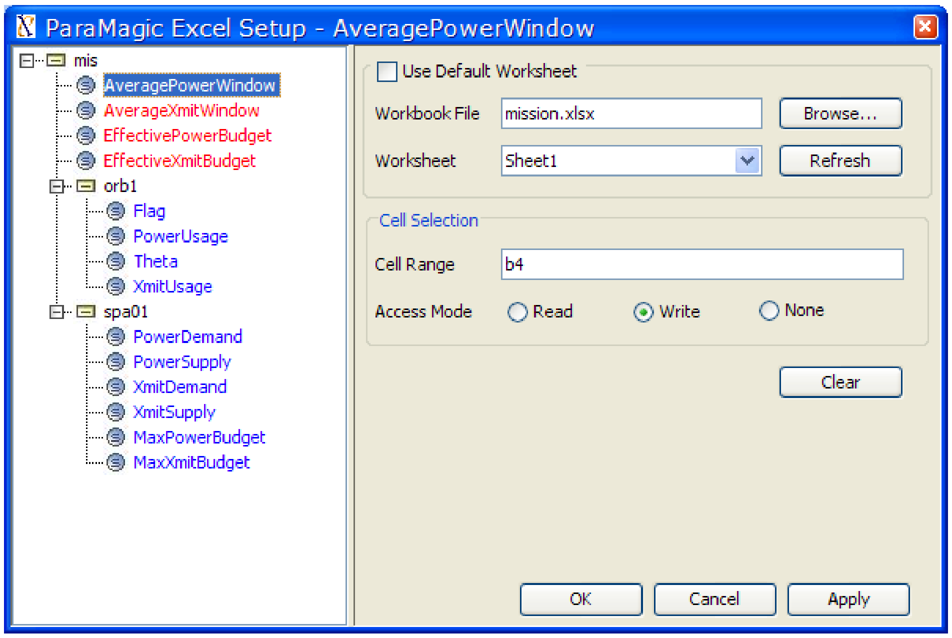
The Orbital model uses two Excel spreadsheets to populate separate orbital mechani
spacecraft model, then the ParaMagic interface to perform parametric calculations on
affects power and data transmission levels for the spacecraft, then write results to a th
spreadsheet.

1. The Orbital spreadsheet assumes a geometry similar to the figure below. It divides
the satellite around Mars into ten sectors, with a mean angle (with respect to the Sun-M
theta for each sector. It uses geometric factors to estimate the level of solar power av
the sun and the bandwidth for data transmission to the earth for each sector, e.g. avail
power is highest when the satellite is between the sun and Mars, lowest when it is beh

2. The Spacecraft spreadsheet lists the power and data transmission requirements fo
subsystems: The three instrument packages each consume (demand) power and de
transmission bandwidth to send data back to earth. The data transmitter demands po
supplies transmission. The power system supplies power. The satellite has problems
demand for power or transmission exceeds the supply.

3. Power and transmission supply is reduced by oribital effects. The SysML model pa
first calculate the mean reduction factors in power and transmission capability over the
sectors, then recalculate the demand-supply balance where supply is reduced by thes

Figure 24: Setup Excel connection for a slot



Use Default Worksheet [ ]
Workbook File: mission.xlsx
Worksheet: Sheet1
Cell Selection
Cell Range: b4
Access Mode: Read [✓ Write] None

Figure 25: ParaMagic Excel Setup utility

The ParaMagic Excel Setup utility has been updated in ParaMagic^{TM} 16.6. Until ParaMagic 16.5, users had to invoke the setup utility individually for each slot. However in ParaMagic^{TM} 16.6, the setup utility can be invoked for a slot, instance, or instance package. When invoked, users can see the entire instance model (with the root instance at the top). Then, users can click on specific slots and link them to Excel spreadsheets.

If Excel connection-related information is already populated for the slot, corresponding values will be shown in the setup window as shown in Figure 25 above.

Step 3. *Specify Excel connection for slots*

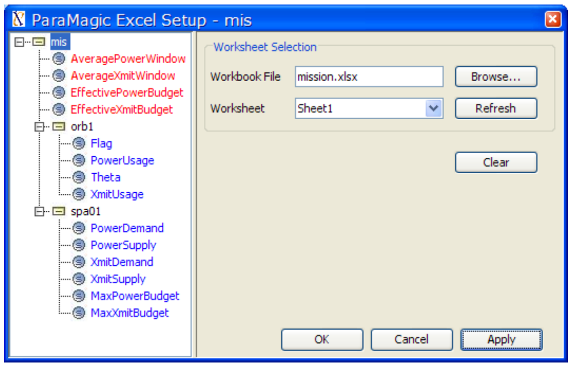
Specify Excel connection information for all slots that need to be linked to Excel spreadsheets. To do this, repeat steps a-f below for all such slots.

a) Select the slot in the instance model shown in the setup utility (left pane). For example, the slot AveragePowerWindow is shown selected in Figure 25.

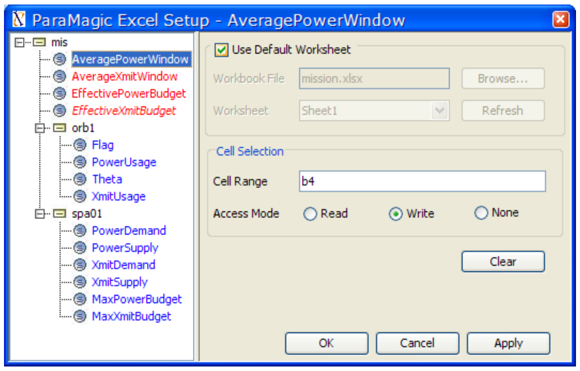
b) Specify the location of the Excel workbook associated with the slot in the *Workbook File* field. If the workbook file is located in the same folder as the MagicDraw file (mdzip), only the name of the workbook file needs to be specified (with the extension .xls or .xlsx). Alternatively, click the Browse button to select the workbook file. Note that storing Excel workbooks with the MagicDraw file enhances the portability of the model. The MagicDraw model file and the workbook files can be distributed together without system-specific folder location settings.

c) Select the worksheet associated with the slot. If you selected the workbook file using the Browse button, the list of spreadsheets in that file is automatically available for selection. However, if you manually entered the workbook file, click on the Refresh button to populate the list of spreadsheets in the workbook file. Note that if the workbook file is not found, the list of available worksheets will be empty.

If multiple slots owned by a single instance are linked to the same workbook and worksheet, it is preferable to specify workbook and worksheet information at the instance level and then select the Use Default Worksheet option for all such slots. Figure 26 below shows an example where the Excel workbook and worksheet is specified for the instance mis. The slot AveragePowerWindow (owned by this instance) uses the worksheet.


ParaMagic Excel Setup - mis

Worksheet Selection
Workbook File: mission.xlsx
Worksheet: Sheet1


ParaMagic Excel Setup - AveragePowerWindow

[✓] Use Default Worksheet
Workbook File: mission.xlsx
Worksheet: Sheet1

Cell Selection
Cell Range: b4
Access Mode: Read [✓ Write] None

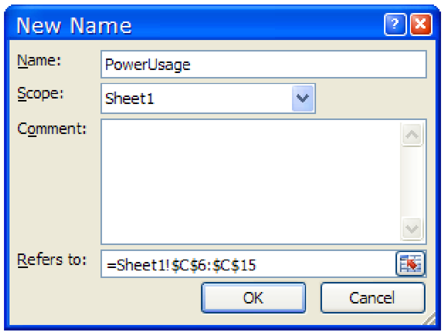
Excel worksheet specified for the instance Slot uses the default worksheet

Figure 26: Specifying workbook and worksheet at the instance level and using it for slots

d) Specify the cell range (in the selected worksheet) associated with the slot. Cell range can be specified using cell name or address. Until ParaMagic™ 16.5, only cell address was acceptable as cell range.

If the cell range is specified using a name, the following rule applies:

i) The scope of the name should be the worksheet and not the workbook. For example, as shown in Figure 25 below, cell range C6:C15 in Sheet1 is named as PowerUsage. The scope of this name is the worksheet Sheet1.


New Name

Name: PowerUsage
Scope: Sheet1
Comment:
Refers to: =Sheet1!$C$6:$C$15

Figure 27: Specify cell range using name

If the cell range is specified using cell address, the following rules apply:

i) Cell range must be specified using the following Excel syntax:
  (1) for multi-valued slots: <first cell address>:<last cell address>. For example, if a multi-valued slot is to be associated with cells from A2 to A10, cell range should be specified as A2:A10.
  (2) for single-valued slot: <cell address>. For example, if a single-valued slot is associated with cell A2, cell range should be specified as A2.

ii) Cell ranges must be specified without the worksheet reference. For example, cell range A2 to A10 in MySheet worksheet should be specified as A2:A10 and not as MySheet1!A2:A10.

