MapleMBSE User's Guide

MapleMBSE User's Guide

This document was produced using Maple and DocBook.

Contents

List of Figures

Introduction

MapleMBSE Overview

MapleMBSE™ gives an intuitive, spreadsheet based user interface for entering detailed system design definitions, which include structures, behaviors, requirements, and parametric constraints.

Related Products

MapleMBSE 2019 requires the following products.

- Microsoft® Excel® 2010 Service Pack 2, Excel 2013 (updated to most recent version, for details please see the Release Notes) or Excel 2016.
- Oracle® Java® SE Runtime Environment 8.

Note: MapleMBSE looks for a Java Runtime Environment in the following order:

1) If you use the -vm option specified in **OSGiBridge.init** (not specified by default), MapleMBSE will use it.

2) If your environment has a system JRE (meaning either: JREs specifed by the environment variables JRE_HOME and JAVA_HOME in this order, or a JRE specified by the Windows Registry (created by JRE installer)), MapleMBSE will use it.

3) The JRE installed in the MapleMBSE installation directory.

If you are using IBM® Rational® Rhapsody® with MapleMBSE, the following versions are supported:

- Rational Rhapsody Version 8.1.2 or Version 8.1.3
- Teamwork Cloud™ server 18.5

Note that the architecture of the supported non-server products (that is, 32-bit or 64-bit) must match the architecture of your MapleMBSE architecture.

Related Resources

Resource	Description
MapleMBSE Installation Guide	System requirements and installation instructions for MapleMBSE. The **MapleMBSE Installation Guide** is available in the **Install.html** file located either on your MapleMBSE installation DVD or the folder where you installed MapleMBSE.
MapleMBSE Applications	Applications in this directory provide a hands on demonstration of how to edit and construct models using MapleMBSE. They, along with an accompanying guide, are located in the Application subdirectory of your MapleMBSE installation.
MapleMBSE Configuration Guide	This guide provides detailed instructions on working with configuration files and the configuration file language.

For additional resources, visit **http://www.maplesoft.com/site_resources**.

Getting Help

To request customer support or technical support, visit **http://www.maplesoft.com/support**.

Customer Feedback

Maplesoft welcomes your feedback. For comments related to the MapleMBSE product documentation, contact doc@maplesoft.com.

1 Getting Started with MapleMBSE

In this chapter:

1.1 MapleMBSE Components

MapleMBSE provides a platform that you can use to map model information from diagram-based models into a table form. The diagram-based models may be models defined in UML or SysML or other formats specific to modeling tools. For the list of supported model formats see *Supported Model Formats (page 4)*.

The figure below shows the components that MapleMBSE works with and how they relate to each other.

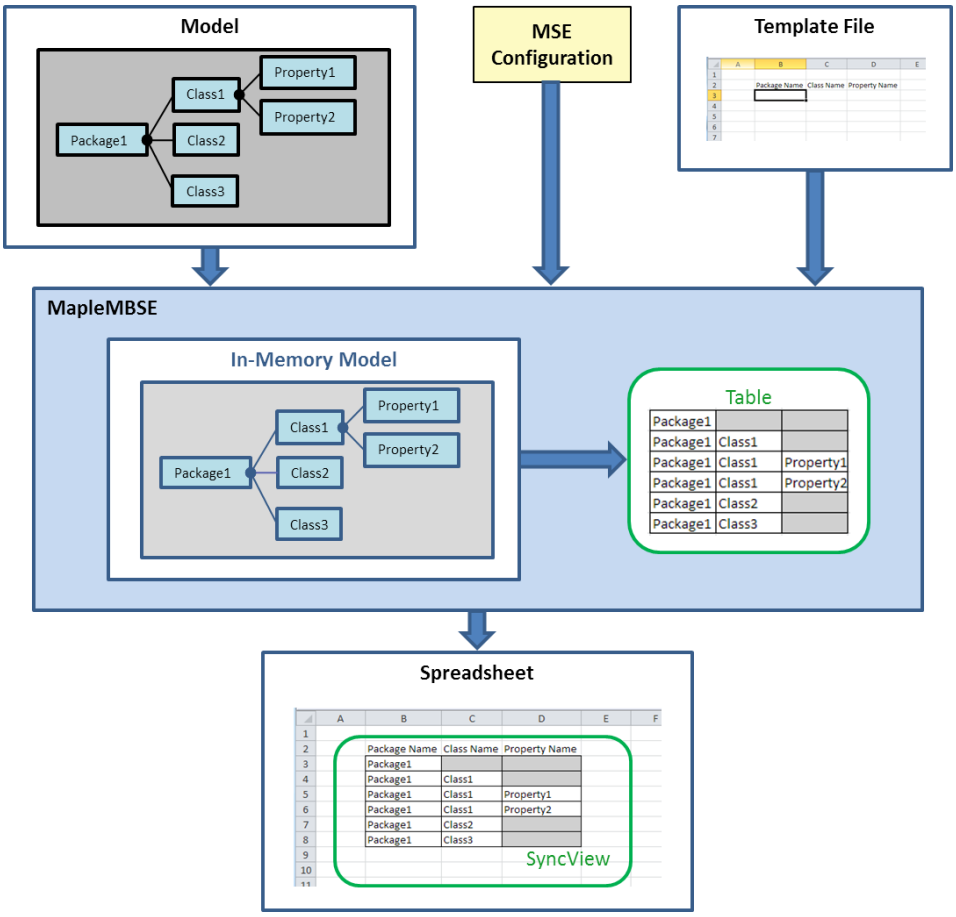


Figure 1.1: A Visual Summary of How MapleMBSE Works

- **Model**
  A model in the file system or on the server that you want to display and edit. See *Supported Model Formats (page 4)*.
- **Spreadsheet**
  An Excel spreadsheet that is used as a user interface for displaying and editing the model data.
- **In-Memory Model**
  When MapleMBSE is started it loads the given model into its internal memory. The data in the spreadsheet is synchronized with this in-memory model.

- **MSE Configuration**
  The configuration file (.MSE) is created by the user. It provides the rules for how the model is converted and represented in a spreadsheet. The rules specify the following:

1. How the data is to be extracted from the model and how it should be mapped into a table format. The table is shown in green in the Figure above.
2. How and where the extracted table is presented on the spreadsheet. The area of the spreadsheet representing the table SyncView. The SyncView is shown in green in **Figure 1.1**.

You can find detailed instructions about the MSE configuration language in the *MapleMBSE Configuration Language Guide*.

- **Template**
  An Excel file with the same name as the MSE configuration file located in the same directory as the MSE configuration file. The file defines formatting such as column width, headers, etc. for the given configuration file. For details see *MapleMBSE Configuration Language Guide*.

The relation between the model file, the model loaded in MapleMBSE's memory, and the spreadsheet is shown in the diagram below.

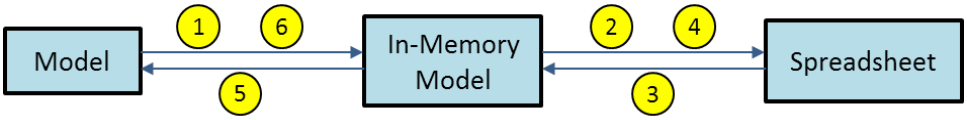


Figure 1.2: Relationship Between Model, In-Memory Model, and Spreadsheet

1. MapleMBSE loads the model from the given file into its internal memory.
2. The data from the in-memory model is loaded into the SyncView area of the spreadsheet.
3. User input on the SyncView area is converted into the changes to the in-memory model.
4. Any other places of the same SyncView or other SyncViews that are linked to the updated element get updated.
5. The changes to the in-memory model are applied to the model in the file system or on the server only when the spreadsheet is saved.
6. If the changes were made to the model in the file system or on the server they can be reloaded into the in-memory model by invoking Reload/Update, see *Using MapleMBSE Specific Functions (page 7)*.

Notes:

- Simultaneous operations on the same model by MapleMBSE and other tools are not supported.

- Modifying the model in other tools while being modified by MapleMBSE is not recommended.

Supported Model Formats

MapleMSBE 2019 supports the following version of the UML model format.

- UML® 2.0 **http://www.uml.org**.

The following models are supported if the corresponding adaptors are installed.

- IBM® Rational® Rhapsody® with MapleMBSE Rhapsody Adapter
- Models on Teamwork Cloud™ with MapleMBSE Teamwork Cloud Adapter

1.2 Launching MapleMBSE

Note: For launching with the No Magic Teamwork Cloud Adapter see *Using the No Magic Teamwork Cloud Adaptor (page 9)*.

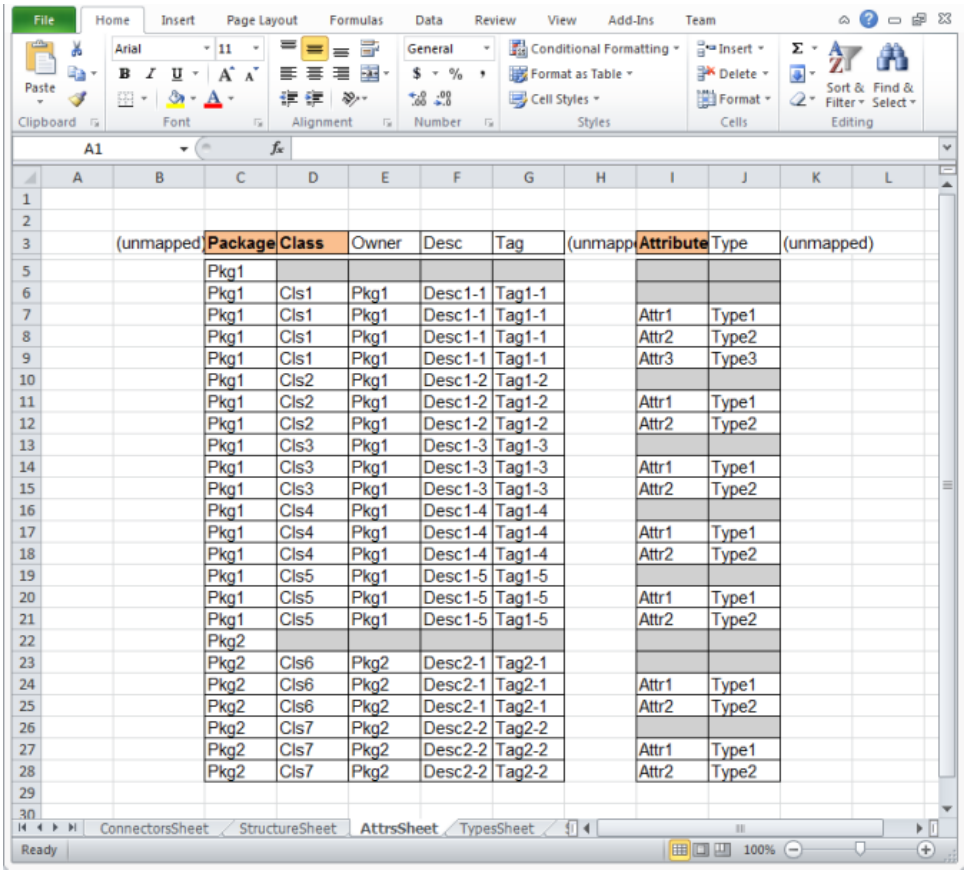
To launch MapleMBSE:

1. From the **Start** menu, select **All Programs > MapleMBSE 2019**.
2. Click **MapleMBSE 2019**.
3. Browse to the location of your MapleMBSE configuration file (configuration files have an *.MSE* file extension), select the configuration file, and then click **Open**.

   Tip: If you do not have a configuration file, you can open the sample configuration file, **<Install Directory>\Example\MechSheet\MechSheet.tritest.MSE**, where **<Install Directory>** is the install location of **MapleMBSE 2019.0**.
4. Browse to the location of your model file, select the model file, and then click **Open**.

   Tip: If you do not have a model file, you can open the sample model file, **<Install Directory>\Example\MechSheet\MechSheet-sample1.tritest**, where **<Install Directory>** is the install location of **MapleMBSE 2019.0**.

Microsoft Excel opens with the model information inserted into the spreadsheet (see **Figure 1.3**). You can edit the models in the spreadsheet by simply modifying cells and inserting rows. The detailed description of how MapleMBSE displays the model information in the spreadsheet and how to edit different aspects see *Editing the Model Spreadsheet MapleMBSE (page 17)*.To save your model you simply save this Excel file. For saving models opened with the No Magic Teamwork Cloud Adapter see *Using the No Magic Teamwork Cloud Adaptor (page 9)*. More about saving is given in *Saving Model (page 8)*.



	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3		(unmapped)	**Package**	**Class**	Owner	Desc	Tag	(unmapp	**Attribute**	Type	(unmapped)	
5			Pkg1									
6			Pkg1	Cls1	Pkg1	Desc1-1	Tag1-1					
7			Pkg1	Cls1	Pkg1	Desc1-1	Tag1-1		Attr1	Type1		
8			Pkg1	Cls1	Pkg1	Desc1-1	Tag1-1		Attr2	Type2		
9			Pkg1	Cls1	Pkg1	Desc1-1	Tag1-1		Attr3	Type3		
10			Pkg1	Cls2	Pkg1	Desc1-2	Tag1-2					
11			Pkg1	Cls2	Pkg1	Desc1-2	Tag1-2		Attr1	Type1		
12			Pkg1	Cls2	Pkg1	Desc1-2	Tag1-2		Attr2	Type2		
13			Pkg1	Cls3	Pkg1	Desc1-3	Tag1-3					
14			Pkg1	Cls3	Pkg1	Desc1-3	Tag1-3		Attr1	Type1		
15			Pkg1	Cls3	Pkg1	Desc1-3	Tag1-3		Attr2	Type2		
16			Pkg1	Cls4	Pkg1	Desc1-4	Tag1-4					
17			Pkg1	Cls4	Pkg1	Desc1-4	Tag1-4		Attr1	Type1		
18			Pkg1	Cls4	Pkg1	Desc1-4	Tag1-4		Attr2	Type2		
19			Pkg1	Cls5	Pkg1	Desc1-5	Tag1-5					
20			Pkg1	Cls5	Pkg1	Desc1-5	Tag1-5		Attr1	Type1		
21			Pkg1	Cls5	Pkg1	Desc1-5	Tag1-5		Attr2	Type2		
22			Pkg2									
23			Pkg2	Cls6	Pkg2	Desc2-1	Tag2-1					
24			Pkg2	Cls6	Pkg2	Desc2-1	Tag2-1		Attr1	Type1		
25			Pkg2	Cls6	Pkg2	Desc2-1	Tag2-1		Attr2	Type2		
26			Pkg2	Cls7	Pkg2	Desc2-2	Tag2-2					
27			Pkg2	Cls7	Pkg2	Desc2-2	Tag2-2		Attr1	Type1		
28			Pkg2	Cls7	Pkg2	Desc2-2	Tag2-2		Attr2	Type2		
29												
30												

Figure 1.3: Model Information Inserted

Note: You can also start MapleMBSE by double-clicking a configuration file from your file manager. This skips the configuration file selection step and takes you directly to the **Open Model File** dialog.

Note: You can only run one instance of MapleMBSE at a time. If you try to open a model while another one is already opened by MapleMBSE you will get the error shown in **Figure 1.4**. You must close the open model before opening a new one.

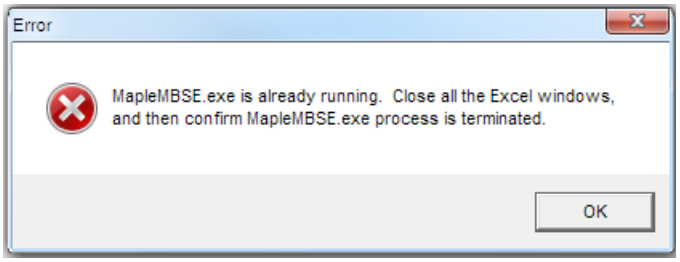


Figure 1.4: Error While Attempting to Run More Than One Model at a Time

Launching from Command Line

MapleMBSE can be launched from the command line by executing the MapleMBSE.exe command in the installation directory. The command has the following syntax.