Copyright © InterCAX LLC. All Rights Reserved.

iii) A cell range should not have special characters (including whitespaces) that Excel cannot recognize. The following are examples of syntactically incorrect cell ranges: A2 : A10, A 2 : A 10, A$%2:A*9

iv) For associating a SysML slot to an Excel spreadsheet using PM-EC, cell ranges must be contiguous and correspond to a single column/row. For example, A2:A10 and A2:F2 are contiguous and correspond to single column and row cell ranges respectively; A2:B10 is a contiguous but not a single column/row cell range; and A2:A10,C2:C10 is a valid but non-contiguous cell range.

v) PM-EC ignores the order in which the first and last cells are specified in the cell range field. For example, A10:A2 is treated the same as A2:A10.

e) Specify the access mode (Read or Write).

If the Access Mode=Read, ensure that the workbook file is saved before the Excel Read operation is executed. If Access Mode=Write, ensure that the workbook file is closed before the Excel Write operation is executed. As shown in Figure 25, slots that are setup to read from spreadsheets are shown in blue and slots that are setup to write to spreadsheets are shown in red.

f) Press the Apply button to save slot setup information to the MagicDraw model. Those slots for which the Excel connection settings have been changed but not saved to the MagicDraw model (Apply button) are shown in *italics*—see the slot mis.EffectiveXmitBudge in Figure 26.

The table below summarizes the commands that can be issued from the PM-EC setup window.

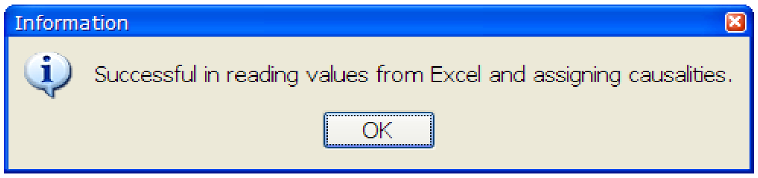
Command	**Description**
Browse	Opens a file browser for selecting an Excel workbook file (.xls or .xlsx)
Refresh	Creates a list of spreadsheets available in the specified workbook file
Apply	Saves the Excel connection settings to the MagicDraw model
OK	Saves the Excel connection settings to the MagicDraw model (Apply) and closes the setup utility
Clear	Removes the Excel connection settings for the subject slot
Cancel	Closes the PM-EC setup utility without saving the Excel connection settings

Note: Since the workbook and worksheet information is specified for a particular slot, different slots in an instance may be connected to different workbooks/worksheets and setup to Read/Write information from/to Excel spreadsheets.

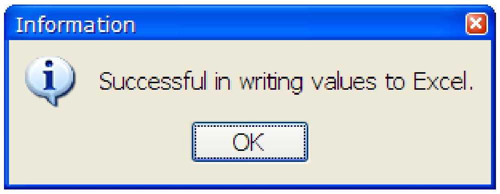
Step 4. *Execute Excel Read / Write operation for the setup slot*

Once a slot is setup, Excel Read or Write operations can be executed by right clicking the slot and selecting ParaMagic→Excel→Read from Excel or Write to Excel menus. When the Read or Write operation is executed successfully, corresponding information messages pop-up as shown in Figure 28.

During the Excel Read operation, the causality of all slot values read from Excel is automatically changed to "given" (from "undefined") in accordance with Table 1. Once a slot has been connected to an Excel worksheet, the Excel Read/Write operations can be invoked whenever new values are to be read from (or written to) the Excel worksheet.


Successful in reading values from Excel and assigning causalities.

PM-EC message for successfully reading slot values from Excel


Successful in writing values to Excel.

PM-EC message for successfully writing slot values to Excel

Figure 28: Information messages for successful execution of PM-EC Read/Write operations

Steps 1-4 above demonstrate the operation of PM-EC in ParaMagic^{TM} 16.6 sp1 for slots. Additional steps for more efficient operation of PM-EC are described below.

Step 5. *Execute Excel Read/Write operation for an instance or an instance package*

Excel Read/Write operations can be invoked for a slot (as demonstrated above), an instance, or a package containing instances. To invoke Excel Read/Write operation on an instance (or instance package), right click the instance (or instance package) and select ParaMagic→Excel→Read from Excel or Write to Excel menu.

If the Excel Read operation is invoked on an instance, PM-EC will execute the Excel Read operation for all slots of that instance that are setup to read from Excel (Access Mode=Read). Similarly, if the Excel Write operation is invoked on an instance, PM-EC will execute the Excel Write operation for all slots of that instance that are setup to write to Excel (Access Mode= Write).

If the Excel Read/Write operation is invoked on an instance package, PM-EC will perform the Excel Read/Write operation for all instances in the instance package.

Step 6. *Initializing slots with empty values before solving with ParaMagic.*

If a slot value is intended to be a target or an undefined variable for ParaMagic solution, users are still required to initialize the slot (as shown in step 1 above). If the slot value is an array, then n values of that slot must be initialized (where n is the array size). This can be a cumbersome process. Using PM-EC, users can initialize a slot with n empty values by setting them up to read values from n empty cells in an Excel spreadsheet and then executing the Excel Read operation. If the slot values do not have any pre-assigned causality, the Excel Read operation will set their causality to "undefined". For those empty values that are the targets, users can change the causality from “undefined” to “target” in the ParaMagic^{TM} browser.

The table below summarizes the PM-EC commands, the arguments on which these commands may be issued, their behavior, and the response message on successful execution of commands. These commands are available under ParaMagic→Excel menu.

Command	**Arguments**	**Description**	**Messages**
Setup	Slot	Opens the Excel Setup utility for the instance owning the slot. The subject slot is selected.	After the Apply (or OK) button is pressed, the setup values are saved to the model. No messages pop-up after the save operation is completed.
	Instance	Opens the Excel Setup utility for the instance. The subject instance is highlighted in the instance model.	
	Instance Package	Opens the Excel Setup utility for the root instance in this package.	
Read	Slot	Reads values from an Excel spreadsheet and populates slot values if slot access mode=Read. The causality is set to "given" for all slot values read from Excel	• If slot values are read successfully from Excel and the causality assignment is successful, the response message states "Successful in reading values from Excel and assigning default causalities." • If slots values are successfully read from Excel but the causality assignment is unsuccessful, the response message states "Successful in reading values from Excel but unsuccessful in assigning causalities." • If slot values are not read from Excel, response messages (indicating the problems) are shown in the MagicDraw Message window.
	Instance	Executes the Excel Read operation for all slots in the instance.	A successful response message is show only when the Excel read and causality assignment operation is successful for all slots (with read access mode) in the instance. Else, no response message is shown.
	Package	Executes the Excel Read operation for all slots of all instances in the package.	A successful response message is shown only when the Excel read and causality assignment operation is successful for all slots of all instances in the instance package. Else, no response message is shown.
Write	Slot	Writes slot values to an Excel spreadsheet if slot access mode = Write.	• If slots values are successfully written to Excel, the response message states "Successful in writing values to Excel". • If slot values are not written to Excel, response messages

			indicate the problem.
	Instance	Executes the Excel Write operation for all slots in the instance.	A successful response message is shown if the Excel Write operation is successful for all slots of the instance.
	Package	Executes the Excel Write operation for all slots of all instances in the package.	A successful response message is shown only if the Excel Write operation is successful for all slots of all instances in the instance package.

7.2.2 Features and Specific Behavior

1) Both versions of MS Excel files are supported—Excel 97-2003 files (.xls extension) and Excel 2007 files (.xlsx extension).

2) Excel Read/Write operations can be invoked for a slot, an instance, or a package containing instances.

3) Both numeric and text (string) values can be read from Excel or written to Excel. If the block value property corresponding to a slot is typed by:
   a) value type Real or its subtype, PM-EC will read/write only numeric values for that slot.
   b) value type String or its subtype, PM-EC will read/write values for that slot as strings.
   c) any other value type, PM-EC will behave the same as (b) above.

4) PM-EC assumes that each slot value is stored in the MD SysML model as Literal String and not in other formats, such as Opaque Expression.