MapleMBSE.exe [/R <memsize>] [/L <logfile>] [/E <excel.exe>] [[/I | CONFIG(*.MSE)] [MODEL]] |

- If you specify **CONFIG** (must be *.MSE) and **MODEL** files, it launches MapleMBSE with these files. If you do not specify a **MODEL** or **CONFIG** file, you will be asked for these files in a subsequent dialog.
- If you specify **/I**, MapleMBSE.exe will always ask for the MSE file in a dialog and regard the first non-slash argument as a model file.
- **MODEL** can be a local file or URI. Currently, URIs for Teamwork Cloud in the following format are allowed, and if you omit some items, MapleMBSE will ask for them in the dialog
  = twc://[SERVER:PORT][/PROJECT][?[username=USERNAME][&password=PASSWORD]]
- If you do not specify any file arguments, it checks if MapleMBSE.xlsm exists in the AddIns directory inside the directory with MapleMBSE.exe. If it is yes, launch it.
- **/R <memsize>**: (optional, default=905) Memory address space in MB to be reserved for Java VM in the Excel process.
- **/L <logfile>**: (optional, default="MapleMBSE_Launcher.log") Log file name, relative to %USERPROFILE% or in absolute path.
- **/E <excel.exe>**: (optional) Path of EXCEL.EXE
  If it's not specified, the registry entries will be used:
  = HKEY_CLASSES_ROOT\CLSID\<CLSID of "Excel.Application">\LocalServer32
  = HKEY_CLASSES_ROOT\CLSID\<CLSID of "Excel.Application">\LocalServer
  = HKEY_CLASSES_ROOT\Wow6432Node\CLSID\<CLSID of "Excel.Application">\LocalServer32

= HKEY_CLASSES_ROOT\Wow6432Node\CLSID\<CLSID of "Excel.Application">\LocalServer

1.3 Using MapleMBSE Specific Functions

To access MapleMBSE specific functions, select the **Add-Ins** menu, and then expand **MapleMBSE**. Here you will find the following items:

- Reload/Update the Model Spreadsheet
- Export to Excel file
- Import Excel file
- Save copy as

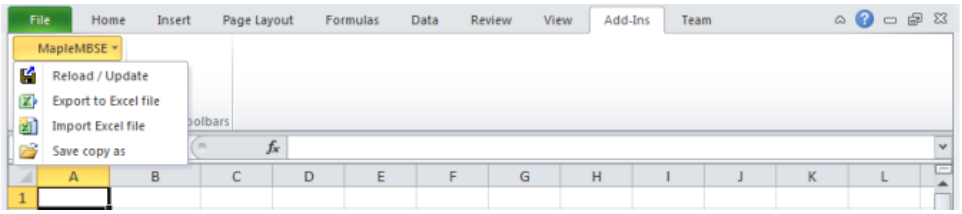


Figure 1.5: MapleMBSE Add-In Menu

Reload/Update Model Spreadsheet

You can refresh your current model so that it reflects any changes made to the model outside of MapleMBSE.

To reload your model:

1. Select **Add-Ins > MapleMBSE > Reload/Update**.
2. In the **Reload/Update** dialog window, click **OK**. **Note:** Reloading the model may discard any unsaved changes.
3. The model is reloaded in the Model Spreadsheet.

Note: Simultaneous operations on the same model by MapleMBSE and other tools are not supported. It is recommended not to modify the model in other tools when it is modified by MapleMBSE.

Exporting and Importing Excel Files

Select **Add-Ins > MapleMBSE > Export to Excel file** to save the current worksheet to an Excel file.

Select **Add-Ins > MapleMBSE > Import Excel file** to load an Excel file into the model you currently have open.

Saving a Copy of the Current Model

You can save the current model to another model file and keep the current model file unchanged. This function allows you to take a snapshot of the model. The in-memory model is still associated with the original model file. Any changes you save after creating the snapshot will only be saved to the original model and not to the snapshot.

To save a copy of the current model:

1. Select **Add-Ins > MapleMBSE > Save Copy As**.
2. Enter a filename and location.
3. Click **Save**.

1.4 Saving Model

Note: For saving models opened with the No Magic Teamwork Cloud Adapter see *Using the No Magic Teamwork Cloud Adaptor (page 9)*.

There are several options how a model data can be saved:

- **Save the in-memory model into the model file.**
  The changes done in a spreadsheet are performed on the in-memory model. To save the changes from the in-memory model to the model file use any of the Excel's usual options for Saving.

- Select **File > Save**
- Press **Ctrl** + **S**
- Click **Save** icon at the top of the window

- **Save the in-memory model as another model file**.
  To save the in-memory model into another model, **File > Save As**, enter a filename and location, click **Save**. The in-memory model is then linked to this new model file. Any future save operations will be performed on this file.
- **Save a copy of the model**.
  See *Saving a Copy of the Current Model (page 8)* to save a copy of the model and to keep the in-memory model linked with the original file.
- **Save the data as an Excel file**.
  To save the model data to an Excel workbook file, select **Add-Ins > MapleMBSE > Export to Excel file**.

1.5 Closing MapleMBSE

When you close a workbook editing the model, MapleMBSE is also closed. If there is no other open workbook, Excel will also quit. The following dialog opens reminding you to close all running instances of Excel before you can use MapleMBSE again.

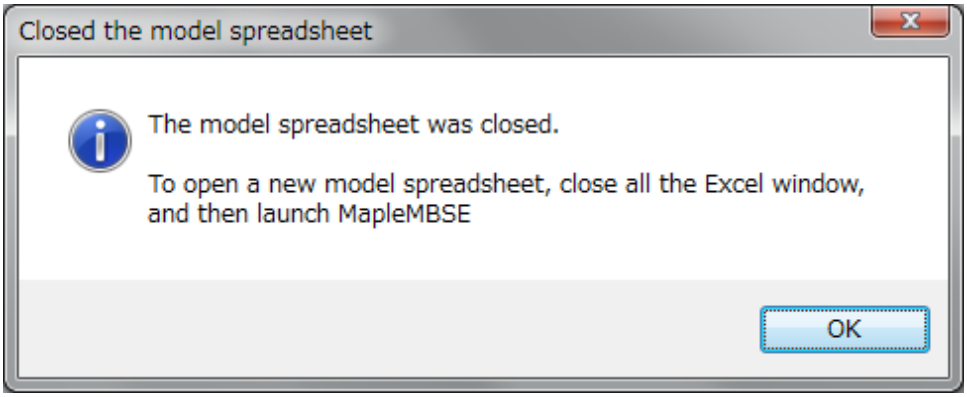


Figure 1.6: Reminder to Close Excel Instances Before Restarting MapleMBSE

All MapleMBSE Excel windows must be closed. The rules for closing Excel windows not related to MapleMBSE are as follows.

- Excel workbooks opened before MapleMBSE was launched can stay open.
- Excel workbooks opened after MapleMBSE was launched must be closed.

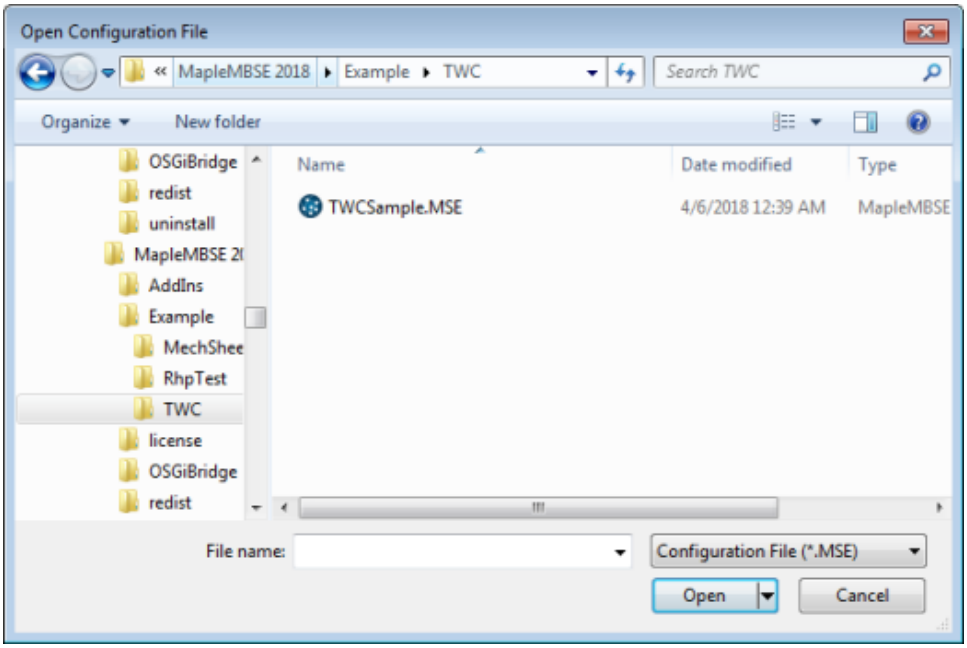
1.6 Using the No Magic Teamwork Cloud Adaptor

The Teamwork Cloud (TWCloud) Adaptor in MapleMBSE allows you to save changes you have made to your model in the Model Spreadsheet to the TWCloud repository.

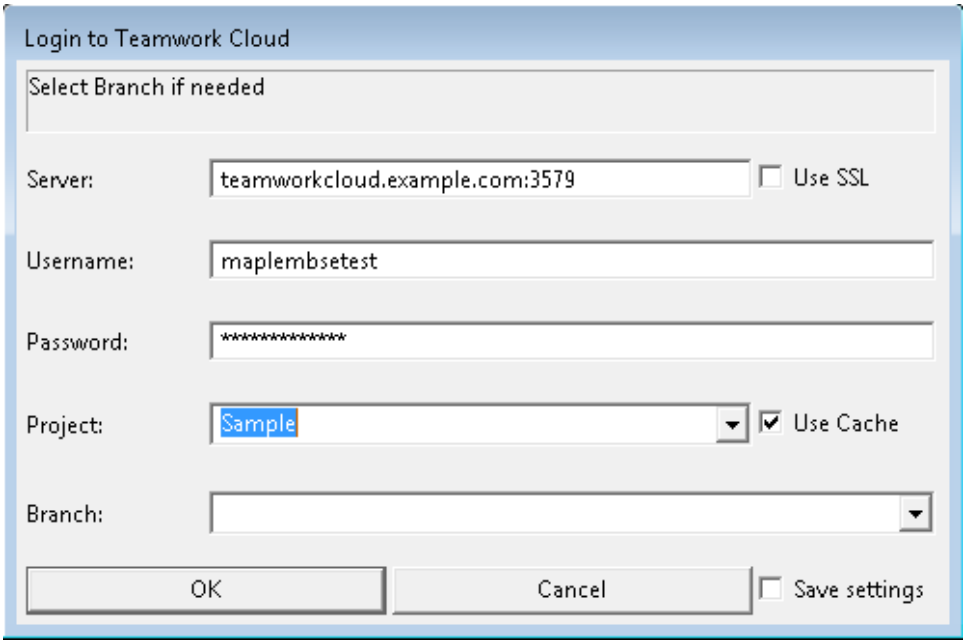
Launching the TWCloud Model in MapleMBSE

To launch your TWCloud model in MapleMBSE:

1. From **Start->All Programs->MapleMBSE 2019** select **Open Teamwork Cloud Model**.
2. Select your TWCloud configuration file.



3. In the **Login to Teamwork Cloud** dialog window, enter your TWCloud project name, username, password, server, project name and the branch the project is stored in. To use an SSL connection between TWCloud and MapleMBSE, select **Use SSL**. For more information, see *Setting up SSL Certificates (page 11)*. To download your model files launch them from a local folder, select *Caching Model Files (page 11)*. For more information see Cache. If you check **Save settings**, MapleMBSE saves the successful login information except for a password and they are automatically filled in the next login.



4. Click **OK**.

Setting up SSL Certificates

For SSL connections, you need to put JKS and PASS files into **<MAPLEMBSE-INSTALLATION-DIRECOTRY>**\certs. By default, TWC adapter will use cert.jsk and cert.pass. By specifying the "*ssl*" parameter in TWC URI, you can configure it. For example, if you launch *MapleMBSE TWCSample.MSE twc:///?ssl=cert2*, cert2.jks and cert2.pass are used for SSL connection.

Caching Model Files

Select the **Use Cache** option to have the MapleMBSE TWC Adapter cache files to a local folder, where the files will be loaded from in the future.

The default location of the cache folder is **<user.home>/.twcloud/<version #>/cache/**

MapleMBSE will update the cache automatically whenever a user performs any of the following operations:

- Reload or update the model.
- Commit changes to the Teamwork Cloud.

When you close a model, the cache files are deleted from the folder and any associated memory resources are released.

The Commit Number Information in the Workbook Title Bar

Once you have launched your TWCloud Model in MapleMBSE you will notice that TWCloud commit information for the model is displayed in the Excel Workbook title bar, as shown in **Figure 1.7**.

The format for the uri displayed in the title bar is:

```
Twc://teamworkcloud.example.com/<ProjectName>?username=maplembsetest#<CommitNumber>
```

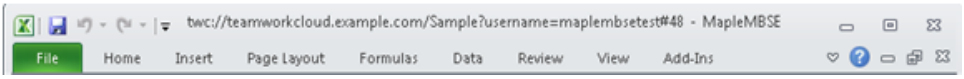


Figure 1.7: Commit Information Displayed in Title Bar

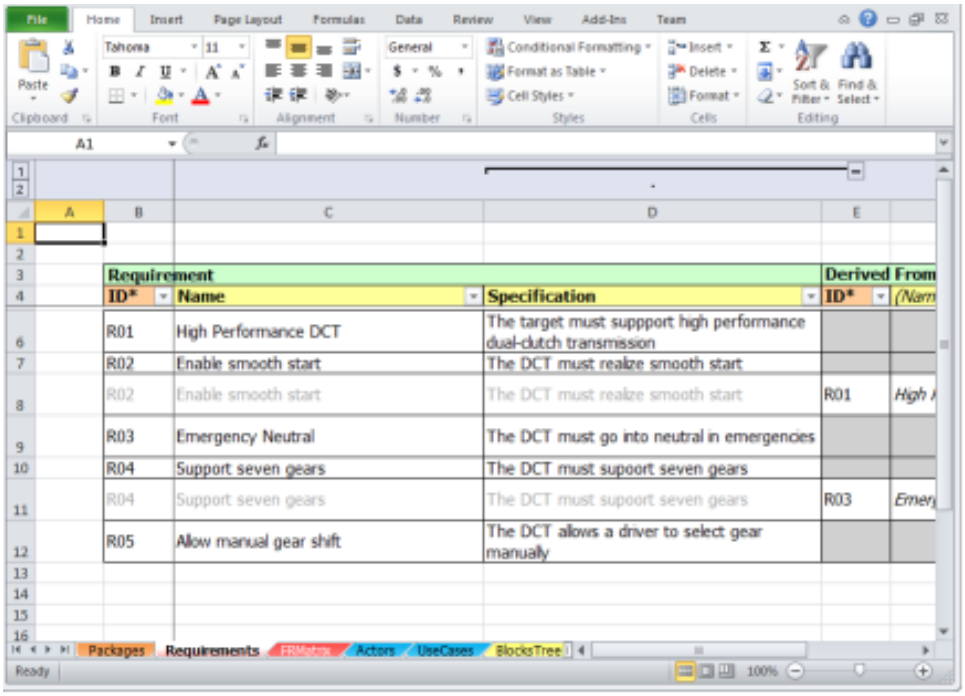
Updating the Commit Number

MapleMBSE updates the commit number using either of these methods:

- If you make changes to the model and commit those changes to the TWCloud, the commit number is updated.
- If someone else updates the model, and commits their changes to the TWCloud, then you reload or update the model in MapleMBSE. At this point, MapleMBSE will retrieve the latest resource and update the commit number.

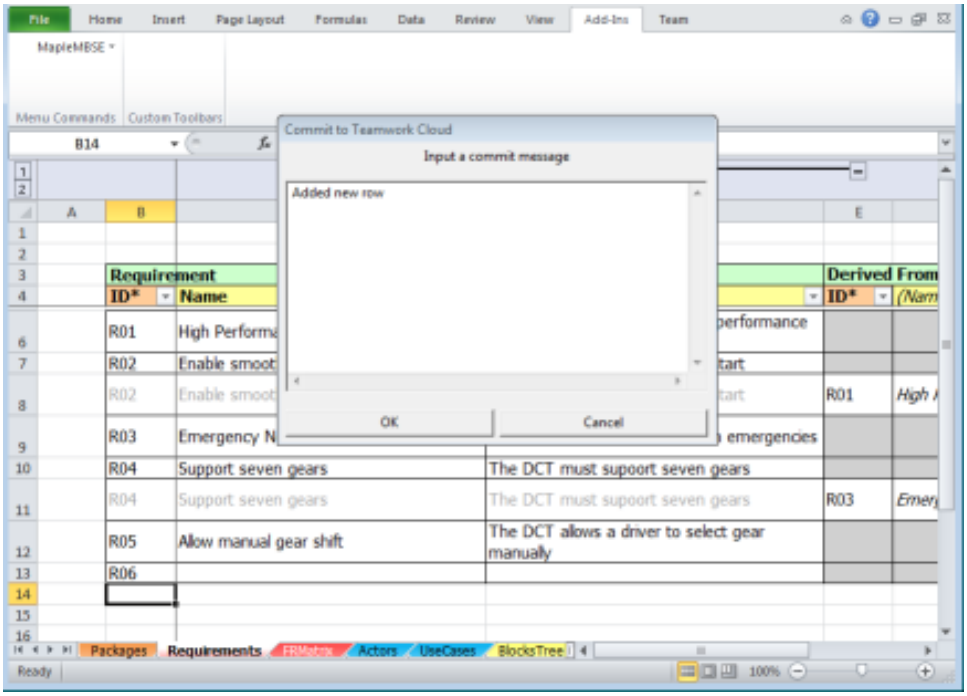
Editing and Saving Changes to Your Model

1. Make changes to the model in MapleMBSE.

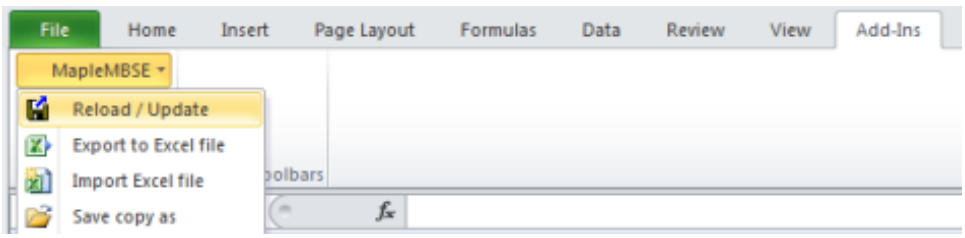


	Requirement			Derived From	
	ID*	Name	Specification	ID*	(Nam
6	R01	High Performance DCT	The target must suppport high performance dual-clutch transmission		
7	R02	Enable smooth start	The DCT must realize smooth start		
8	R02	Enable smooth start	The DCT must realize smooth start	R01	High
9	R03	Emergency Neutral	The DCT must go into neutral in emergencies		
10	R04	Support seven gears	The DCT must supoort seven gears		
11	R04	Support seven gears	The DCT must supoort seven gears	R03	Emer
12	R05	Allow manual gear shift	The DCT allows a driver to select gear manually		

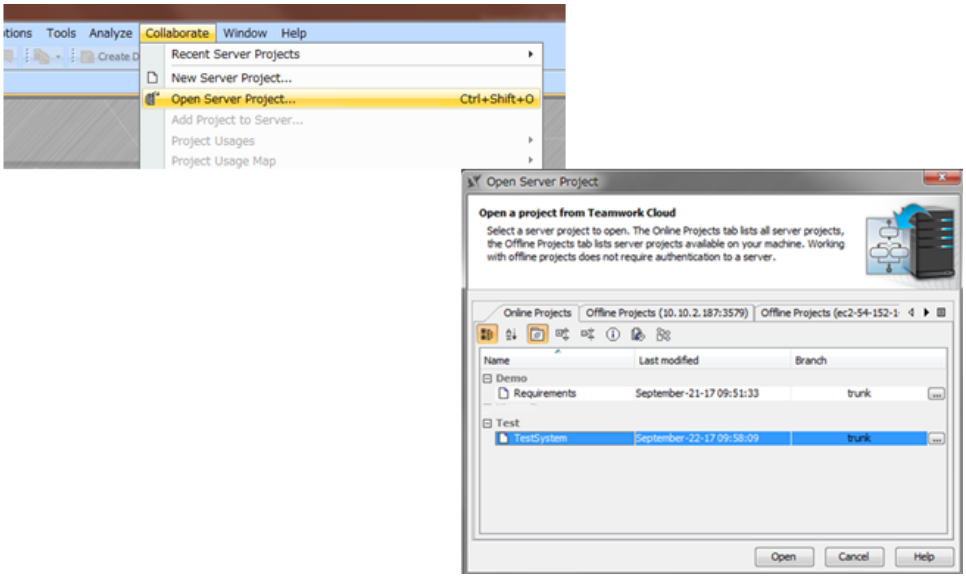
2. Single-click **Save** to save your changes to a (local) snapshot.
3. Double-click on **Save** to save to the TWCloud repository. The **Commit to Teamwork Cloud** dialog appears.



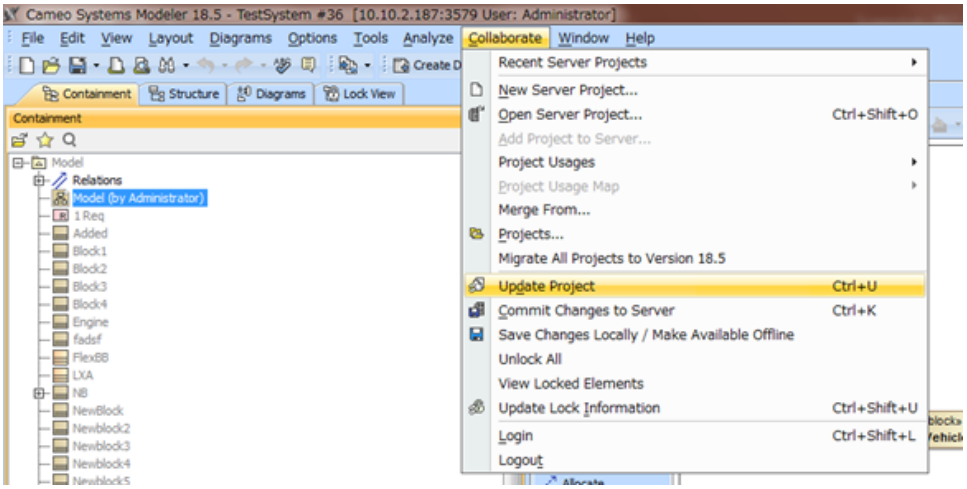
4. Enter a commit message, then click **OK**. The changes will be committed to the TWCloud repository, to be shared with other users.
5. Reload/Update the model in MapleMBSE.



6. Open the Server Project in MagicDraw™/Cameo Systems Modeler™.



7. Update the Project in MagicDraw/Cameo Systems Modeler.



Automatically Saving Model Changes

The TWCloud Adapter in MapleMBSE automatically creates a snapshot of model changes and uses this snapshot to recover after a MapleMBSE crash.

MapleMBSE will check periodically (every ten minutes) for a one minute period of inactivity. At this time, a snapshot will be automatically created.

Loading Models after a Crash

The model resource that is loaded after a crash depends on the most recent save or sync operation performed just before the crash.

For example, if the last operation before a crash was commiting changes to the teamwork cloud, MapleMBSE will load the snapshot that resides in the Teamwork Cloud.

However, if the last operation was saving to your local drive, MapleMBSE will load this local snapshot.

Finally, If the autosave has been initiated just before a MapleMBSE crash, a dialog window will be displayed after recovering from the crash letting you select which snapshot to load.

2 Editing the Model Spreadsheet MapleMBSE

In this chapter:

2.1 Workbook and Layout

The content of a workbook for a model is controlled by the MSE configuration file. The configuration file specifies the rules for what model data should be displayed in a spreadsheet and how to display this data. The remainder of this chapter reviews how the data can be displayed and what implications it has on user interactions with a spreadsheet. For detailed instructions about the rules governing what model data should be displayed, see the **MapleMBSE Configuration Language Guide**.

The configuration file specifies how many spreadsheets a workbook has and where and how the SyncViews are placed on the spreadsheets. Any user interactions that may change those specifications will result in an error and the actions will not be performed. In particular, the following information is specified for a workbook by the MSE configuration.

- The number of spreadsheets
- The names of the spreadsheets
- The order of the spreadsheets

Adding or deleting, renaming or moving the spreadsheets will result in an error. Note that the order of the spreadsheets is also controlled by the template file. See the **MapleMBSE Configuration Language Guide** for details.

For a SyncView, the MSE configuration specifies the cell in a spreadsheet where the SyncView area starts, the number of fields (see *Operations Overview (page 19)*) and the type of the SyncView placement. Any action that moves the location of the SyncView on the spreadsheet or interferes with the number of fields will be blocked by MapleMBSE. The specific actions that interfere with the number of fields depend on the placement type. There are three types of SyncView placement: a vertical table, a horizontal table, and a matrix.

- **Vertical Table**
  A vertical table is a table where elements are arranged vertically: each row represents a model element, and each column represents a particular attribute of that element. The example is given in the Figure below. The MSE configuration specifies the cell from

which the SyncView starts (B3 in the example). Inserting or deleting columns or rows in front of that cell (rows above row 3 or columns to the left of the column B) would shift the SyncView which would contradict the specification and result in an error. The configuration also specifies the number of columns for a vertical table (3 columns starting with column B). Inserting or deleting columns in that area (from B to D) would result in an error. The number of rows is dictated by the number of elements. Adding or deleting rows results in adding new or deleting existing elements.

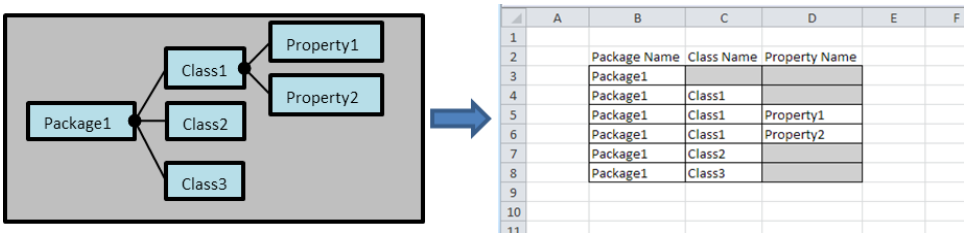


Figure 2.1: Vertical Table

- **Horizontal Table**
  A horizontal table is a table where elements are arranged horizontally: each column represents a model element, and each row represents a particular attribute of that element. The example is given in the Figure below. The MSE configuration specifies the cell from which the SyncView starts (C2 in the example). As in the case of the vertical table, inserting or deleting columns or rows in front of that cell (rows above row 2 or columns to the left of the column C) is forbidden. For a horizontal table the configuration specifies the number of rows in the table (3 rows starting with row 2). Inserting or deleting rows in that area (from row 2 to row 4) would result in an error. The number of columns corresponds to the number of elements. Adding or deleting columns results in adding new or deleting existing elements.

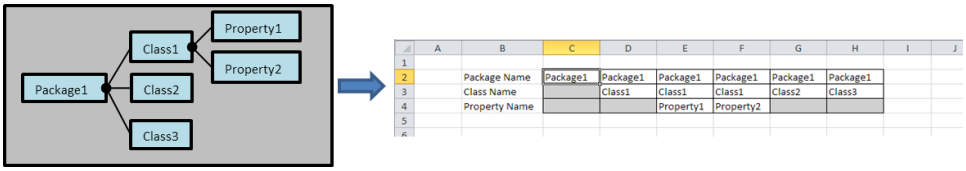


Figure 2.2: Horizontal Table

- **Matrix**
  A matrix combines both a vertical and a horizontal table and also provides a matrix form to indicate a relationship between the two tables. It is convenient to use to represent dependencies among model elements. In the example,**Figure 2.3**, the matrix represents the dependency of dependent classes on the supplier classes. The dependent classes are represented in the vertical table (outlined in green), and their supplier classes are in the hori-

zontal table (blue). The matrix (red) is used to indicate the dependencies. In the example Class1 from Package1 depends on Class2 from Package1 and Class3 from Package1. The MSE configuration specifies the cell where the matrix part starts (D5 in the example). The placements of the vertical and horizontal tables are determined from there. Inserting or deleting rows or columns before that cell (the rows above row 5 or columns to the left of the column D) is not allowed. The numbers of columns and rows in the matrix are determined by the number of elements represented by the vertical and horizontal tables, respectively. Inserting or deleting rows or columns in the region that affects the matrix area (rows between 5 and 7 or columns between D and F) is allowed and results in adding or deleting elements represented by the vertical or horizontal table, respectively.

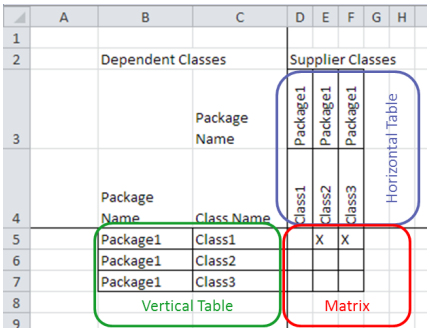


Figure 2.3: Dependency Matrix

2.2 Operations Overview

MapleMBSE allows users to edit the models via tables created inside the SyncView area by translating insertions, deletions, and updates of table rows into additions, removals and updates of model elements, respectively.

The basic elements of data that MapleMBSE displays on a spreadsheet are SyncView, records, and fields. The area on a spreadsheet representing model data is called the SyncView, outlined in red in **Figure 2.4**. For simplicity, the SyncView is a vertical table in this example (see Vertical Table in *Workbook and Layout (page 17)*). Only operations in the SyncView region of a spreadsheet will be transferred to the operations on the model. Operations in other places of a spreadsheet do not affect the model. The empty row (or column) at the end of the vertical (or horizontal) SyncView is used to insert new elements and is called the insertion area (green dashed line in **Figure 2.4**). Each SyncView corresponds to a set of model elements. Each element corresponds to a record, represented by a row in a vertical table (outlined in blue in **Figure 2.4**). Each record is an array of fields, represented by cells in the row. Each field represents some property of the model element.

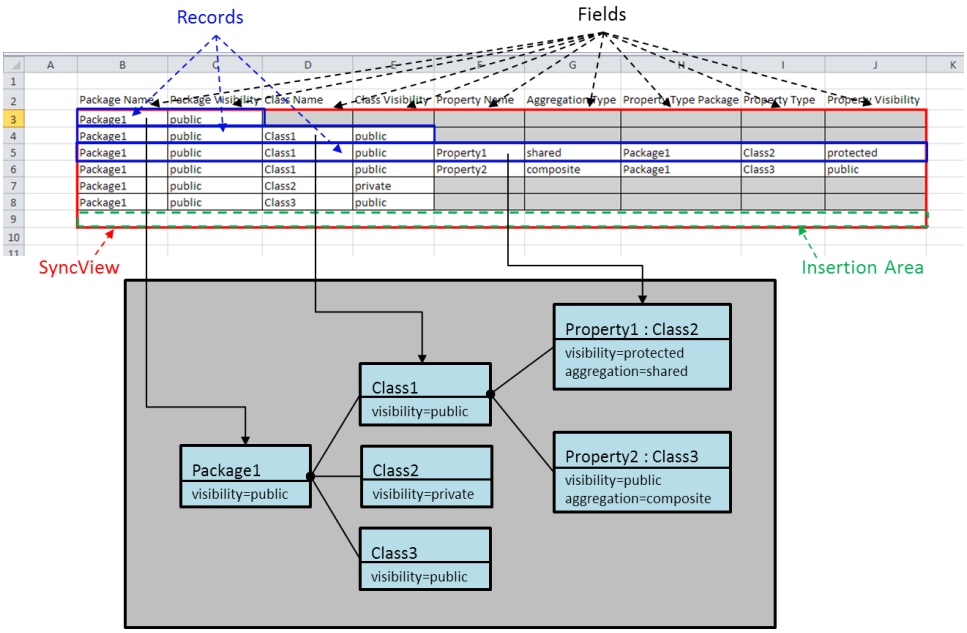


Figure 2.4: Data Elements Displayed By MapleMBSE

The SyncView areas for a workbook can be looked up in the **Name Box** (the box to the left of the formula bar). See **Figure 2.5**. Expanding the list in the **Name Box** gives the list of all SyncView areas on all spreadsheets in the workbook. The names include the names of the spreadsheets and the names of the SyncView areas defined by the MSE configuration.

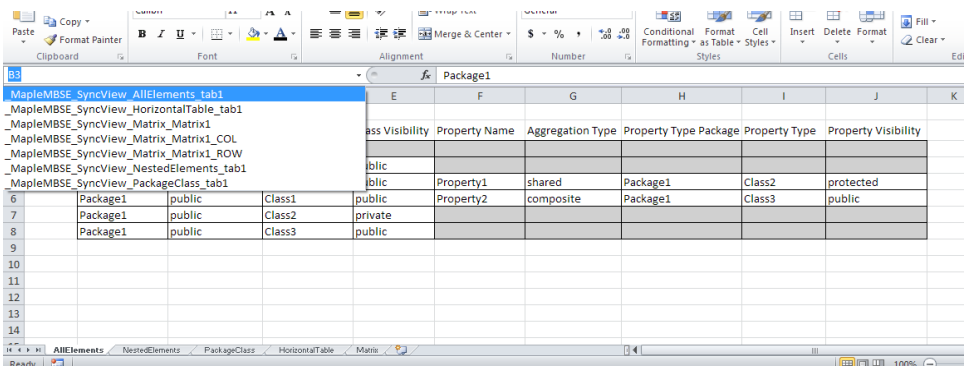
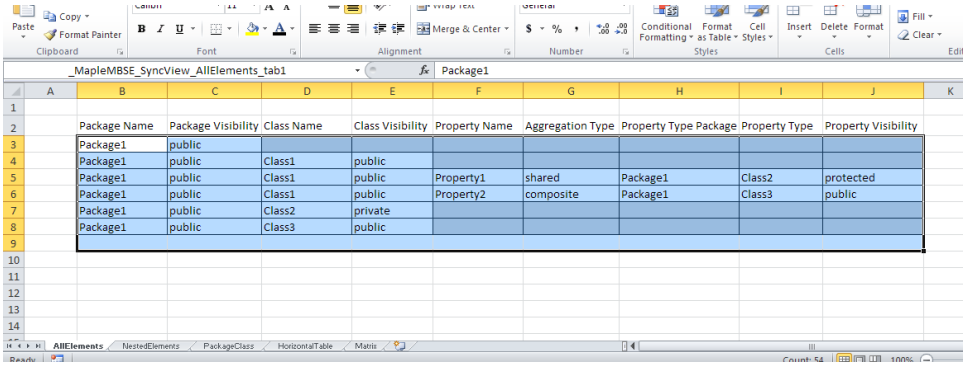


Figure 2.5: SyncView Areas Displayed in Name Box

Selecting one of the names shows the corresponding range of the cells.



	Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Aggregation Type	Property Type Package	Property Type	Property Visibility
3	Package1	public							
4	Package1	public	Class1	public					
5	Package1	public	Class1	public	Property1	shared	Package1	Class2	protected
6	Package1	public	Class1	public	Property2	composite	Package1	Class3	public
7	Package1	public	Class2	private					
8	Package1	public	Class3	public					

Figure 2.6: Highlighting SyncView Cell Range

In the remainder of this section we give an overview of the Excel operations on the spreadsheets.

Allowed Operations

The given table assumes that the SyncView is a vertical table. For a horizontal table the operations apply to columns instead of rows.

Editing Operation	Description
Insert a new row into the SyncView table.	Use the Excel context menu to insert a new row anywhere in the table.
Insert a new row by typing into a cell directly below the last row of the table.	Type content into the cell directly below (and adjacent to) the table and press Enter. This will add a new row to the table and increase the SyncView area.
Insert copied cells as new rows.	Insert rows in the table area by inserting or pasting copied cells (using Insert Copied Cells feature from Excel context menu).
Update cell content in an existing row of the table.	Update table cell contents within the SyncView area
Delete rows	Delete table rows within the SyncView area.
Sort rows in SyncView	Select then sort the entire table in order to replace rows in the table.

Editing Operations That Will Result in a MapleMBSE Error Message

The editing operations listed in the table below disrupt the structure of the SyncView table, as defined in the MapleMBSE configuration file and will result in an error message being generated and the editing operation being undone:

- Inserting a new row at the top of the spreadsheet, above the SyncView table
- Inserting a column in the spreadsheet to the left of the leftmost column of the table
- Updating across several non-adjacent rectangular areas. That is, copying and attempting to paste cell content into multiple selected areas in the table.
- Deleting rows or columns in the insertion area
- Moving the SyncView area. Note that you must edit the SyncView table schema in the configuration file (.mse) to move the SyncView area.

Operations That Could Negatively Change the Mapping between MapleMBSE and the Model Element

When making edits to the rows and columns in a table, the following operations do not alter the SyncView table layout as defined in the MapleMBSE configuration file. However, if you perform any of these operations there is a risk of breaking the mapping between the rows of the table in the model spreadsheet and the model element itself. This could result in unexpected behavior afterwards.

Editing Operation	Description
Sorting only a portion of the table in the SyncView area.	Sorting only selected rows, cells or columns in the SyncView area
Inserting content into the table and then shifting cells up.	Inserting content into the table (for example by right-clicking on a table cell, then selecting Insert), then selecting either: Shift Cells Up or Shift Cells Down.
Deleting content from selected table cells.	Similar to inserting content into the table, and then shifting cells up or down.

2.3 Adding Model Elements

Adding elements is done by adding records to the SyncView areas. How the records are added depends on the SyncView layout. For a vertical table, adding records corresponds to adding rows and for the horizontal table, to adding columns. For simplicity, in the following instructions, assume the SyncView is a vertical table. Adding records to a vertical table can be done in the following two ways.

Tip: You can follow the steps in this example using the *UserGuide.MSE* configuration file, *AddingNewElements.uml* model and the *Packages* spreadsheet. These files can be found in the **<MapleMBSE>/Example/UserGuide** directory, where <**MapleMBSE**> is your MapleMBSE installation directory.

1. Type the necessary information in the insertion area.
   In **Figure 2.7**, the SyncView is outlined in red and the insertion area in green (cell B4 in a). Type Package2 in the insertion area (b) and press enter. MapleMSBE checks that no other element with this name exists and adds a new package element with the name Package2 (c). The model is shown in the bottom of the Figure. The SyncView area is grown by one row. The insertion area is the cell B5.

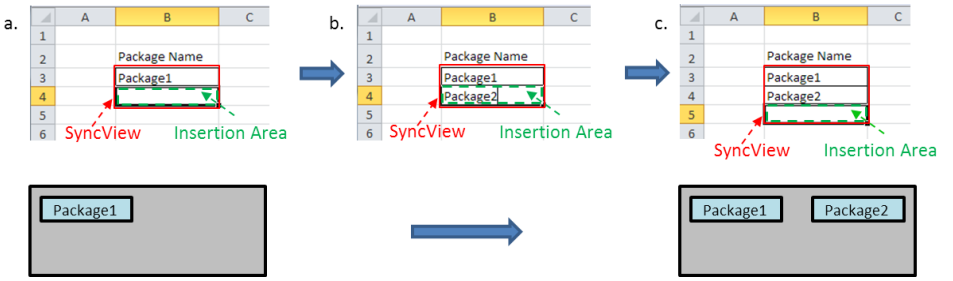


Figure 2.7: Entering Information in Insertion Area

2. Insert a row in the SyncView area and enter the necessary information.
   In **Figure 2.8** the SyncView area is outlined in red (a). For example, select row 4 and in the context menu choose **Insert** (b). A row inserted in place of row 4, and the previous row is shifted below (c). The SyncView area is increased by one row. Enter the name of the new package - Package3. MapleMBSE creates a new element with the name Package3.

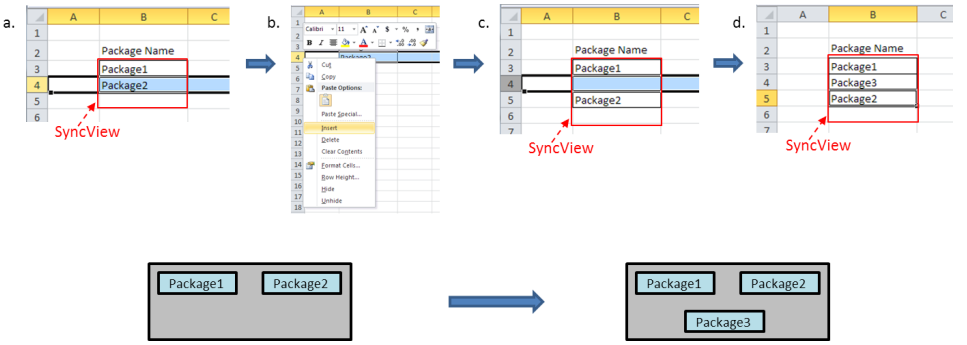


Figure 2.8: Inserting a Row in the SyncArea

The changes are performed on the model in the internal MapleMBSE memory. To make your model changes persist, save the workbook. For details, see *Saving Model (page 8)*.

MapleMBSE does not sort new elements automatically. The sorting can be done manually, for details see *Sorting Records (page 32)*. Adding records will also be reflected in other spreadsheets that refer to the same model elements. In other worksheets, additional records are always added to the last line.

Nested Elements

A model usually has a lot of dependencies which are represented by a tree structure. To reflect that structure in a table a staircase table is used. Consider the example in <Hyperlink Target="UserGuideNestedElements" Type="FigureRef" Filename="Chapter02">Chapter02#UserGuideNestedElements</MHyperlink>. Assume we want to define a structure as on the left-hand side of the Figure: Package1 contains Class1, which contains Property1. Each element is represented by one record in the table. We define each element in turn, starting with Package1.

Tip: You can follow this example using the: UserGuide.MSE configuration file, AddingNewElements.uml model and the *NestedElements* spreadsheet. These files can be found in the **<MapleMBSE>/Example/UserGuide** directory, where <**MapleMBSE>** is your MapleMBSE installation directory.

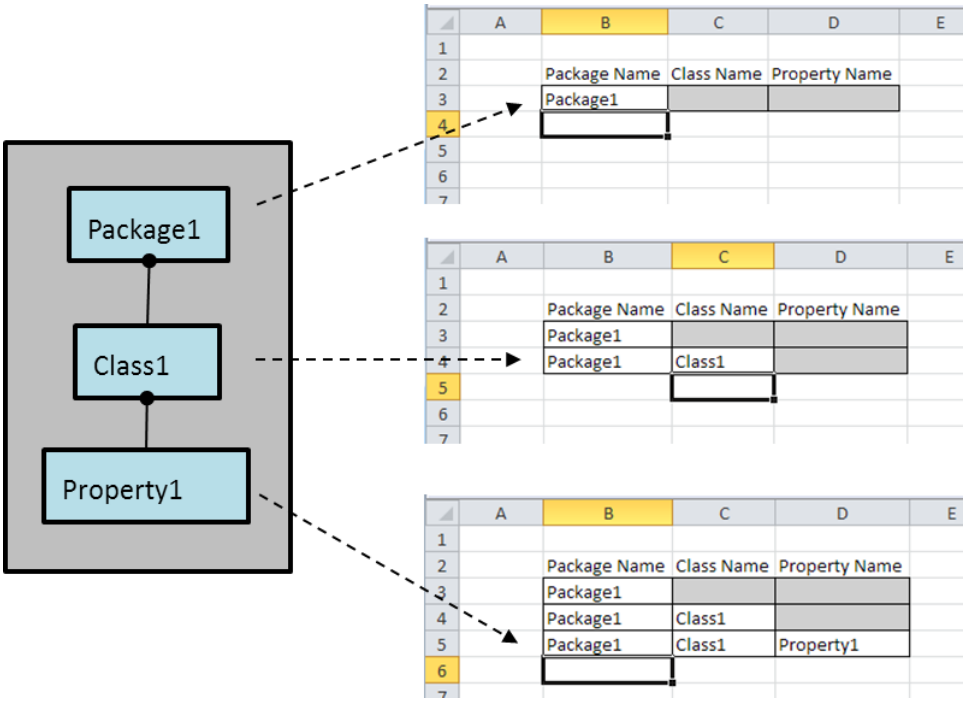


Figure 2.9: Defining a Nested Structure in the NestedElements spreadsheet

- *Define Package1*
  The steps for defining Package1 are shown in **Figure 2.10** . In the insertion area in the column corresponding to the package definitions (a) type Package1 (b) and press Enter. When you press Enter, MapleMBSE checks whether there is an element in the in-memory model with that name already. It finds that such element does not exist yet, so the row represents a new element. Since one row corresponds to one element we cannot define any additional elements inside it in this row. MapleMBSE indicates that by showing fields corresponding to the nested elements in gray.

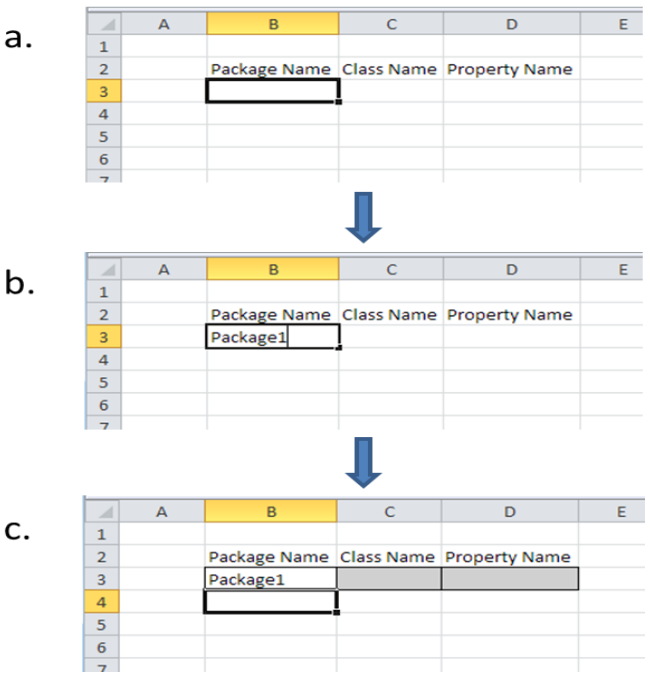


Figure 2.10: Defining Package1

- *Define Class1*
  The steps for defining Class1 in Package1 are shown in **Figure 2.11**. To define Class1, you need to first specify the package it belongs to. Type Package1 in the column corresponding to packages and press the right arrow key (a). MapleMBSE checks whether there is an element with this name already in the model. It finds that the element exists and shows the "Duplicated key" warning for this field. Other fields are still white so you can now define nested elements inside Package1. Enter Class1 (b) and press Enter. Again MapleMBSE verifies if Class1 already exists inside Package1, finds that it does not and creates a new element. The warning from the package field is cleared (c). Since one record

can correspond only to one element and this record now defines Class1, the same row cannot be used to define any elements inside Class1. MapleMBSE indicates this by showing the Property Name column in gray.

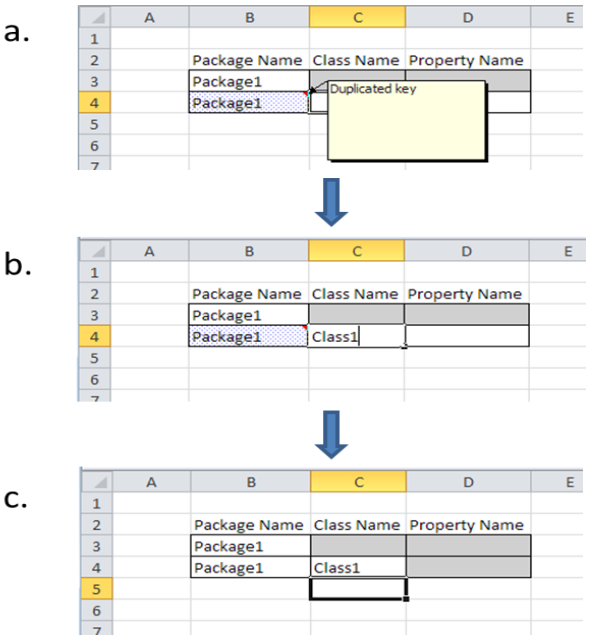


Figure 2.11: Defining Class1

- *Define Property1*
  The steps for defining Property1 are shown in <MHyperlink Target=". To define Property1, you need to specify the package and the class it belongs to. Type "Package1" in the column corresponding to packages and press the right arrow key (a). As in the case above, MapleMBSE identifies that the element, Package1 already exists and adds a warning mark to that cell. Type "Class1" and press the right arrow key (b). This time, Class1 already exists in Package1 so MapleMBSE gives the "Duplicated key" warning for that cell as well. The property field is white and you can now define an element representing a property inside Class1. Type Property1 (c) and press Enter. MapleMBSE verifies that there is no existing property inside Class1 called Property1 and the record represents a new unique element. All warnings from the other fields are cleared (d).

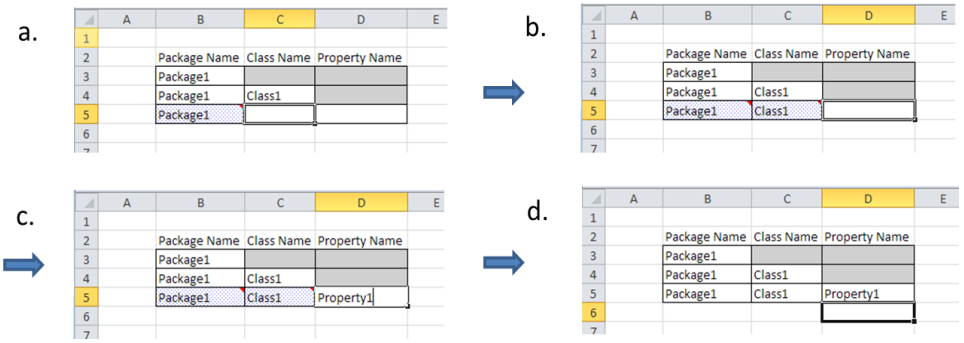


Figure 2.12: Defining Property1

Alternative Groups

In a model, different types of elements may belong to the same parent. For example, a package may contain classes or other packages. An MSE configuration specifies what type the record must have thus specifying the type of the element it defines. In the example in *Nested Elements (page 24)*, the MSE configuration allowed for 3 types of records: packages defined at the top level, classes defined inside top-level packages, properties defined inside classes. It did not allow for defining other packages inside top-level packages or classes inside other classes, for example. To be able to define different types of elements at the same nested level, alternative groups must be defined in the MSE configuration, see *MapleMBSE Configuration Language Guide* for details. Here we consider when such a configuration is used.

Tip: You can follow the steps in this example using the: UserGuide.MSE configuration file, AddingNewElements.uml model and the *AlternativeGroups* spreadsheet. These files can be found in the **<MapleMBSE>/Example/UserGuide** directory, where <**MapleMBSE**> is your MapleMBSE installation directory.

The MSE configuration in this example allows us to define the following 4 types of elements by defining the corresponding fields in the SyncView.