5) PM-EC operations are one-time read/write operations and not a live connection. If values in Excel workbooks (associated with slots in MagicDraw) are updated, the Read operation must be re-invoked on all slots to read updated values after the Excel spreadsheet has been saved. Similarly, if slot values in SysML instance model (associated with values in Excel workbooks) are updated, the Write operation must be re-invoked to write updated values from SysML model to Excel spreadsheets after the workbooks have been closed.

6) Empty cells in an Excel spreadsheet are read as empty slot values in the Excel Read operation. Similarly empty slot values are written as empty cells in Excel spreadsheet in the Excel Write operation. Note that cells that look empty (no contents) may be null (esp. if they have not stored a value previously). In an Excel read operation, null cells are not read as empty slot values.

7) If a slot has "n" values and it is associated with "k" cells in a spreadsheet then the behavior of Excel Read and Write actions invoked on the slot are specified in the table below.

Excel Read operation	
n>=k	n<k
• Only the first k values of the slot (has n values) are updated with the values in k cells in the Excel spreadsheet. • The remaining values of the slot—(k+1)th value to the nth value—are deleted. • After a read operation, causality is assigned	• The n values in the slot are updated with the values in first n (out of k) cells in the Excel spreadsheet. • New slot values are created for values of (n+1)th to the kth cell in the Excel spreadsheet. • Causality is assigned based on Table 1 – if slot

based on Table 1 – if slot entry has a value, then set to given or keep as target. If slot entry is null, then assign/keep as undefined.	entry has a value, then set to given or keep as target. If slot entry is null, then assign/keep as undefined.

Excel Write operation	
n>k	n<k
Excel write operation will not work if n is not equal to k. This is to avoid overwriting existing cells in spreadsheets for ambiguous cases such as these.	

7.2.3 Limitations

1) The Excel workbook needs to be closed before the Excel Write operation is invoked on a slot or instance or instance package.
2) The Excel Write operation will not create a new workbook file if it does not exist.
3) PM-EC ignores the order in which the first and last cells are specified in the cell range field. For example, A10:A2 is treated the same as A2:A10.

Note: Although PM-EC is designed to ensure that existing features in Excel workbooks, such as charts, formulas, macros, and pivot tables, are preserved when reading and writing values from/to workbooks, all of these features are not tested with PM-EC. As an extra caution, PM-EC creates a backup of Excel spreadsheets, at the same location as the original spreadsheet, before the Write operation.

7.3 ParaMagic - MATLAB Connection

The ParaMagic – MATLAB connection (PM-MC) enables users to wrap MATLAB functions and scripts as SysML constraint blocks and use them in parametric models as regular constraint properties. ParaMagic™ solves for the constraints by invoking MATLAB functions/scripts when required. MATLAB scripts are commonly used to invoke and execute Simulink models. Since PM-MC can solve for constraints that wrap MATLAB scripts, it can be used to execute Simulink models and feed the results of the execution back into SysML models.

The HomeHeating tutorial provided with ParaMagic is an example usage of the PM-MC feature. It is located under <MD_Root>\samples\ParaMagic\Tutorials\HomeHeating.

This feature is not supported if OpenModelica is selected as the core solver.

A MATLAB script[19] is used for automatically executing a series of MATLAB commands. Users that perform computations on MATLAB command line write MATLAB scripts which can be loaded in the MATLAB environment to achieve the same effect. A script has no input and output arguments. However, a script may create and access variables in MATLAB workspaces. In contrast, a MATLAB function[19] accepts input arguments and has output arguments.

MATLAB scripts and functions are written in MATLAB files—commonly known as M-files. These files also have a .m extension like Mathematica files. Users should be careful to distinguish a .m file native to Mathematica versus a .m file native to MATLAB. A MATLAB M-file containing a script is known as a script M-file, and a MATLAB M-file containing a MATLAB function is known as a function M-file.

[19] MATLAB scripts and functions: http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_prog/f7-38085.html

If all relations in your SysML model are MATLAB relations (i.e. they wrap function or script M-files), then ParaMagic™ 16.6 does not require Mathematica to solve the model. This is contrast to ParaMagic™ 16.0 that required Mathematica for solving MATLAB-based relations.

In the following two sections, the process of wrapping MATLAB scripts M-files and function M-files using constraint blocks is demonstrated. Once wrapped, ParaMagic can invoke MATLAB scripts/functions when the constraints need to be solved. Step 1 and Step 2 below are common to using script and function M-files.

Step 1. *Check MATLAB installation on your computer*

Follow the steps below to ensure that MATLAB is installed correctly on your computer.

1) Go to the command prompt. On Windows, you may do this by selecting Run from the Start menu and typing the command cmd and pressing the OK button.
2) Type matlab at the command line. This should launch MATLAB on your computer.
3) Before wrapping MATLAB function or script M-files, ensure that they are correct, i.e. they have valid MATLAB syntax and provide valid results for valid inputs. To do this, run the script M-file on MATLAB installed on your computer, or call the function M-file from your MATLAB workspace. Once MATLAB scripts/functions have been tested to work with MATLAB, then they are ready to be used with ParaMagic™.

Note: PM-EC in ParaMagic™ 16.6 sp1 has not been tested with MATLAB on a Mac or Linux. Please contact us (support@intercax.com) if you face issues.

Step 2. *Specify default location of MATLAB function/script M-files*

It is preferred that users provide a default location (folder) of MATLAB function/script M-files. If the M-file location is not specified with the constraint block (that wraps the M-file), ParaMagic will search for the M-file in this default location. This behavior can used to your advantage if all (or most) of your M-files are at the same location (say X). If so, you do not need to specify the M-file location for each constraint block that wraps it but only specify location X in the manner described below.

To specify the parent folder location, follow the steps below:

1) Open ParaMagic.ini file located under MD_Root>\plugins\com.intercax.paramagic\xfw\conf\.
2) Specify location of the folder as the value of the following variable in ParaMagic.ini file
   com.intercax.xaitools.local.matlab.mfile.location
   For example, if the folder is C:\Data\My_MATLAB_Files, then the variable-value entry in the ParaMagic.ini file should look like:
   com.intercax.xaitools.local.matlab.mfile.location=C:\\Data\\My_MATLAB_Files

Note: The location of a Matlab M-file can be specified with each constraint block that wraps the M-file. This feature was not available in ParaMagic™ 16.0.

Step 3. *Specify a timeout for expecting results from MATLAB functions and script executions*

The ParaMagic.ini file includes a variable that specifies the timeout interval for ParaMagic™ when waiting for MATLAB functions and scripts to finish execution. By default, this is set to 180 seconds as shown below.

com.intercax.xaitools.solver.timeout.in.seconds=180

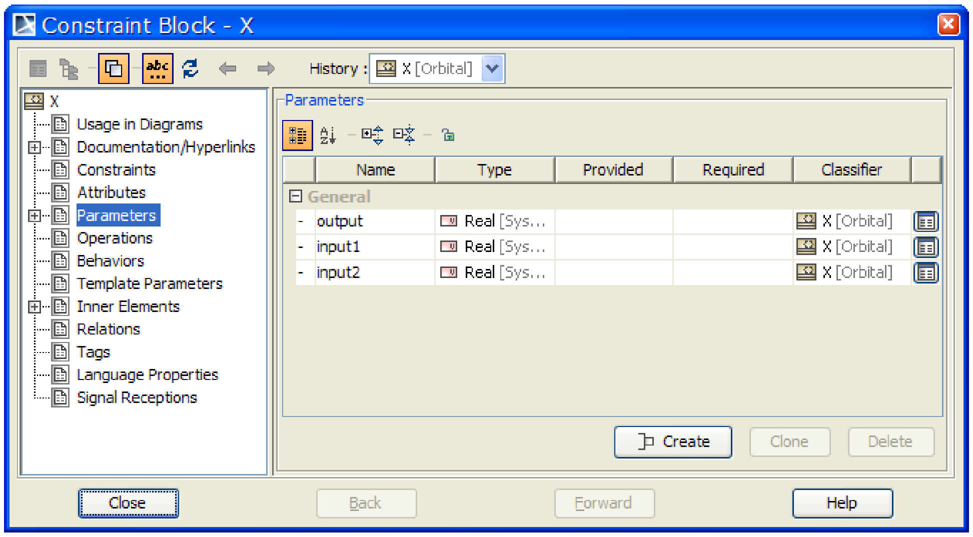
Users are recommended to specify an upper-bound value for this variable depending on the time typically taken to execute the MATLAB functions/script that they intend to use with ParaMagic^{TM}. Once the timeout is reached, ParaMagic^{TM} stops expecting results from MATLAB but does not terminate the MATLAB execution process.