- Top level packages - **Package Name**
- Classes inside top level packages - **Class Name**
- Packages nested inside the top level packages - **Nested Package Name**
- Classes inside nested packages - **Nested Class Name**

	A	B	C	D	E	F
1						
2		Package Name	Class Name	Nested Package Name	Nested Class Name	
3						
4						
5						
6						
7						
8						

The model we are going to define in this example is shown in **Figure 2.13**. A top-level package (Package1) contains two elements: one of class type (Class1) and one of a package type (Package2). The nested package (Package2) has one class defined in it (Class2).

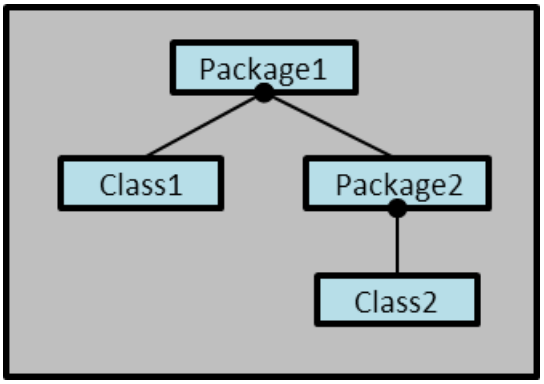


Figure 2.13: Alternative Groups Example Model

- *Define Package1*
  The process of defining Package1 is the same as in the examples in *Adding Model Elements (page 22)* and *Nested Elements (page 24)*. Enter Package1 in the Package Name column in the insertion area and press Enter.

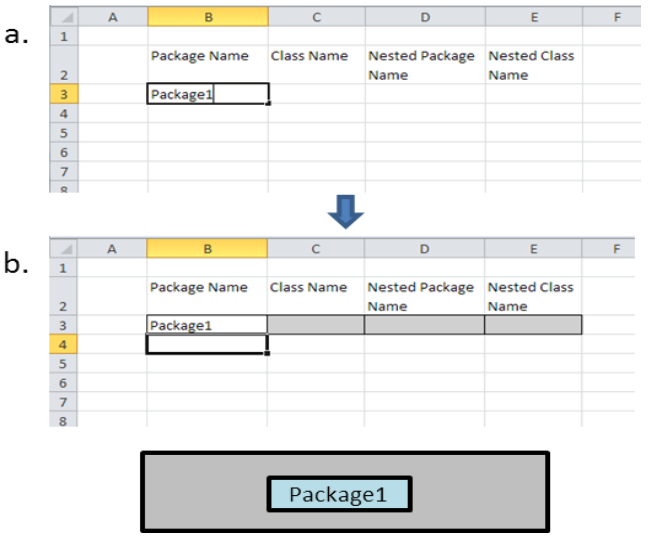


Figure 2.14: Defining Package1

- *Define Class1*
  To define Class1, follow the steps in **Figure 2.15**. Type Package1 in the Package Name column on the new line (row 4) and press Enter (a). As in the examples in *Nested Elements (page 24)*, *Define Class1*, a warning is added to the cell defining Package1 indicating that it is a duplicated key. Unlike in that example, the background of the other cells in this row is changed to dark gray. This color is different from the lighter gray used to indicate that no more elements can be defined in this row (row 3). The dark gray color is used to indicate the fields represent alternative fields. Type Class1 in the **Class Name** (b) and press Enter. A new class called Class1 is added to Package1 (c). The background of the remaining cells in this row in the SyncView area is changed to light gray indicating that no more elements can be defined in this row.

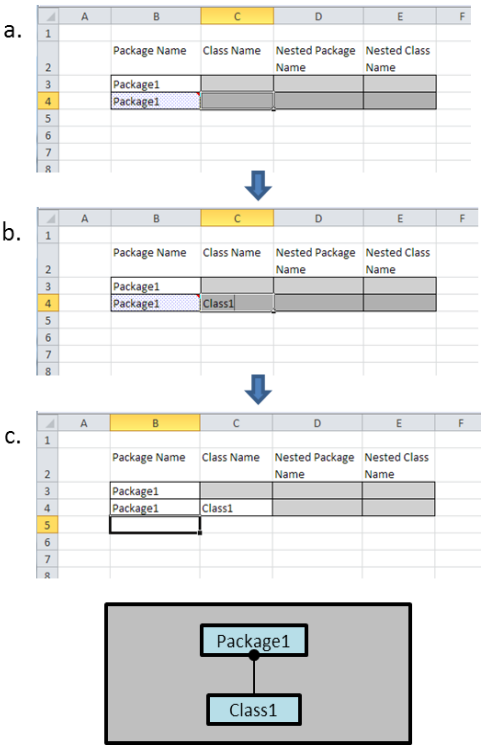


Figure 2.15: Defining Class1

- *Define Package2*
  To define Package2 inside Package1, follow the steps in **Figure 2.16**. First, type Package1 on the new line (row 5) and press Enter (a). Again, a duplicated key warning is added to the cell with Package1 and the background of other cells in this row inside the SyncView area is changed to dark gray indicating the rest of the cells represent alternative groups. Leave the **Class Name** column empty and type Package2 in the **Nested Package Name** column (b). Press Enter (c). A new package Package2 is added to the model. The background of the cells representing nested elements inside Package2 (**Nested Class Name**) is changed to light gray indicating that this record cannot be used to define nested classes inside Package2. The background of the cell representing the alternative (**Class Name**) remains dark gray.

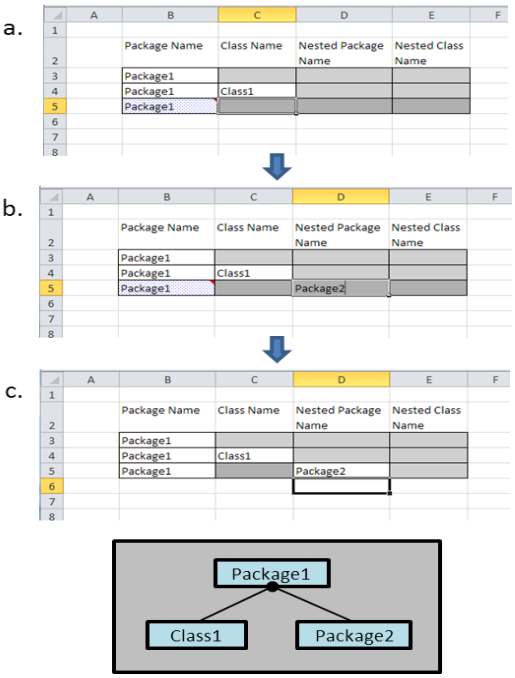


Figure 2.16: Defining Package2

- *Define Class2*
  The steps defining Class2 inside Package2 are shown in the Figure below. Type Package1 in the **Package Name** column and press Enter (a). Type Package2 in the **Nested Package Name** (b) and press Enter (c). Now a package called Package2 inside Package1 already exists and a duplicated key warning is added to the cell with Package2. The background of the **Nested Class Name** cell remains dark gray. Type Class2 in the **Nested Class Name** cell (d) and press Enter (e). A class called Class2 is added to Package2 in the model. The warnings are cleared. The background of the **Class Name** cell remains dark gray indicating that it represents an alternative type of elements in Package1. Any valued entered there will be ignored and cleared after Enter is pressed.

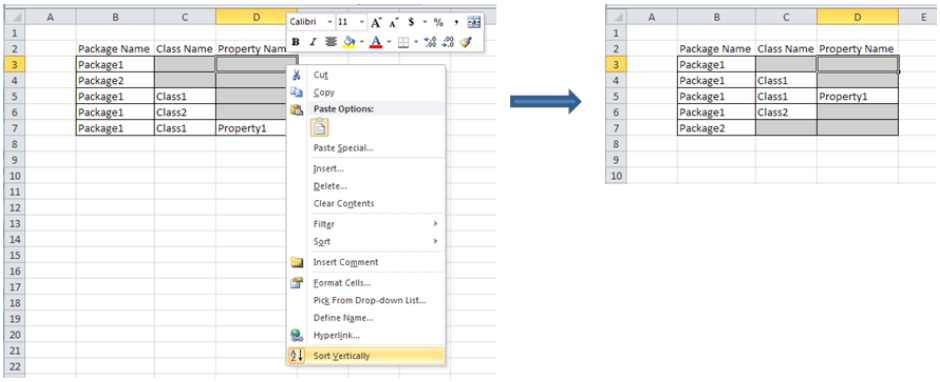
Figure 2.17: Defining Class2

2.4 Sorting Records

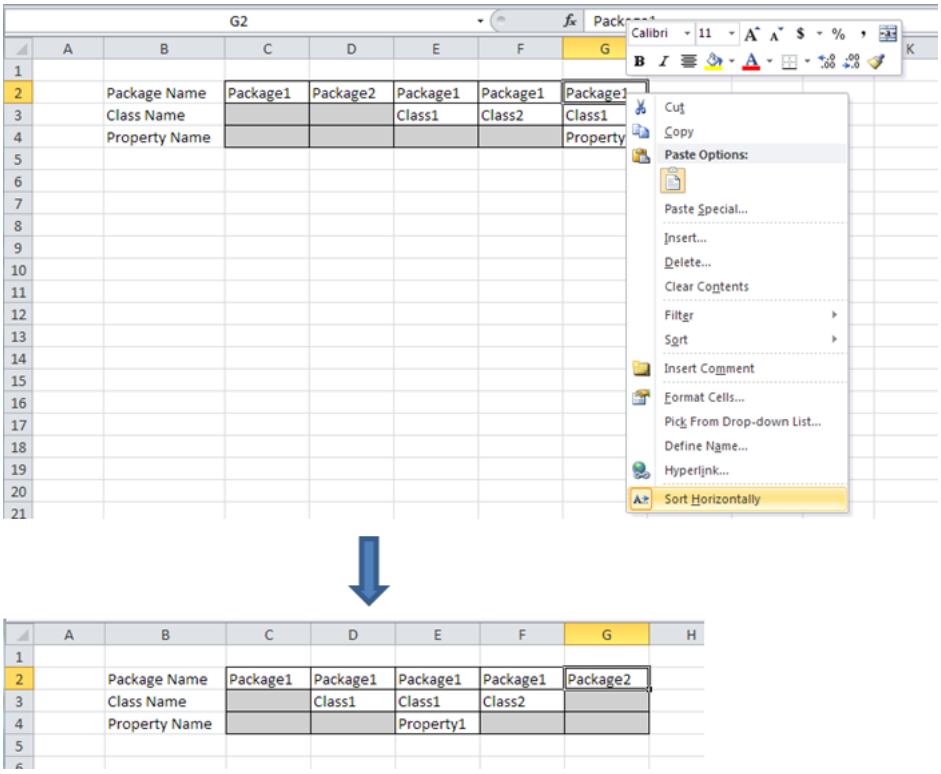
The order for sorting is specified in the MSE configuration file. When a model is opened all records are sorted according to the these specifications. When new records are added to the tables they are not sorted automatically. You can sort the records manually. Sorting records has no effect on the model.

- **Vertical Table**
  To sort the records manually in a vertical table, right-click anywhere in the SyncView

area, and then select **Sort Vertically**.

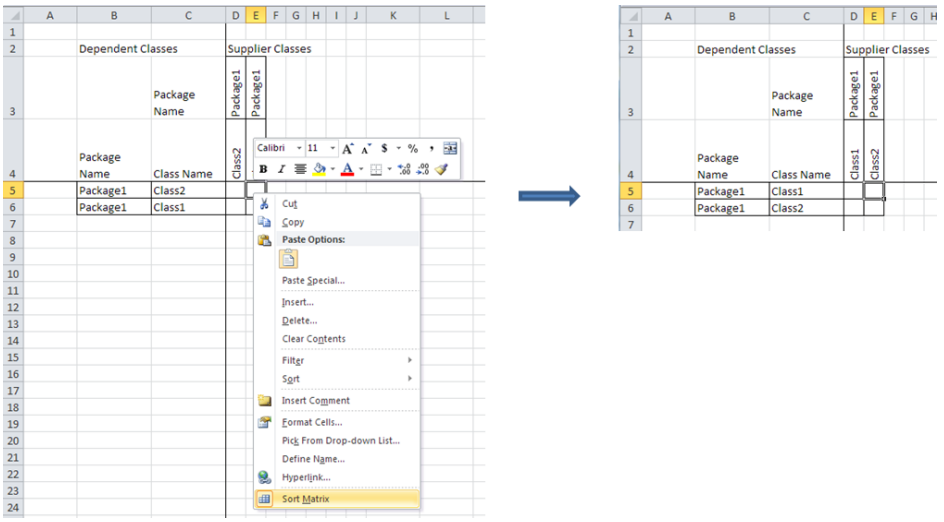


- **Horizontal Table**
  To sort the records in a horizontal table, right-click anywhere in the SyncView area, and then select **Sort Horizontally**.



- **Matrix**
  To sort the records in a matrix, right-click anywhere in the matrix SyncView area, and

then select **Sort Matrix**.



2.5 Deleting Model Elements

Deleting elements is done by deleting the corresponding record in the SyncView area. In a vertical table it corresponds to deleting a row, in a horizontal table - a column. In a matrix, deleting a row or a column also deletes the row or the column in the corresponding vertical or horizontal table, correspondingly, and, thus, the element represented by that row or column. In **Figure 2.18**, the SyncView is a vertical table (a). To delete Class1 in Package1, you need to delete the row representing this element - row 4 (b). The element is deleted from the table and the model (c).

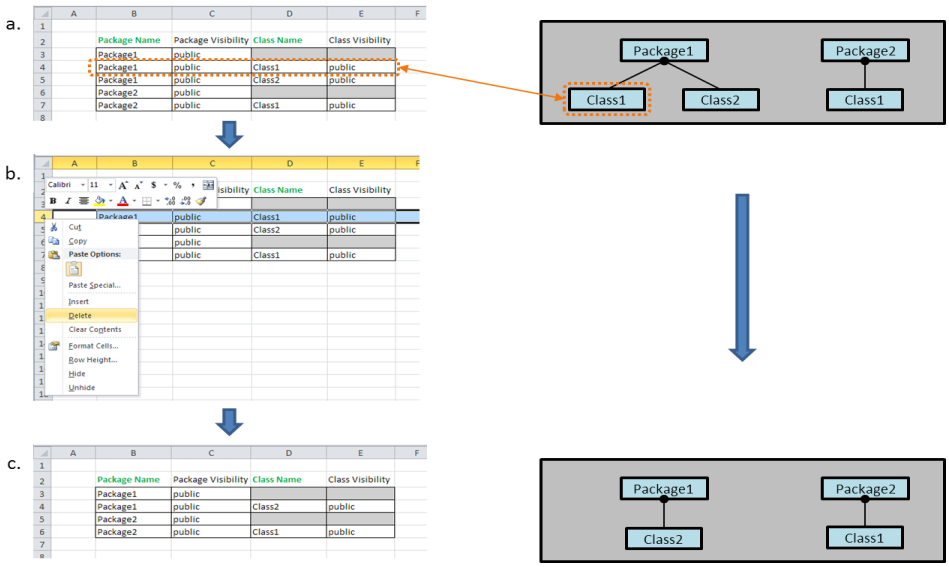


Figure 2.18: Deleting the Class1 Element from the Model

Deleting an element also deletes all elements belonging to it. For example, deleting Package1 in **Figure 2.19** (a) also deletes Class1 and Class2 belonging to it.

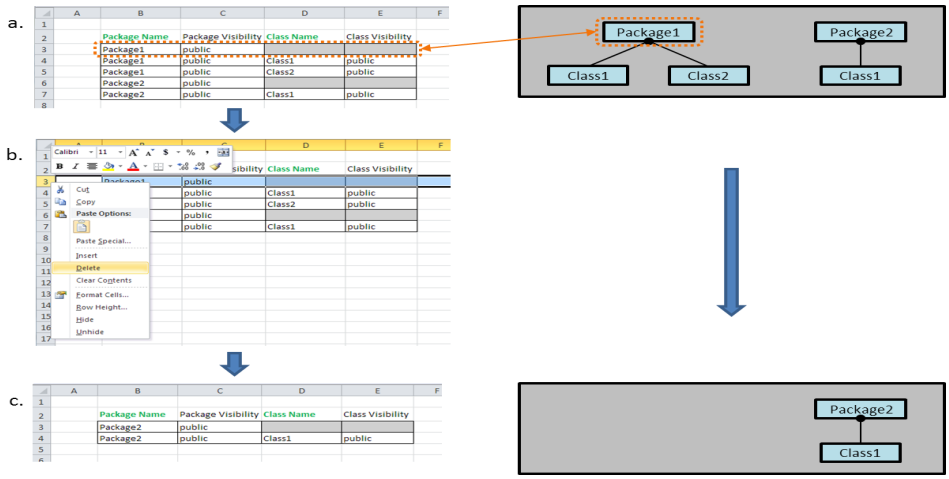


Figure 2.19: Deleting Package1 and Associated Elements

Deleting Records with References

Reference fields in a record point to other elements in the model, see *Field Types (page 36)* for details. When deleting a record that contains reference fields, the corresponding referenced element is not deleted.

2.6 Modifying Model Elements

In the examples we considered in *Adding Model Elements (page 22)* and *Sorting Records (page 32)*, the SyncViews were used to add new elements but they did not allow to edit or view any properties of the elements. In this section we define other characteristics of SyncViews that allow us to work with different properties of the elements.

As discussed in *Operations Overview (page 19)*, model elements are represented in a SyncView by records, and each record consists of one or more fields. The fields represent different properties of the element. By editing the fields in the Excel spreadsheet you modify the corresponding properties of the model elements. What effect editing a field has on the model element depends on the type of the field. The types of the fields are set by the MSE configuration. Here we describe the different types of fields and what effect they have on editing operations.

Field Types

There are three main types of fields in SyncView:

- An attribute field;
- A reference field;
- An unmapped field.

An attribute field points to the value of a property with a primitive type, such as a string or a number. A reference field represents a property with a nonprimitive type, such as another element in the model. An unmapped field is a field that is not mapped to any property of the element. It has no influence on the model but may be useful for comments or performing Excel operations related to other fields. See *Unmapped Fields (page 55)* for details.

In addition, fields can be either **key** or **nonkey** fields. The **key** fields are used to uniquely identify a model element. Therefore, multiple records with the same key fields cannot exist in the same SyncView. One or more key fields exist in one record. Whether a field is key or not is set in the MSE configuration. The **nonkey** fields simply represent the properties of the element defined by the key fields.

Note: There is a distinction between **record key** fields and **reference key** fields. If there are reference fields in a record some of them must be reference key fields that determine which element the reference points to. The record key fields are used to determine the record uniquely. The reference key fields may also be the record key fields.

In **Figure 2.20**, the SyncView is a vertical table. Each row is a record, and different fields are represented by different columns. The record key fields are outlined in green: Package Name, Class Name, and Property Name. They are attribute fields. The record nonkey attribute fields are outlined in blue: Package Visibility, Class Visibility, Property Visibility, and Aggregation Type. Both key and nonkey attribute fields are strings. A property in a class must have a type which is defined by some other class. In other words, a type of a property is another element in a model. So, to represent a type of a property, reference fields must be used. To define a class uniquely in the model, we need to specify the name of the package it belongs to and its name. The fields Property Type Package and Property Type, outlined in orange, are reference key fields. In this example, reference key fields are not record key fields (they are not required to determine the record itself, only the element it refers to). Providing Property Type Package and Property Type is enough information to define the type of a property. However, you might be interested in viewing some information about the type in the same record - for example, the visibility of the type. For that purpose a nonkey reference field is used. The field Type Visibility, outlined in purple, is a nonkey reference field representing the visibility of the class used as a type for the property.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1	public	Class1	public	Property1	protected	shared	Package1	Class2	private
Package1	public	Class1	public	Property2	public	composite	Package1	Class3	public
Package1	public	Class2	private						
Package1	public	Class3	public						

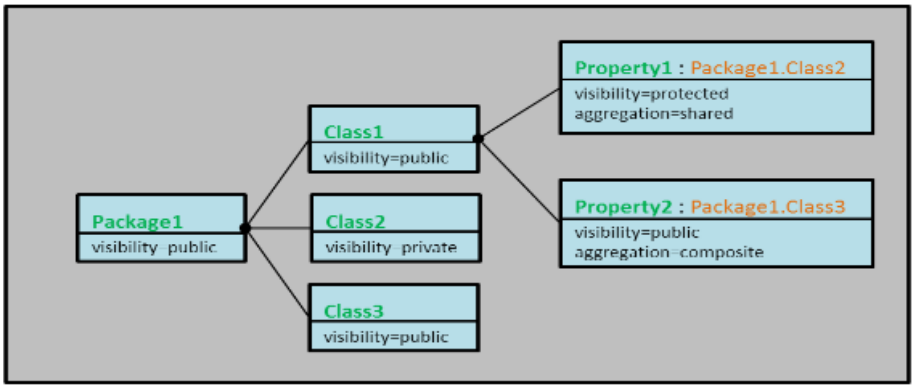


Figure 2.20: Illustration of Main Types of Fields in SyncView

In the examples we considered in *Adding Model Elements (page 22)* and *Sorting Records (page 32)*,the records of the SyncViews consisted only of key attribute fields. In the example in *Deleting Model Elements (page 34)*, the SyncView had record key attribute

fields (Package Name and Class Name) and nonkey attribute fields (Package Visibility and Class Visibility).

Record State

Another definition important for working with records in SyncViews is the the state of the record. Each record has two states: **bound** and **unbound**, depending on whether it is associated with the model loaded in memory. When a new record is first created it is in an unbound state - it is not associated with the model. Once all record key fields are filled and it is verified that together they represent a unique element the record changes the state to bound - the record is associated with the model in memory. If it represents an existing element, the nonkey fields will be populated with the information from the in-memory model. If the element does not exist it is created in the model. Depending on the modeling tool some of the attributes are given default values to new elements. If any of those attributes are mapped to nonkey fields, the fields will be populated with those values.

Example of Using Different Fields When Adding Elements

As an example of adding elements to a SyncView with several different types of fields we create the model in **Figure 2.21**. The MSE configuration for the SyncView specifies the types of the fields as in the example in *Field Types (page 36)*.

Tip: You can follow the steps in this example using the: UserGuide.MSE configuration file, AddingNewElements.uml model and the *AllElements* spreadsheet. These files can be found in the **<MapleMBSE>/Example/UserGuide** directory, where <**MapleMBSE>** is your MapleMBSE installation directory.

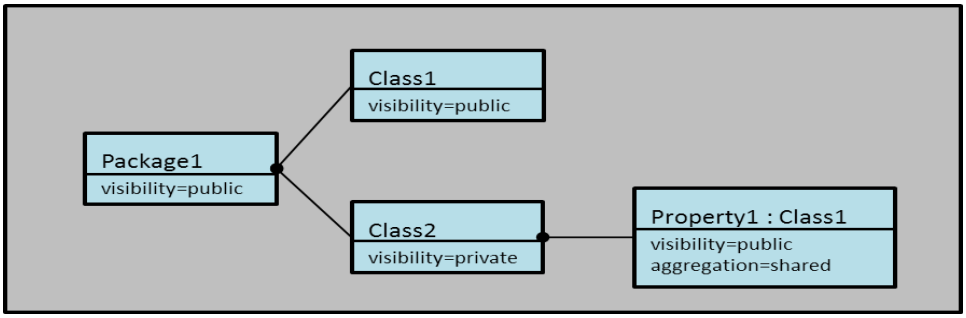


Figure 2.21: Adding Elements to a SyncView with Different Types of Fields

- *Define Package1*
  The steps defining Package1 are shown in **Figure 2.22**. The names of the columns corresponding to the record key fields are shown in green. Type Package1 in the Package Name column (a). The record is unbound. Press Enter. MapleMBSE verifies that the record defines a unique element. The record's status is changed to bound. MapleMBSE creates

a new package in the model with the name Package1. By default, new elements in UML have visibility assigned to public. Since the record is now bound, the value of the Package Visibility field is populated automatically (b).

a.

	Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
3	Package1									

b.

	Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
3	Package1	public								

Figure 2.22: Defining Package1

- *Define Class1*
  The steps defining Class1 in Package1 are shown in **Figure 2.23**. To define Class1, you only need to fill out the key fields: Package Name and Class Name (a). When you enter Package1 in the Package Name column, MapleMBSE checks that an element with this name already exists, so the status of the record remains unbound. The Package Visibility remains unpopulated. Enter Class1 in the Class Name column and press Enter. MapleMBSE verifies that the record now represents a unique element and binds the record to the model. It creates a new class inside Package1 called Class1. The visibility is set to public by default. The nonkey fields for Package and Class Visibility are populated (b).

a.

	Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
3	Package1	public								
4	Package1		Class1							

b.

	Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
3	Package1	public								
4	Package1	public	Class1	public						

Figure 2.23: Defining Class1

- *Define Class2*
  The first steps defining Class2 in Package1 are the same as defining Class1 (a and b in **Figure 2.24**). The last step is to change the default value for the visibility of Class2 to private (c).

a.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1		Class2							

b.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1	public	Class2	public						

c.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1	public	Class2	private						

Figure 2.24: Defining Class2

- *Define Property1*
  The steps defining Property1 in Class2 are shown in **Figure 2.25** and **Figure 2.26**. First, only key fields need to be entered. When you enter the package and the class name the record remains unbound because a package and a class with those names already exist. Nonkey fields remain unpopulated. Type Property1 in the Property Name column (a) and press Enter. Now the record represents a new unique element, so MapleMBSE binds it to the model and creates a new element. The nonkey fields are populated (b). The values for the visibilities of the existing elements (Package1 and Class2) are taken from the values defined for these elements (public and private, respectively). Property1 is assigned the default visibility - public. The Property Type Package and the Property Type columns do not have default values. They are defined (in the corresponding configuration file) as references to existing classes. Since the fields are empty, they do not refer anywhere, and the warning "No reference target" is shown for the Property Type Package (c) and for the Property Type fields.

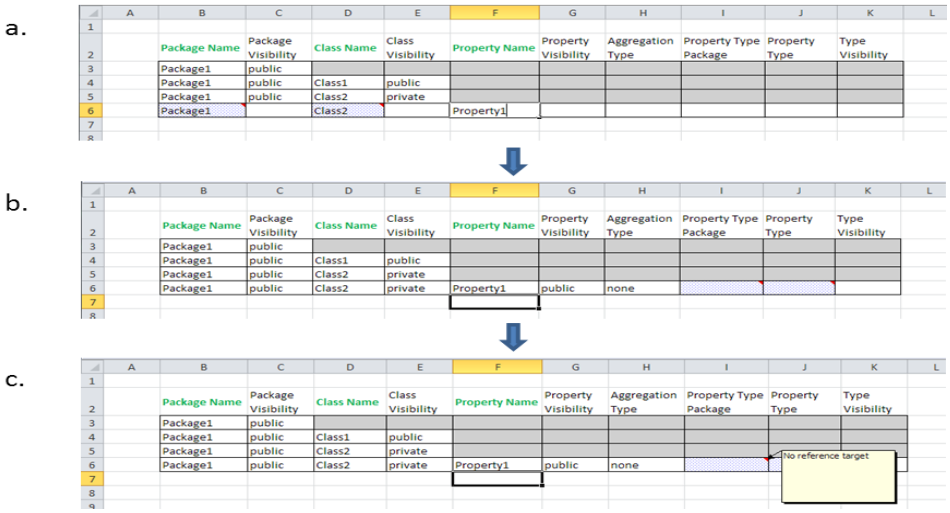


Figure 2.25: Creating New Bound Records

To define the type, you need to specify the full name of the class that property instantiates. The full name includes the name of the package the class belongs to and the name of the class itself (d). Note that just specifying the package name does not resolve the class reference, and so the warnings are not cleared. Only after entering the name of the class and pressing Enter the reference is resolved and the warnings are cleared (e).

Note: The reference target must exist in the model already. This line cannot be used to create the referenced element.

Finally, change the Aggregation Type from its default value (none) to shared (f).

d.

e.

f.

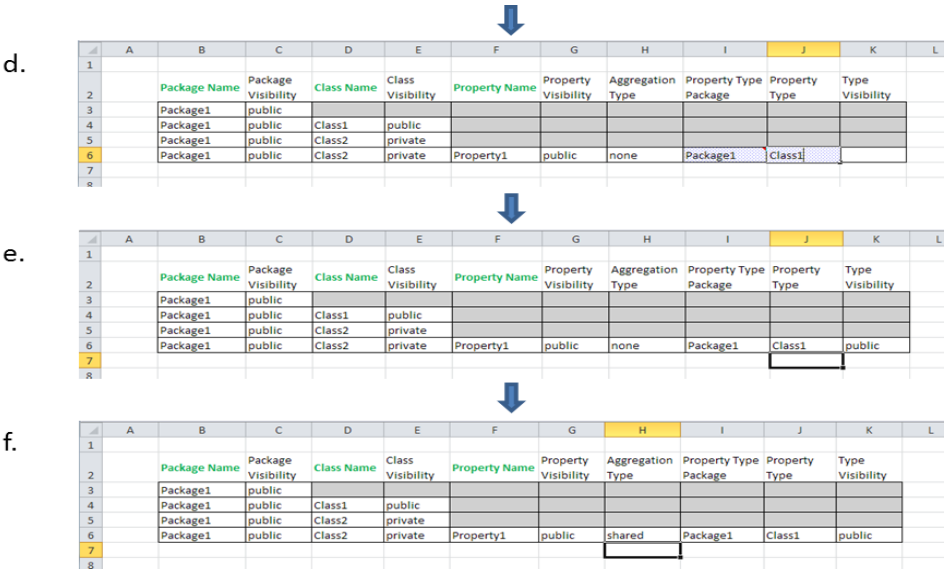


Figure 2.26: Defining the Type

Modifying Attribute Fields in Unbound Records

For definitions of bound and unbound records see *Record State (page 38)*.

- *Modifying key attribute field*
  When a new record is defined before it is bound to a model, entering and modifying values in the key fields specifies the new model element the record represents. This is what we have been doing in the examples of adding elements.
- *Modifying nonkey attribute fields*
  If any values are entered into the nonkey fields when the record is unbound, the result depends on whether the field represents an existing element or the new element that the record defines.

- *For existing elements*
  The entered value will be ignored and overwritten with the existing value of the element. In the example in **Figure 2.27**, when adding Class1 inside Package1, Package1 already exists and has public visibility. During the definition of the Class1 the visibility of Package1 is set to private while the record is unbound (a). When the input of the class name is finished and Enter is pressed, the record becomes bound, and the visibility of Package1 is set to the value defined in the model - public.

a.

	A	B	C	D	E	F
1						
2		Package Name	Package Visibility	Class Name	Class Visibility	
3		Package1	public			
4		Package1	private	Class1		
5						
6						

b.

	A		C	D	E	F
1						
2		Package Name	Package Visibility	Class Name	Class Visibility	
3		Package1	public			
4		Package1	public	Class1	public	
5						
6						

Figure 2.27: Modifying Existing Elements in Unbound Records Results in the Manually Entered Values Being Overwritten

- *For new elements*
  The entered value will be assigned to the corresponding property of the new element when the element is created. In the example in **Figure 2.28**, when adding Class1, the visibility for it is entered before all key fields define the element uniquely and the record is unbound (a). The key field for the name of the class is field (b) and entered. The record becomes bound to the model. The visibility of Package1 is determined from the definition of Package1. A new class, Class1 is created in Package1 and its visibility is set to private.

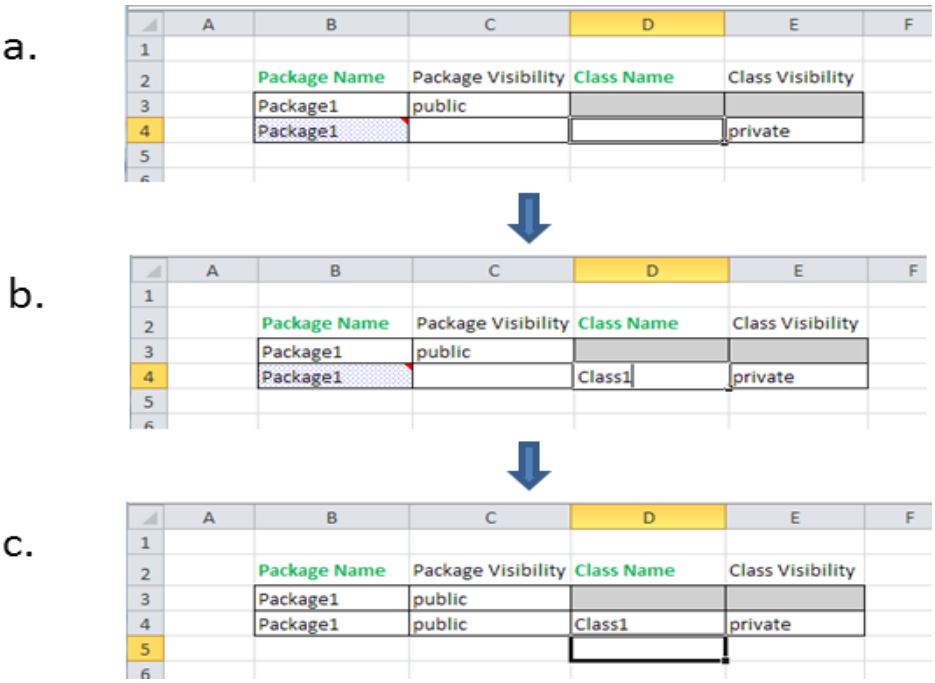


Figure 2.28: Adding New Elements to a Model

Modifying Reference Fields in Unbound Records

For definitions of bound and unbound records see *Record State (page 38)*.