Step 4. *Define a constraint block to "wrap" MATLAB script/function M-file*

1) Locate the xfwExternal_Matlab_Script or xfwExternal_Matlab_Function constraint block in the Constraint Block Library package loaded with the ParaMagic Profile. The former is setup to wrap script M-files and the latter is setup to wrap function M-files.

2) Copy the xfwExternal_Matlab_Script or xfwExternal_Matlab_Function constraint block to your package and rename it (say X).

3) Double click on the new constraint block X. This will open the Specification window as shown below.



Parameters

Name	Type	Provided	Required	Classifier
General				
output	Real [Sys...			X [Orbital]
input1	Real [Sys...			X [Orbital]
input2	Real [Sys...			X [Orbital]

Figure 29: Constraint block specification window – constraint parameters view

4) By default, this constraint block is setup to wrap a Matlab M-file with 2 inputs and 1 output. Add/remove input parameters depending on the number of inputs required by your M-file. Only 1 output parameter is allowed (single-valued or a single-dimensional array). For M-file functions, the input parameters and output parameters correspond to the arguments passed to and the value returned by the function. For M-file scripts, the input parameters correspond to those value properties that need to be passed to the script and the output parameters correspond to those value properties that are to be populated at the end of the script execution. See section 7.3.1 for specifics related to Matlab scripts.

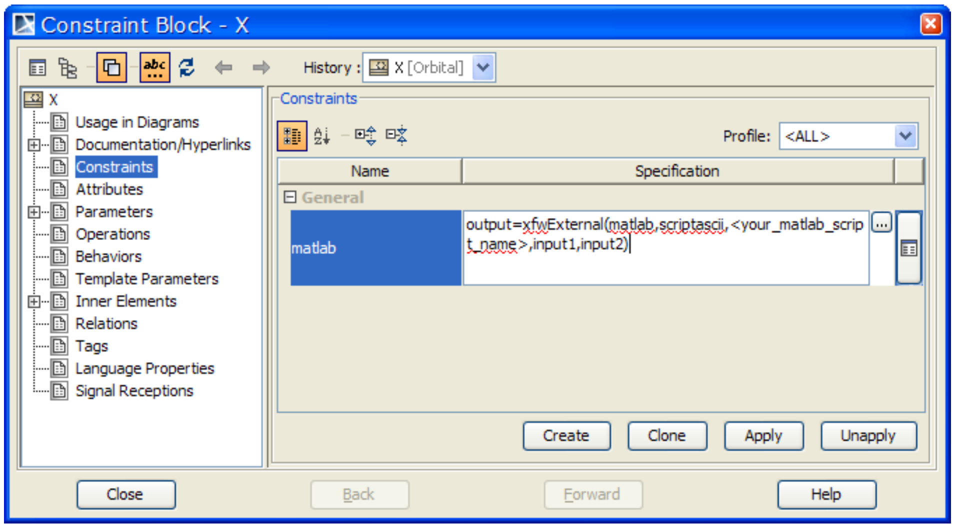
5) Click on the Constraints menu in the Constraint Block specification window. By default, a constraint has been specified. Modify this constraint equation per your requirements. The general format of the constraint relation is:

<out_param> = xfwExternal(matlab, <script_or_function>, <name_of_ M-file>, <in_param_1>,...)

where:

- <out_param> is the name of the output parameter (result computed during execution and to be read back into SysML instance)
- <script_or_function> is set to scriptascii (for MATLAB scripts) or function (for MATLAB functions)
- <name_of_M-file> is the name of the MATLAB M-file without the extension (.m)
- <in_param_1>,... is a comma-separated list of input parameters (given values to be sent from SysML instances to MATLAB script/function)

Terms enclosed in < > are variables that can have different names, while those not enclosed in < > are keywords/constants that should not be changed.



Name	Specification
matlab	output=xfwExternal(matlab,scriptascii,<your_matlab_script_name>,input1,input2)

Figure 30: Constraint block specification window – constraints view

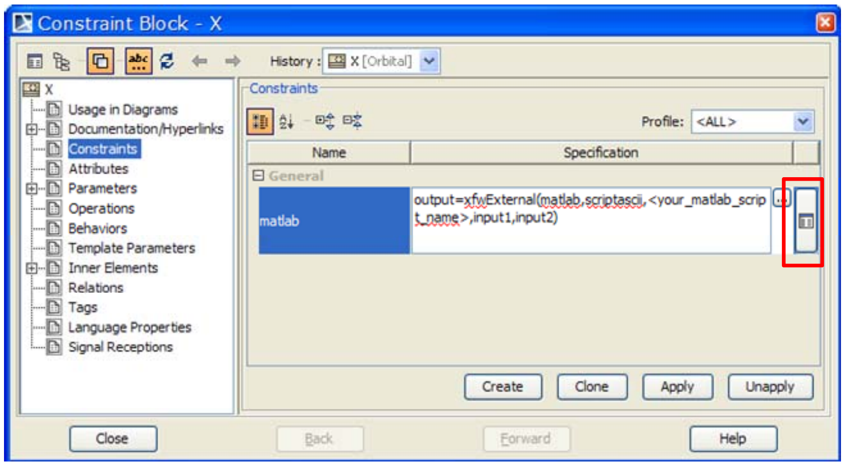
For example, Figure 35 below shows a constraint block with five parameters that wraps a MATLAB M-file script (demoscriptasciisimulink.m). These parameters correspond to the given and computed variables in the script. For example, values of row, col, outtemp, and daycyc variables are given, and the value of the variable cost is computed during the script execution.

In ParaMagic^{TM} 16.6, both input and output parameters of the constraint block wrapping a M-file could be single-valued or multi-valued (e.g. array). This is contrast to ParaMagic^{TM} 16.0 where the output parameter could only be single-valued but not an array.

Step 5. *Specify the location of the folder containing the M-file*

To specify the location of the folder containing the M-file (to be wrapped by the subject constraint block), follow the steps below:

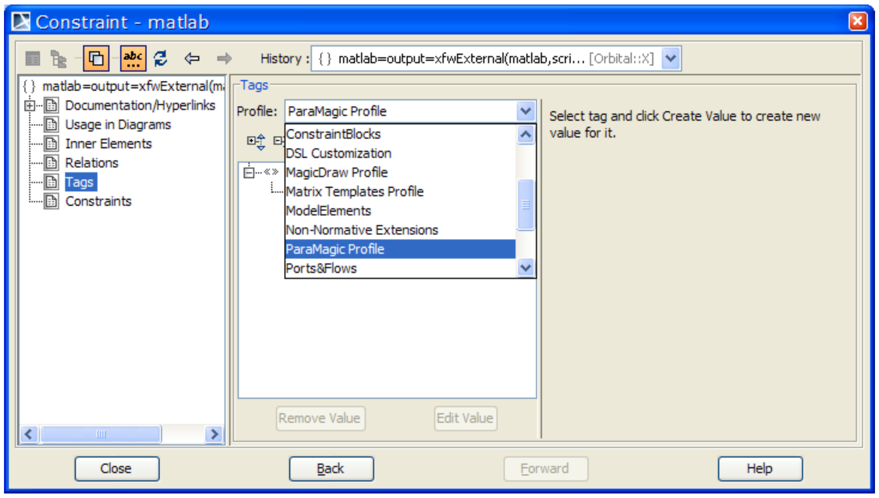
1) Click on the icon highlighted below or double click the constraint specification in the MD containment tree.



Name	Specification
matlab	output=xfwExternal(matlab,scriptascii,<your_matlab_script_name>,input1,input2)

Figure 31: Invoking the constraint specification window from the constraint view

2) The action in the previous step will open the specification window for the constraint. Click on the Tags menu as shown below and select ParaMagic Profile in the Profile list. This will display all stereotypes (and their tags) that are provided by the ParaMagic Profile for constraints.



Select tag and click Create Value to create new value for it.

Figure 32: Specifying the location of M-file - selecting the ParaMagic profile

3) Select the working_dir tag and click on the Create Value button, as shown below.

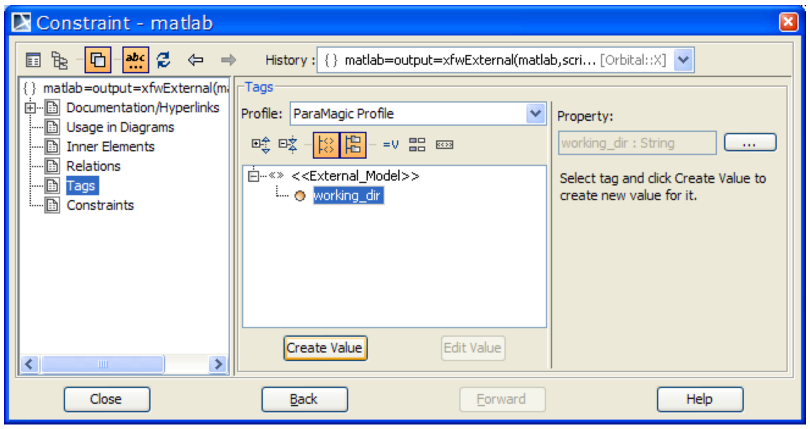


Figure 33: Specifying the location of M-file – selecting the tag to populate

4) Specify the location of the folder containing the M-file. For example, if the M-file is located under C:\Data\My_MATLAB_Files, specify the following value for the working_dir tag: C:\\Data\\My_MATLAB_Files.

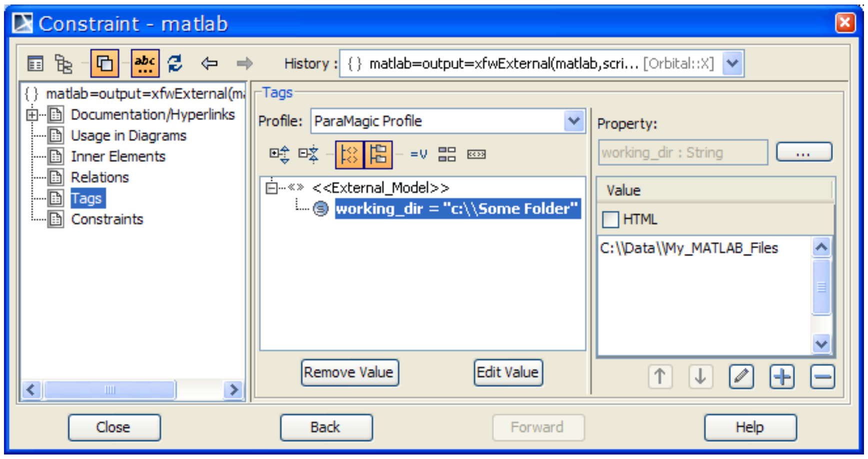


Figure 34: Specifying the location of M-file by populating the tag

Note that you only need to specify the location of the folder containing the M-file. The name of the M-file is specified in the constraint specification.

Step 6. *Define constraint properties typed by the wrapped constraint block*

After defining a constraint block that wraps MATLAB M-files, constraint properties could be defined for blocks whose values properties are to be related using the subject MATALB function/script. These constraint properties should be typed by the constraint block created in Step 4 above. When creating SysML parametric models using parametric diagrams, the constraint parameters of the constraint property should be connected to the value properties of the block. For example, as shown in Figure 37, a constraint property SHH (of type SimulinkHomeHeating constraint block) is defined for a block and its constraint parameters outtemp, daycyc, row, col, and cost are connected to a block's value properties Outdoors.Temp, DailyCycle, OutputRow, OutputColumn, and DailyCost respectively.

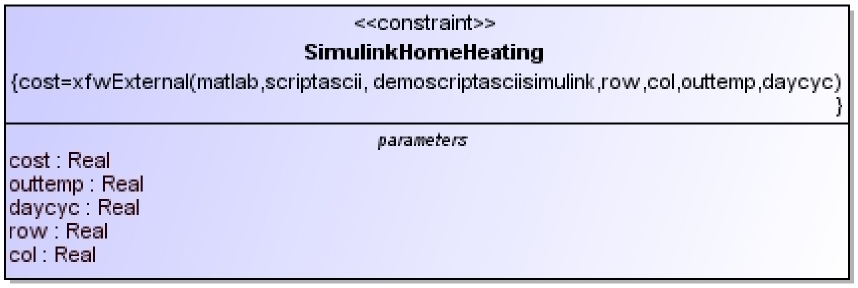
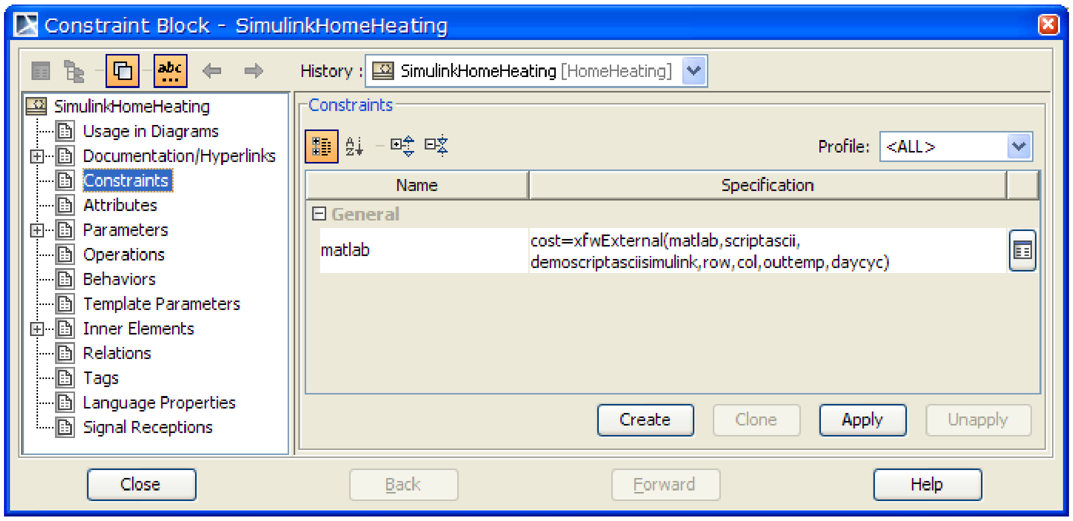


Figure 35: SysML constraint block to wrap MATLAB scripts



Name	Specification
General	
matlab	cost=xfwExternal(matlab,scriptascii, demoscriptasciisimulink,row,col,outtemp,daycyc)

Figure 36: Defining constraint specification for a constraint block to wrap MATLAB script M-file

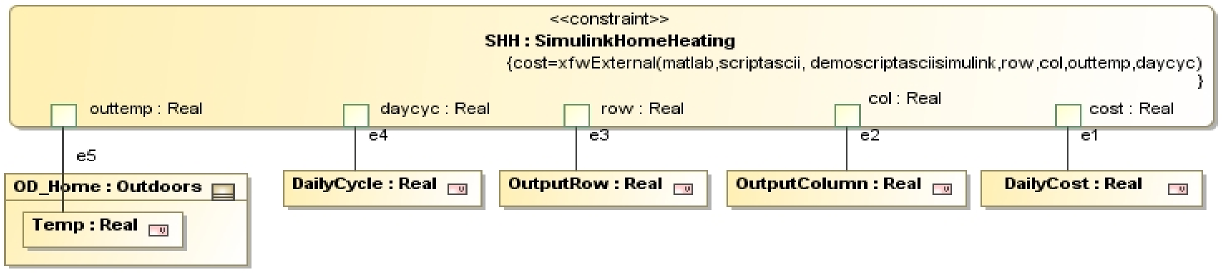


Figure 37: SysML constraint property typed by a constraint block (that wraps MATLAB scripts) and used in a SysML parametric diagram

7.3.1 Using MATLAB scripts

Unlike MATLAB functions, scripts do not have input arguments and output/return values. ParaMagic™ uses intermediate input and output files to transfer SysML instance values (givens) from MagicDraw to a MATLAB script before executing the script, and to transfer results obtained by executing the script to SysML instance values (targets). To use MATLAB scripts with ParaMagic™, follow the steps below after finishing Step 6 above.

Step 7. *Setup script M-file to read/write values from/to SysML instance model*

Since MATLAB scripts do not have input and output arguments, users must add commands to the beginning/end of the script to read/write values from/to SysML instance model. Follow the steps below to setup your script M-file to read values from input.txt file and write results to output.txt file.

1) Add commands to achieve the following at the beginning of your script M-file
   a) Define an array variable in the MATLAB script that will hold values of input parameters.
   b) Load the input.txt file to populate this variable.
   c) Assign the values to variables in the script

   For example, a variable insel is defined to contain values loaded from input.txt file. Then, four values contained in the insel variable are assigned to four variables in the script.

   ```
   inSel= load('input.txt');

   o1=inSel(1);
   o2=inSel(2);
   TempOutsite=inSel(3);
   Amplitute=inSel(4);
   ```

   Note that the order in which the values are written to the input.txt file is the order in which input parameters are listed in the constraint specification of the constraint block. As an example, for the constraint block in Figure 35, the input.txt will contain values of the variable in the following order: row, col, outtemp, daycyc.