- *Modifying reference key field*
  When a new record is being added, the reference key fields must be set to an existing element. The reference fields cannot be used to create a new element. If there are multiple keys that define the reference target it is necessary to define all of them and all of them should be pointing to elements that already exist in the model. See step *Define Property1* in *Example of Using Different Fields When Adding Elements (page 38)*,
- *Modifying reference nonkey field*
  If the values are entered into nonkey reference fields in an unbound record the result depends on whether the reference keys were defined.

- *Reference keys were not defined*
  *W*hen no key reference fields are defined, the values entered in the nonkey reference fields of an unbound record (a, outlined in purple) are ignored and cleared when the record becomes bound (b).

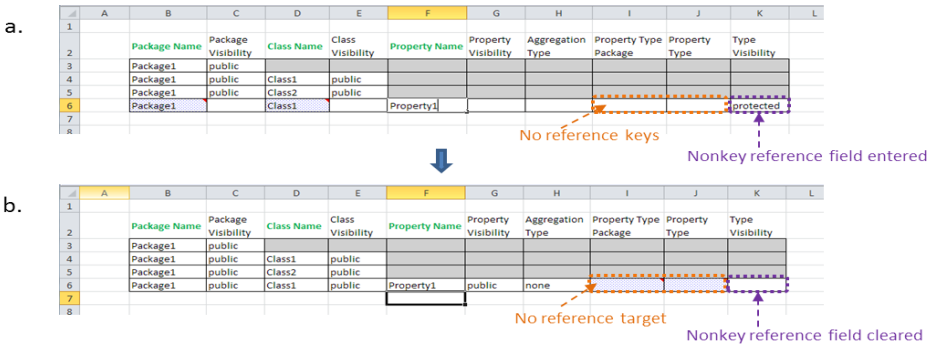


Figure 2.29: Values Entered in Nonkey Reference Fields and No Key Reference Fields Defined

- *Reference keys were defined*
  When reference keys are defined and refer to an existing record (a, outlined in orange), the values entered into the nonkey reference fields of an unbound record (outlined in purple in a.) are applied to the element that is being referenced when the record becomes bound. Other fields and SyncViews bound to the same property are also updated (b).

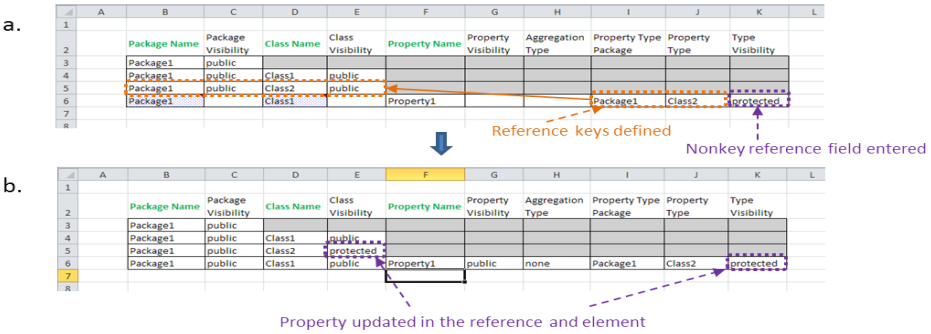


Figure 2.30: Values Entered in Nonkey Reference Fields with Key Reference Fields Defined

Modifying Attribute Fields in Bound Records

For definitions of bound and unbound records see *Record State (page 38)*.

A bound record is linked to an element in the model. Changing the value of an attribute field in a bound record modifies the corresponding property of the element. If there are other fields in other records or other SyncViews that show the same attribute of that element they will also be updated.

- *Modifying key attribute field*
  When a key attribute is changed, MapleMBSE first performs checks to identify if the new combination of the key fields for the record is unique. If it is, the new value is assigned to the property in the model and in the SyncViews.

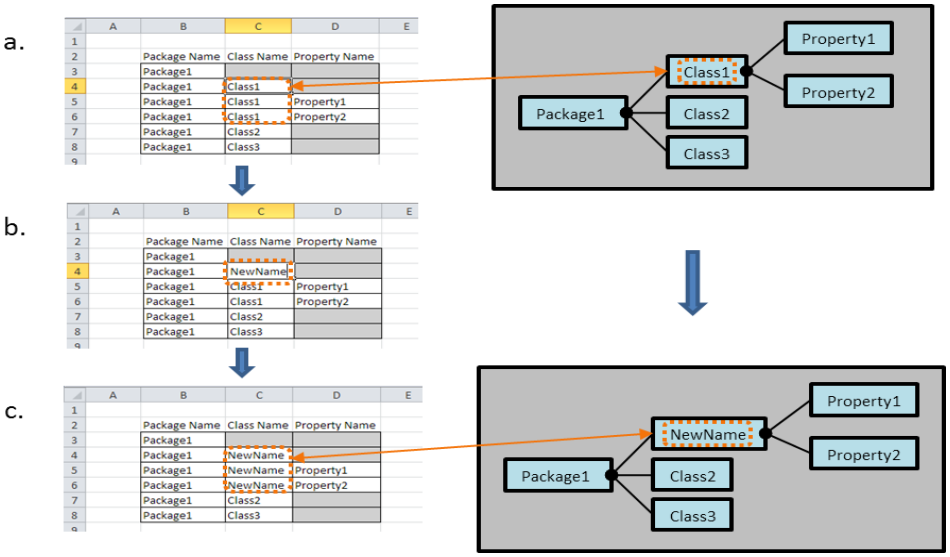


Figure 2.31: Modifying a Key Attribute Field

Note: MapleMBSE can check only that the combination of the keys for the record in which change is performed is unique. For example, if in the example above we changed Class1 to Class2 in the record defining Property1, row 5 (b), the record's key fields would be Package1-Class2-Property1. There is no element with such keys so the change would be applied. The change would then propagate to the model and Class1 would be renamed Class2 where Class2 already exists. MapleMBSE cannot detect this conflict. When the model is saved and opened in the corresponding tool, the tool will perform validation and detect the conflict. For this reason, it is recommended that if the key attribute needs to be changed it should be done in the record where the key is the final key field, like in the example above.

- *Modifying nonkey attribute field*
  When a nonkey attribute is changed, the change is applied to the model and is reflected in any fields and SyncViews bound to the changed property of the changed element. The modification can be done in the record that defines the element, like in the example below, or in other fields linked to this property. For example, the same change for Package1 as in the example below could be done in any of the cells outlined in orange in column C (a and c).

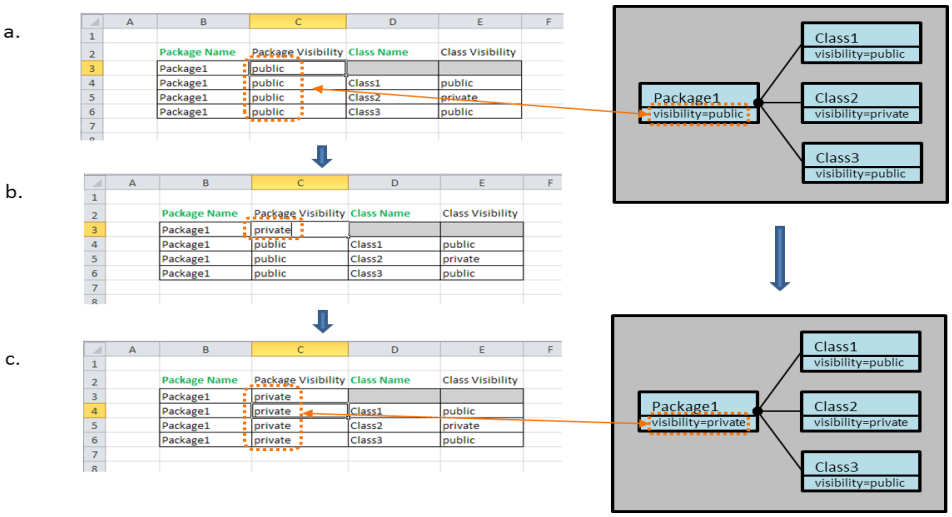


Figure 2.32: Modifying a Nonkey Attribute Field

- *Deleting value of key attribute*
  It is an error to delete the value of a key field attribute. Changes are not reflected in the model.

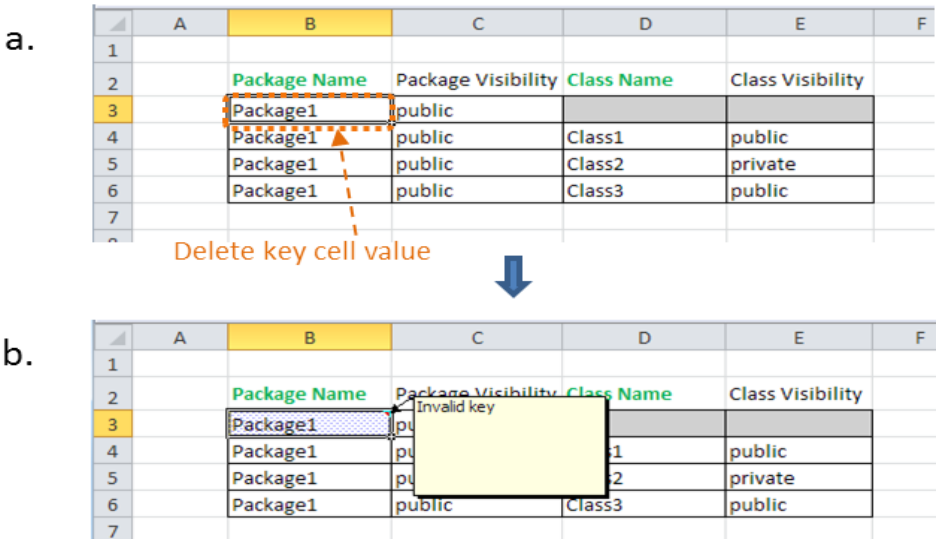


Figure 2.33: Deleting the Value of a Key Field Attribute

- *Deleting value of nonkey attribute*
  If you delete the value of a nonkey attribute cell the value will be cleared. However, depending on the model, the value may be reset to a default value.

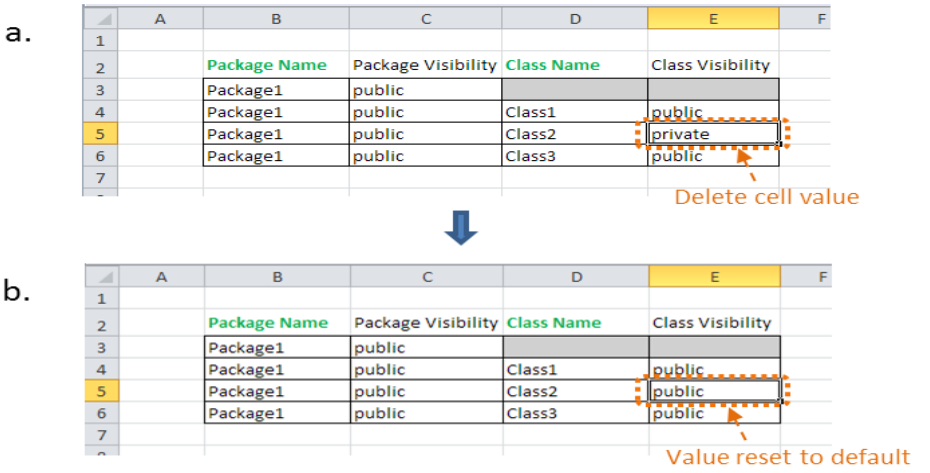
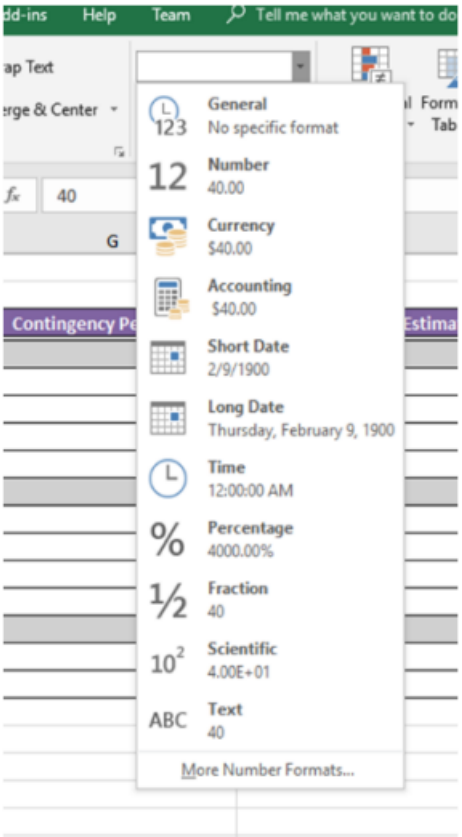


Figure 2.34: Deleting the Value of a Nonkey Field Attribute

Data type conversion

Excel cell values and model element values each have asssociateddata types. By default, Excel uses the General number format for worksheet cell values. The number format of a cell can be configured manually through the Format Cells menu. However, depending on the input value, there are cases where Excel will convert cell values to another format (for example, a date).



To avoid a mismatch of number types, MapleMBSE provides implicit data type conversion of cell values to match the values of the target model element.

- String conversion
  - When the target model element type is string, cell values are converted to string
- Enum conversion
  - When the target model element type is enum, string or integer is converted to enum. If it fails, "Invalid Value" error is annotated to the cell.
- EDataType conversion
  - When the target model element is of other EDataType and the cell value is text, try to convert it to the target EDataType. If it fails, "Invalid Value" error is annotated to the cell.
- Number conversion

- When the target model element is number and the cell value is also number, the cell value is automatically cast to fit with the target model element type. For example, if the model element is of type integer and the cell value is of type double, the cell value is truncated to an integer and set to the value of the model element.

Notes:

- Currently MapleMBSE supports Number, Text, and Date types
- You can put an apostrophe(') to quote a cell value to specify it is a text instead of setting cell format.
- While MapleMBSE does try to convert data types, there can be no guarantee that in all cases MapleMBSE will succeed. The best way to ensure the number type consistency is by manually formatting your Excel template.
- Some MapleMBSE configurations allow you to instantiate arrays of elements within a single cell. Each element must be separated with the delimiter specified by the mse configuration. See the following example.

In the example below (**Figure 2.35**) the column, **opaque expression body** (column D) accepts an array of type, String. To input this kind of data you must know: which cells accept this kind of input, the delimiter used (by default ; semi-colon) and the escape (by default \ back-slash).
To input the array:

```
["a", "b", "c", ";", "\"]
```

First, you need to prepend the escape string to any of the delimiter or escape symbols within your array of string:

```
["a", "b", "c", "\;", "\\"]
```

Next, you concatenate the string separated with the delimiter:

```
"a;b;c;\;;\\"
```

Now, do the same exercise backwards. To understand how to interpret a string as an array of string, use the example provided in cell D6:

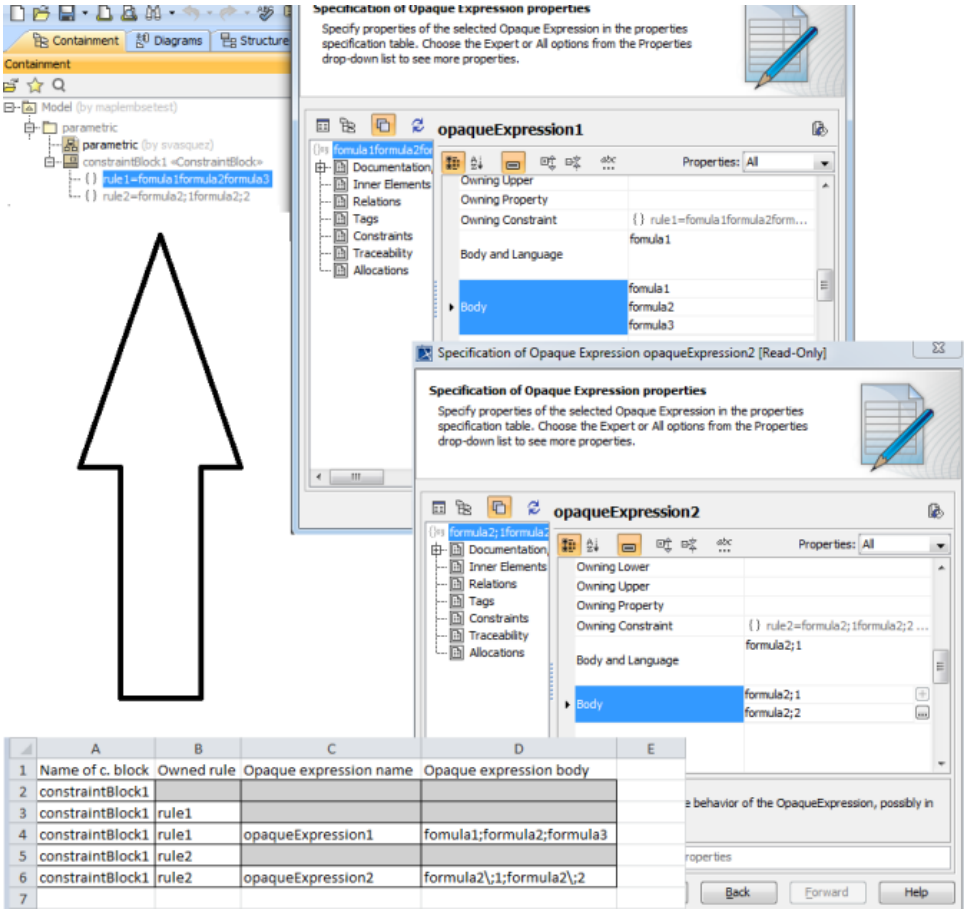
```
"formula2\;1;formula2\;2"
```

First, you need to split the string into several strings using the delimiter that are not escaped, in other words those that are not prepend by an odd number of escapes. Notice that `"2\;1"` is not a valid delimiter because it is escaped.

```
["formula2\;1"; "formula2\;2"]
```

Next, remove the escape string within each string in the array. Sometimes this can be tricky to do because `"\\"` should give us `" \"` meaning that the user intended to use the escape string and it is escaping the escape that was needed.

```
["formula2;1"; "formula2;2"].
```



	A	B	C	D	E
1	Name of c. block	Owned rule	Opaque expression name	Opaque expression body	
2	constraintBlock1				
3	constraintBlock1	rule1			
4	constraintBlock1	rule1	opaqueExpression1	fomula1;formula2;formula3	
5	constraintBlock1	rule2			
6	constraintBlock1	rule2	opaqueExpression2	formula2\;1;formula2\;2	
7					

Figure 2.35: Example of a MapleMBSE Configuration that Acdepts Arrays of Elements in a Single Cell

Modifying Reference Fields in Bound Records

For definitions of bound and unbound records see *Record State (page 38)*.