2) Add a save command at the end of your script M-file to save the value of the solved variable—corresponding to the output parameter of the constraint property—in output.txt file. For example, to save the value of variable a, the following command is used.

   ```
   save('output.txt','a','-ASCII');
   exit
   ```

   The exit command ensures that the MATLAB session ends after script execution. This will avoid having multiple sessions of MATLAB running as the SysML model is solved multiple times.

ParaMagic^{TM} writes SysML instance values corresponding to the input parameters of a constraint property to a text file (input.txt) located in the same folder as the MATLAB script M-file. To import the value of the variables computed from script execution to the SysML instance model, ParaMagic reads a text file (output.txt) containing the variable value and located in the same folder as the MATLAB script M-file. Users do not need to worry about the input.txt and output.txt files created for transferring values between MagicDraw and MATLAB. These are automatically created and managed by ParaMagic^{TM}.

7.3.2 Using MATLAB functions

For using ParaMagic^{TM} with M-file functions, ensure that in Step 4 above, the constraint specification—for the constraint block that wraps the M-file function—uses the keyword function (as shown below) instead of scriptascii.

```
<out_param> = xfwExternal(matlab, function, <name_of_function_M-file>, <in_param_1>,...)
```

Figure 38 and Figure 39 below illustrate an example of a MATLAB function (DemoAddition) with 2 input parameters wrapped in a constraint block, which is then used to type constraint properties in a SysML parametric model.

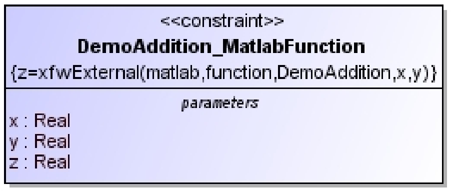

<<constraint>>
DemoAddition_MatlabFunction
{z=xfwExternal(matlab,function,DemoAddition,x,y)}
parameters
x : Real
y : Real
z : Real

Figure 38: SysML constraint block to wrap MATLAB function

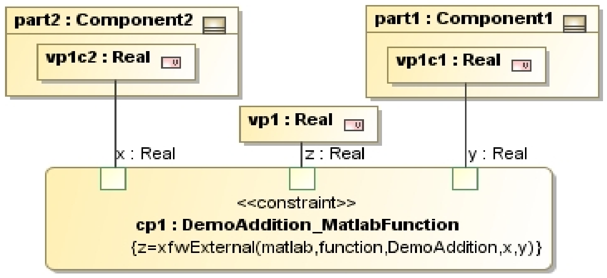


Figure 39: SysML constraint property typed by a constraint block (that wraps a MATLAB function) and used in a SysML parametric diagram

Follow the step below after completing Step 6 above.

Step 6. *Export return values to an output.txt file.*

Add a save command at the end of the MATLAB function to save the value of the output/return variable in output.txt file. The code snippet below shows the save and exit commands added at the end of the definition of function DemoAddition in a function M-file.

```
function z = DemoAddition(x,y)
z=x+y;
save('output.txt', 'z', '-ASCII')
exit
```

Note that functions can have input arguments and hence PM-MC does not require users to read values from an input.txt file (as in scripts). In the example below, the save command is added at the end of a function DemoAddition that returns the sum of two numbers. The sum is saved to output.txt file. As in the case of scripts, input parameters and return values of M-file functions could be single-valued or a multi-valued (e.g. array) but they cannot be complex data structures (such as an array of arrays, etc.).

8 TRADE STUDIES

With ParaMagic^{TM} 16.6, users can easily setup and run trade studies on their existing SysML models. The capability to run trade studies on SysML parametric models allows users to compute performance, reliability, cost, and other measures-of-effectiveness—especially those used to verify requirements—for a large set of system alternatives at each development phase, and then select the best-in-class alternatives for the next development phase. With ParaMagic^{TM} 16.6, trade studies can be now be performed from the earliest stages of system development. The overall process for setting up and running trade studies is as below.

1. Verify that your existing SysML instance model can be solved using ParaMagic^{TM}.
2. Setup a trade study
   a. Identify trade study inputs, outputs, and constants.
   b. Link inputs and outputs to Excel spreadsheets. ParaMagic^{TM} reads values of trade study input variables for all scenarios from linked Excel spreadsheets. After completion, the values of trade study output variables for all scenarios are written to the linked spreadsheets.
   c. Specify number of scenarios
3. Run trade study

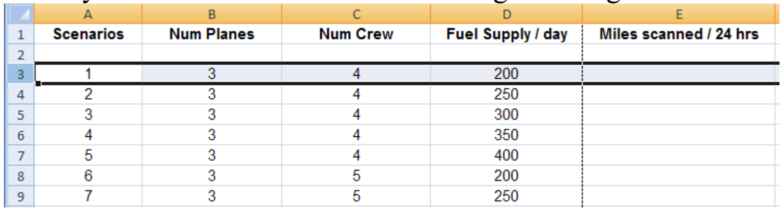
8.1 Operation

The detailed steps for setting up and running trade studies on SysML models are as follows. The process described below assumes that you have setup a SysML schema and instance model in the same manner that you do for regular ParaMagic^{TM} solving purposes—see the Tutorials document for details.

Step 1. Verify that your SysML instance model can be solved in ParaMagic^{TM}
The series of steps below are used to check if your SysML schema and instance models are structurally valid can be solved. Solving an instance model is similar to running a single scenario in a trade study.
1) *Browse the instance model*: Right click on the instance package and select ParaMagic → Browse. If the ParaMagic browser opens up, this implies that the schema and instance are structurally valid.
2) *Solve the instance model*: Click on the Solve button in the ParaMagic^{TM} browser. See section 6.1 for details. If the model solves correctly, it implies that your instance model
3) *Update SysML instance model*: Click on the Update to SysML button in the browser. Check that the target slot values are updated in the SysML instance model.

Step 2. Prepare Excel spreadsheet(s) with values of trade study input variables
1) Trade study variables are arranged in columns, and the scenarios are specified in rows.

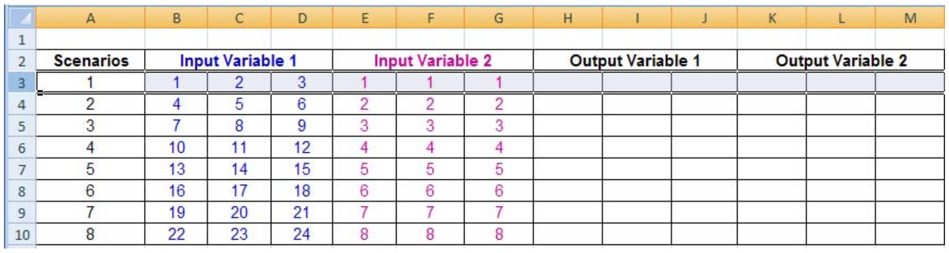
   All values of a trade study input variable should be in columns, such that each row in those columns contains values for a single scenario. In the spreadsheet shown below, NumPlanes, NumCrew, and Fuel Supply / day are the trade study input variables, and Miles scanned / 24 hrs is the output variable. Note that values for input/output variables are in columns. Each row, starting with row 3, represents the different trade study scenarios that will be solved using ParaMagic^{TM}.



	A	B	C	D	E
1	Scenarios	Num Planes	Num Crew	Fuel Supply / day	Miles scanned / 24 hrs
2					
3	1	3	4	200	
4	2	3	4	250	
5	3	3	4	300	
6	4	3	4	350	
7	5	3	4	400	
8	6	3	5	200	
9	7	3	5	250	

Figure 40: Trade study scenarios must be organized in rows—one scenario per row

For multi-valued variables (e.g. arrays), the values for each scenario should be in contiguous columns. For example, the values of Input Variable 1 and Input Variable 2 are arranged in columns for each scenario. Hence for scenario 1, Input Variable 1 = {1,2,3} and Input Variable 2 = {1,1,1}.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2	Scenarios	Input Variable 1			Input Variable 2			Output Variable 1			Output Variable 2		
3	1	1	2	3	1	1	1						
4	2	4	5	6	2	2	2						
5	3	7	8	9	3	3	3						
6	4	10	11	12	4	4	4						
7	5	13	14	15	5	5	5						
8	6	16	17	18	6	6	6						
9	7	19	20	21	7	7	7						
10	8	22	23	24	8	8	8						

Figure 41: Values of multi-valued variables are specified in contiguous columns for each scenario

2) Trade study variables may be linked to different workbooks/worksheets, and may have the first scenario specified in different rows for each variable, unlike as shown in the figures above.

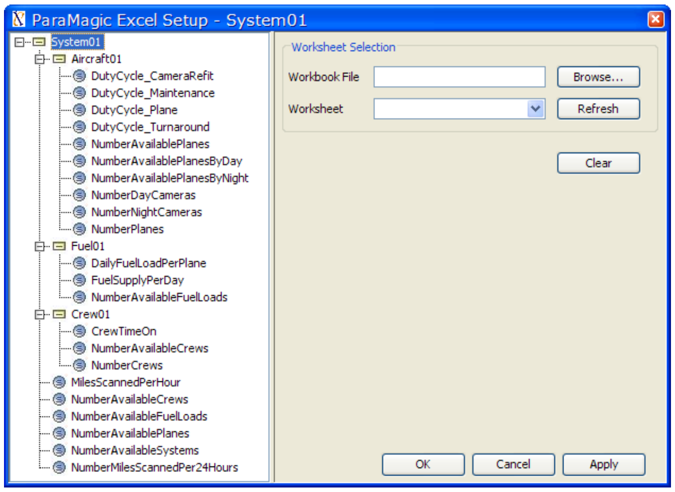
Step 3. Connect trade study variables to Excel spreadsheets

ParaMagic uses the following logic to identify trade study input and output variables, and constants:

a) Slots with causality "given" and Excel access mode "Read" are treated as trade study input variables.
b) Slots with causality "target" and Excel access mode "Write" are treated as trade study output variables.
c) Slots with causality "given" but with no connection to Excel are treated as trade study constants. Hence, the value(s) specified for these slots are repeated for each trade study scenario.

Causalities were assigned to all slots in Step 1 above. In this step, you will link slots corresponding to trades study inputs and output to Excel spreadsheets. To do this, follow the steps below:

1) *Launch Excel setup*: Right click on the instance package and select ParaMagic → Excel → Setup. This will launch the Excel setup utility, as shown below.



ParaMagic Excel Setup - System01

- System01
  - Aircraft01
    - DutyCycle_CameraRefit
    - DutyCycle_Maintenance
    - DutyCycle_Plane
    - DutyCycle_Turnaround
    - NumberAvailablePlanes
    - NumberAvailablePlanesByDay
    - NumberAvailablePlanesByNight
    - NumberDayCameras
    - NumberNightCameras
    - NumberPlanes
  - Fuel01
    - DailyFuelLoadPerPlane
    - FuelSupplyPerDay
    - NumberAvailableFuelLoads
  - Crew01
    - CrewTimeOn
    - NumberAvailableCrews
    - NumberCrews
  - MilesScannedPerHour
  - NumberAvailableCrews
  - NumberAvailableFuelLoads
  - NumberAvailablePlanes
  - NumberAvailableSystems
  - NumberMilesScannedPer24Hours

Worksheet Selection

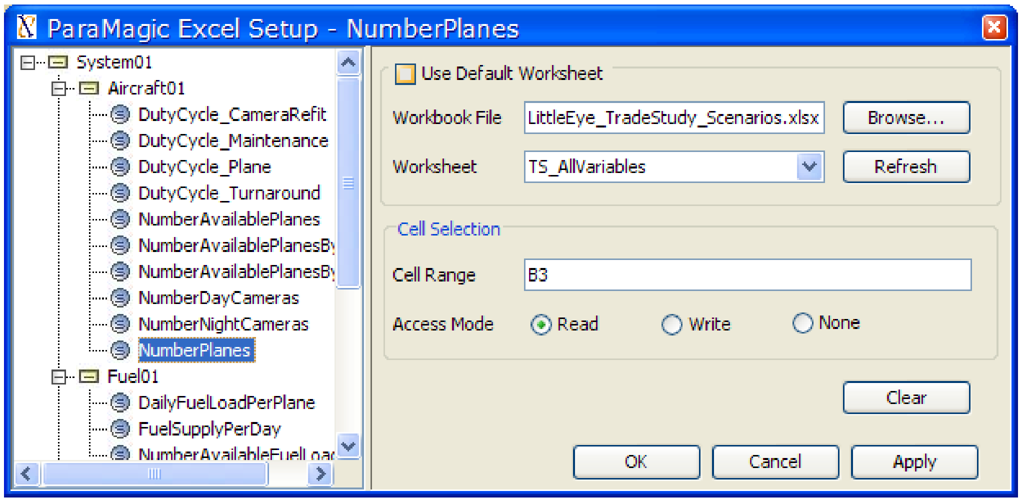
Workbook File

Worksheet

Figure 42: Use the Excel setup utility to link trade study inputs and outputs to spreadsheets

2) *Connect trade study inputs/outputs to Excel spreadsheets*: To do this, follow the steps below for each slot corresponding to a trade study input or output variable.
   a) Click on the slot in the instance tree on the left pane.

b) Specify the Excel workbook and worksheet that contains scenario values for this slot, as shown in Figure 43 below.



ParaMagic Excel Setup - NumberPlanes

- System01
  - Aircraft01
    - DutyCycle_CameraRefit
    - DutyCycle_Maintenance
    - DutyCycle_Plane
    - DutyCycle_Turnaround
    - NumberAvailablePlanes
    - NumberAvailablePlanesBy
    - NumberAvailablePlanesBy
    - NumberDayCameras
    - NumberNightCameras
    - NumberPlanes
  - Fuel01
    - DailyFuelLoadPerPlane
    - FuelSupplyPerDay
    - NumberAvailableFuelLoad

Use Default Worksheet [ ]

Workbook File: LittleEye_TradeStudy_Scenarios.xlsx

Worksheet: TS_AllVariables

Cell Selection

Cell Range: B3

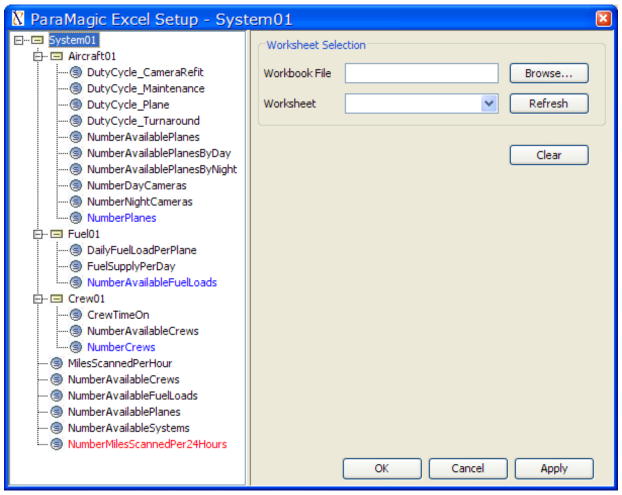
Access Mode: [✓ Read] Write None

Figure 43: *Use the Excel setup utility to link trade study inputs and outputs to spreadsheets*

c) Set the cell range to the cell(s) that contain value(s) for the first scenario. For single-valued slots, the cell range is a single cell. For multi-valued slots, the cell range is a set of contiguous cells in a single row. As shown above, the cell range for NumberPlanes is set to B3 (spreadsheet shown in Figure 40). Similarly, the cell range for Input Variable 2 and Output Variable 1 (spreadsheet shown in Figure 41) would be E3:G3 and H3:J3 respectively.

d) Set the access mode to
   i) Read for slots corresponding to trade study input variables.
   ii) Write for slots corresponding to trade study output variables.

e) Click on the Apply button.
   Since trade study input/output variables are setup to read/write from Excel, they are shown in blue/red color, as shown in Figure 44 below. For more details on how to use the Excel setup utility (PM-EC feature), refer to section 7.2.



ParaMagic Excel Setup - System01

- System01
  - Aircraft01
    - DutyCycle_CameraRefit
    - DutyCycle_Maintenance
    - DutyCycle_Plane
    - DutyCycle_Turnaround
    - NumberAvailablePlanes
    - NumberAvailablePlanesByDay
    - NumberAvailablePlanesByNight
    - NumberDayCameras
    - NumberNightCameras
    - NumberPlanes
  - Fuel01
    - DailyFuelLoadPerPlane
    - FuelSupplyPerDay
    - NumberAvailableFuelLoads
  - Crew01
    - CrewTimeOn
    - NumberAvailableCrews
    - NumberCrews
  - MilesScannedPerHour
  - NumberAvailableCrews
  - NumberAvailableFuelLoads
  - NumberAvailablePlanes
  - NumberAvailableSystems
  - NumberMilesScannedPer24Hours

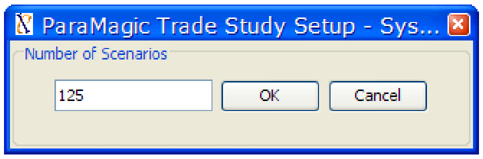
Worksheet Selection

Workbook File:

Worksheet:

Figure 44: *Slots setup to read (write) from Excel are shown in blue (red) color*

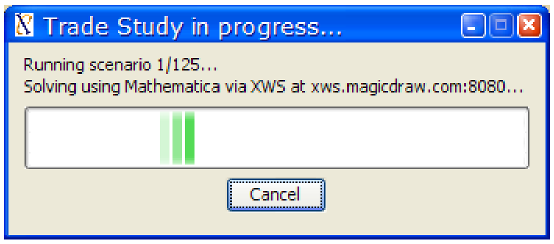
Step 4. Specify number of scenarios
Right click on the instance package and select ParaMagic → Trade Study → Setup. Specify the number of scenarios in the dialog box, as shown below. The value (say n) specified for the number of scenarios will be used by ParaMagic™ to construct n scenarios by reading the values in n rows (in spreadsheets connected to input variables) starting with the first row specified for each input variable.



125

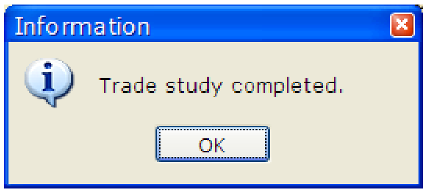
Figure 45: Specify number of scenarios for a trade study

Step 5. Run trade study
Ensure that all spreadsheets connected to trade study output variables are closed. Then, right click on the instance package and select ParaMagic → Trade Study → Run. The trade study progress window (as shown in Figure 46) indicates the specific scenario being run and the core solver being used.



Running scenario 1/125...
Solving using Mathematica via XWS at xws.magicdraw.com:8080...

Figure 46: Trade study progress window



Trade study completed.

Figure 47: Message to indicate completion of a trade study

Completion of a trade study is indicated by the message above (Figure 47).

Step 6. View trade study outputs and perform post-processing
Open spreadsheets connected to trade study output variables to see results. You can use Excel for post-processing the values, such as for computing statistical metrics or plotting output variables against input variables.

Note that trade studies can be performed with either Mathematica or OpenModelica as a core solver. SysML parametric models executed in trade studies may include all types of relations as solved using regular ParaMagic™ operation except for custom Mathematica relations that create plots for each scenario[20].

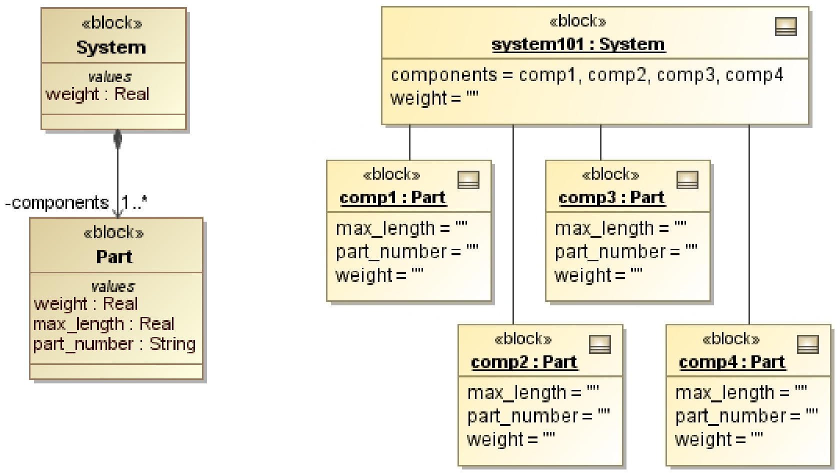
[20] Plots created for a scenario will overwrite those created for the previous scenario.

Copyright © InterCAX LLC. All Rights Reserved.

8.2 Limitations

The trade study capability in ParaMagic^{TM} 16.6 sp1 has the following limitations:

1) A trade study is based on a SysML instance model which represents the structure of all scenarios. The scenarios may differ in the values assigned to the slots but not the number of instances in the SysML instance model. Figure 48 illustrates a simple SysML schema model that represents a system composed of 1 or more parts. The figure also shows a SysML instance model that conforms to the schema model. The instance model represents a system (system101) with 4 components. If one were to setup a trade study for this example, it would be setup for the instance model. All scenarios will represent systems with 4 components. The scenarios may vary in slot values, i.e. the values of the slots max_length, part_number, and weight (of the component instances comp1, comp2, comp3, and comp4) may be different in the scenarios.



SysML Schema SysML Instance structure

Figure 48: Trade studies in ParaMagic^{TM} work on a fixed SysML instance structure

2) Values of trade study input variables must be explicitly specified for all scenarios in Excel spreadsheets. Specification of values as intervals and automated generation of scenarios by creating combinations of these intervals is not supported in this version of ParaMagic^{TM}.

3) Trade study runs are functionally similar to batch execution of a set of pre-defined scenarios. Automated generation of scenarios based on techniques to explore the design space is not supported. Contact us (info@intercax.com) for tailored interfaces to commercial tools, such as Isight[21] and ModelCenter[22], that provide design space exploration and optimization capabilities.

4) Plotting capabilities, such as the generation of single factor plots, interaction effects matrix plots, and carpet plots for trade studies, are not available natively with this version of ParaMagic^{TM}. Since trade study outputs are written to Excel spreadsheets, users may leverage the extensive plotting and post-processing capabilities of Excel.

[21] Isight: http://www.simulia.com/products/isight.html
[22] ModelCenter: http://www.phoenix-int.com/software/phx_modelcenter.php

9 UPDATES SINCE THE LAST RELEASE (PARAMAGIC™ 16.6)

ParaMagic ™ 16.6 sp1 has several efficiency-related improvements as compared to ParaMagic™ 16.6. The key user-oriented updates are as follows.

1) *Support for OpenModelica on Mac OS X*—With ParaMagic™ 16.6 sp1, Mac users will be able to use OpenModelica with ParaMagic™. See Step 5 in section 3.2 for installation and configuration instructions.

2) *Improvements in Excel R/W speed*—ParaMagic™ 16.6 sp1 has significant improvements in the read/write speed for Excel spreadsheets. This reduces the time to read/write a large set of values from minutes to few seconds.

3) *Excel R/W supports numeric and text values*—With ParaMagic™ 16.6 sp1, users will be able to read/write both numeric and text-based values from/to Excel spreadsheets to/from the SysML instance models. See section 7.2.2 for details.

4) *Improved Mathematica and OpenModelica model preparation*—ParaMagic™ 16.6 sp1 provides several efficiency-related improvements in generating Mathematica and OpenModelica models from SysML parametric models. End users will observe a faster solution time for large and complex models.

5) *Control the number of decimal places displayed in the browser*—ParaMagic™ 16.6 sp1 provides end users the control to specify the max number of decimal places to be displayed for values in the ParaMagic™ browser. The actual value can be seen by hovering the mouse pointer to the displayed value or when editing a given value. See section 6.2.5 for details.

6) *Support for recursive instance structure*—ParaMagic™ 16.6 sp1 provides support for part/reference/shared properties with the lower multiplicity bound equal to 0 (zero). Until ParaMagic™ 16.6, users had to populate instance slots even if the lower multiplicity bound for the corresponding property was equal to 0. With this capability and other improvements, users can now model recursive instance structures for roll-ups (mass, cost, etc.).

7) *Saving partial results for trade studies*—ParaMagic™ 16.6 sp1 allows users to save partial results (solved scenarios) of a trade study if it is canceled by a user during execution.

8) *Bundled ParaMagic and OpenModelica*—ParaMagic™ 16.6 sp1 onwards, users will be able to download a bundle (zip) from the ParaMagic download site[23] which includes both the ParaMagic plugin and OpenModelica (Win and Mac OS X).

In addition to the new features, ParaMagic™ 16.6 sp1 has bug-fixes and general usability improvements.

10 COPYRIGHT

10.1 Copyright statement from InterCAX LLC

This Users Guide, and the software described therein, are copyrighted. No part of this user guide or the described software may be copied, reproduced, translated, or reduced to any electronic medium or machine-readable form without the prior written consent of InterCAX LLC.

[23] ParaMagic™ download site - www.magicdraw.com/download/paramagic

10.2 Liability disclaimer from InterCAX LLC

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNERS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.