- *Modifying reference key field*
  If you change the value of a reference key field to a value that represents another element already existing in the model, the element being referenced is changed. An attempt to change to a key value that does not exist in the model will result in an error.

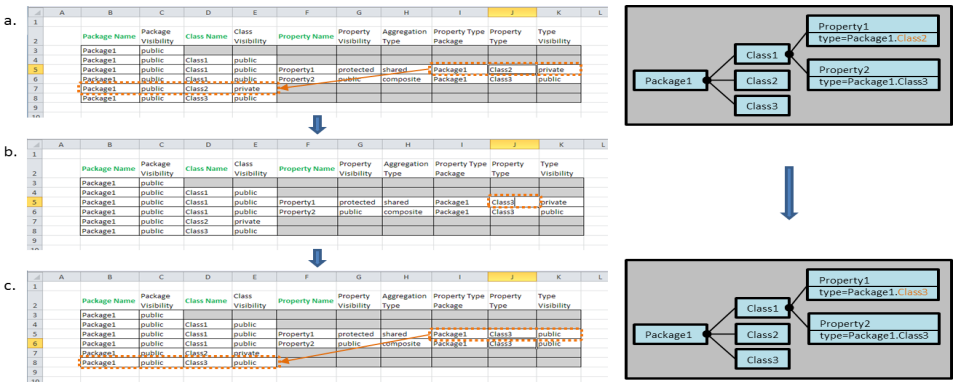


Figure 2.36: Changing Value of Reference Key Field, Property Type

- *Modifying reference nonkey field*
  If you change the value of a nonkey reference field the corresponding field in the referenced element is updated. If there is another field in the SyncView that is bound to this property it will also be updated.

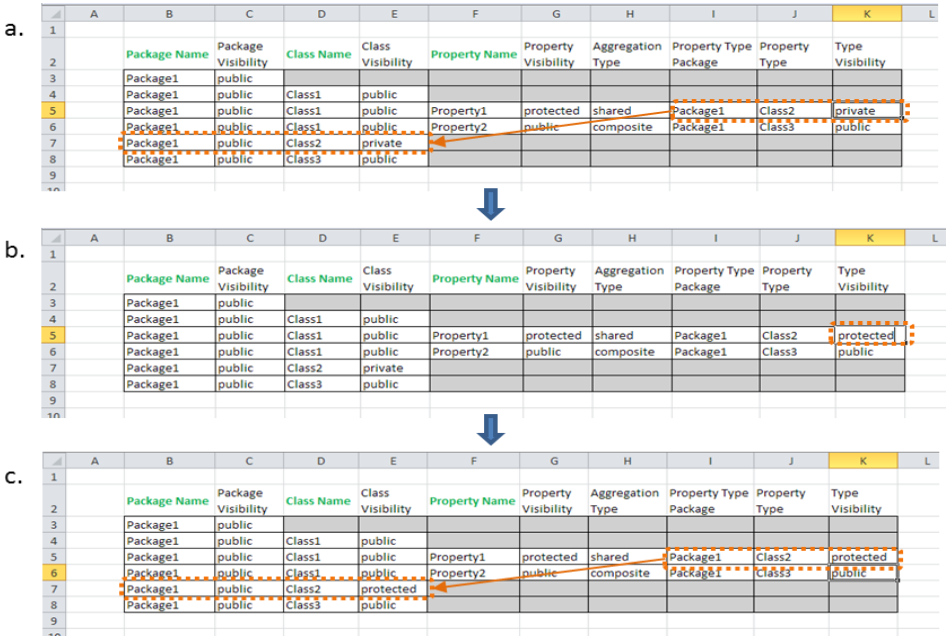


Figure 2.37: Changing Value of Reference Nonkey Field, Type Visibility

- *Deleting value of reference key field*
  Deleting the value of a reference key field is an error.

a.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1	public	Class1	public	Property1	protected	shared	Package1	Class2	private
Package1	public	Class1	public	Property2	public	composite	Package1	Class3	public
Package1	public	Class2	private						
Package1	public	Class3	public						

Delete key reference value

b.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1	public	Class1	public	Property1	protected	shared	Package1	Class2	
Package1	public	Class1	public	Property2	public	composite	Package1		
Package1	public	Class2	private						
Package1	public	Class3	public						

No reference target

Figure 2.38: Deleting Value of a Reference Key Field

- *Deleting value of reference nonkey field*
  If you delete the value of a nonkey reference field the value is cleared. The changes are reflected in the referenced element. However, depending on the model, the value may be reset to a default value.

a.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1	public	Class1	public	Property1	protected	shared	Package1	Class2	private
Package1	public	Class1	public	Property2	public	composite	Package1	Class3	public
Package1	public	Class2	private						
Package1	public	Class3	public						

Delete cell value

b.

Package Name	Package Visibility	Class Name	Class Visibility	Property Name	Property Visibility	Aggregation Type	Property Type Package	Property Type	Type Visibility
Package1	public								
Package1	public	Class1	public						
Package1	public	Class1	public	Property1	protected	shared	Package1	Class2	public
Package1	public	Class1	public	Property2	public	composite	Package1	Class3	public
Package1	public	Class2	public						
Package1	public	Class3	public						

Value reset to default

Figure 2.39: Deleting Value of a Reference Key Field

Unmapped Fields

An unmapped field is a field in a record that is not mapped to any property of the element. As such, values entered in unmapped fields have no influence on the model. However, unmapped fields may be useful for comments or performing Excel operations related to other fields. Formulas for unmapped fields can be entered in the template file (see *MapleMBSE Components (page 1)*) to be available whenever a model is opened. The following example is taken from the UserGuide.MSE configuration file and the *UnmappedFields* spreadsheet that is used with the AddingNewElements.uml model. These files can be found in the **<MapleMBSE>/Example/UserGuide** directory, where <**MapleMBSE>** is your MapleMBSE installation directory.

In **Figure 2.40**, columns B, D, G, and I are unmapped fields. They contain formulas that get updated as you fill out the application. For example, the formula for cell B3 is displayed in the Figure below in the Formula Bar.

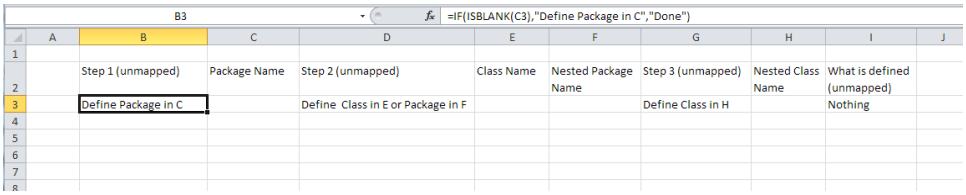


Figure 2.40: Unmapped Fields

As you complete Step 1, filling the package name in column C, the fields get updated.

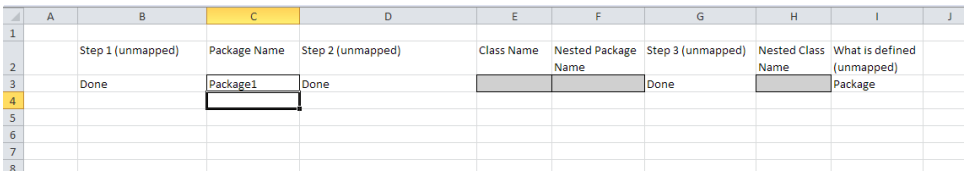


Figure 2.41: Entering a Value for Package Name Results in Updated Mapped Fields

The formulas do not get applied to new rows in a SyncView automatically. You can do so manually by clicking on the bottom right corner of the cell with the formula and dragging in down to expand to the number of desired rows.

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Define Package in C							
5	Define Package in C							
6	Define Package in C							

Figure 2.42: Formulas Must be Updated Manually

The SyncView area is extended to include those rows. The warnings in the cells in column C indicate that the key field needs to be filled. You can extend the other formulas over the SyncView area as well. The formulas get updated with respect to the rows. For example, the formula for B4 is updated with respect to C4.

B4 =IF(ISBLANK(C4),"Define Package in C","Done")

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing
5	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing
6	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing

Figure 2.43: With the SyncView Extended, Key Fields Need to be Filled

Now you can follow Steps specified in in columns B, D, and G to define desired elements.

Tip: This example follows the steps of the example in *Alternative Groups (page 27)*.

Define Class1 in Package1.

a.

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Done	Package1	Define Class in E or Package in F	Class1		Define Class in H		Package
5	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing
6	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing

b.

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Done	Package1	Done	Class1		Done		Class
5	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing
6	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing

Figure 2.44: Defining Class1 Inside of Package1

Define Package2 in Package1.

a.

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Done	Package1	Done	Class1		Done		Class
5	Done	Package1	Define Class in E or Package in F		Package2	Define Class in H		Package
6	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing

b.

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Done	Package1	Done	Class1		Done		Class
5	Done	Package1	Done		Package2	Done		Nested Package
6	Define Package in C		Define Class in E or Package in F			Define Class in H		Nothing

Figure 2.45: Defining Package2 Inside of Package1

Define Class2 in Package2.

a.

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Done	Package1	Done	Class1		Done		Class
5	Done	Package1	Done		Package2	Done		Nested Package
6	Done	Package1	Done		Package2	Define Class in H	Class2	Nested Package

b.

	Step 1 (unmapped)	Package Name	Step 2 (unmapped)	Class Name	Nested Package Name	Step 3 (unmapped)	Nested Class Name	What is defined (unmapped)
3	Done	Package1	Done			Done		Package
4	Done	Package1	Done	Class1		Done		Class
5	Done	Package1	Done		Package2	Done		Nested Package
6	Done	Package1	Done		Package2	Done	Class2	Nested Class

Figure 2.46: Defining Class2 Inside of Package2

2.7 Locking Elements in a Worksheet

The locking features available in MapleMBSE provide a means of controlling simultaneous editing of model elements so that one user's changes are not overwritten.

When users edit models, MapleMBSE attempts to lock the specific model element being changed. If successful, the locks are kept by MapleMBSE and the models will be changed. Otherwise, if unsuccessful, MapleMBSE will report errors and the changes will be reverted.

All the locks automatically acquired by MapleMBSE will be released when:

- Users commit their changes;
- Users reload the model; or
- MapleMBSE terminates.

Note: If MapleMBSE crashes the locks on the elements will remain.

Also note, locking success is not guaranteed (for example, if you are using an older version of MapleMBSE that does not support element locking).

Acquiring Locks While Editing

If you change something in MapleMBSE, it tries to acquire locks of model elements to be changed.

In the image below, the name of requirement, X02 is changed to "New Requirement-Under-Edit".

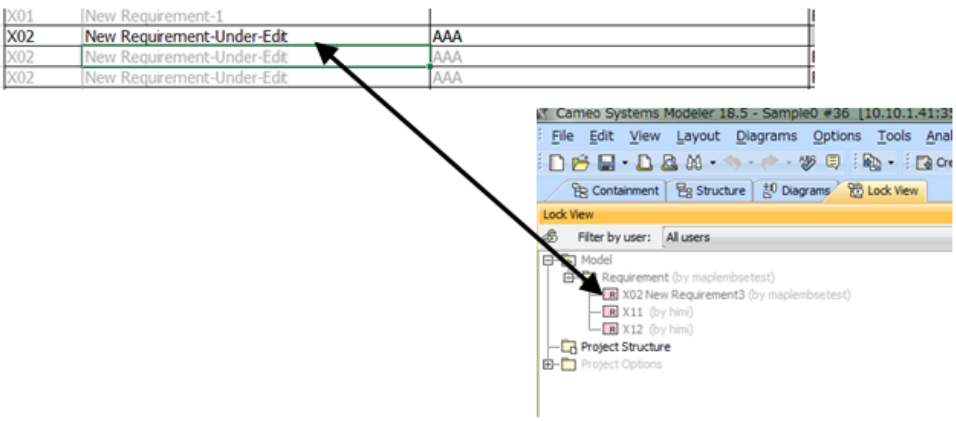


Figure 2.47: MapleMBSE Attempts to Acquire Locks When Element is Being Changed

Failure to Acquire Locks

MapleMBSE shows error message in Excel as "Unable to lock the model elements". MapleMBSE will revert user's changes.

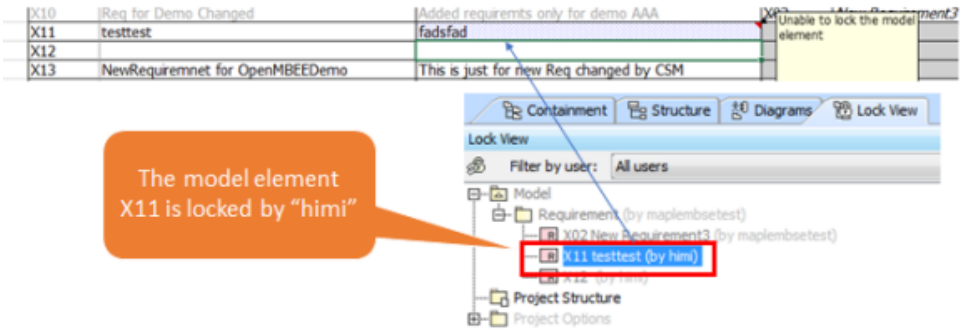


Figure 2.48: Failure to Lock Model Elements Error Message

Committing Changes and Cleaning Up Code

When MapleMBSE commits the changes, terminates, or reloads the models, it automatically releases all of the locks automatically acquired while the user edits the model. Other locks, already acquired by other applications, remain in place.

Note: If MapleMBSE is abnormally terminated (crashes) the locks it has set will remain. Users will have to manually remove these locks through: Cameo Systems Modeler, MagicDraw, or through the Teamwork Cloud Web UI.

Figure 2.49: Manually Releasing Locked Element

Working with Other Dependencies

If there are additional dependencies or requirements, then they are locked.

2.8 Other Editing Operations in Excel

CutCopy Mode

The CutCopy mode in Excel is a mode in which the area selected for cutting/copying is surrounded by a dotted line as shown in the Figure below. Normally in Excel, the CutCopy mode is maintained after pasting is performed. In MapleMBSE the CutCopy mode is cancelled after pasting and processing is done.

	A	B	C	D	E	F
1						
2		Package Name	Class Name	Nested Package Name	Nested Class Name	
3		Package1				
4		Package1	Class1			
5		Package1		Package2		
6		Package1		Package2	Class2	
7						

Figure 2.50: CutCopy Mode

Cell Formatting

- Only the value of the cell is saved in the model. The style and formatting of the cell is not saved and is not reflected in the model.

Appendix A Log Files

A.1 Description of Log Files

The table below describes the log files created by MapleMBSE.
In the table, <InstallDir> refers to the installation folder of MapleMBSE and %USERPROFILE% refers to the value of environment variable, USERPROFILE.

Normally the value USERPROFILE is set to the following:

- In Windows 7 and 10: C:\Users\<User Name>

Log File	Description
%USERPROFILE%\MapleMBSE_Launcher.log	Log data from MapleMBSE.exe
%USERPROFILE%\MapleMBSE_OSGiBridge.log	• Log data from a module (OSGiBridge) which is used to load a module of MapleMBSE in Excel. • You can specify a log level in **<InstallDir>\OSGiBridge.ini**.
%USERPROFILE%\Documents\MapleMBSE.log	• The MapleMBSE.log file is located in the **My Documents** folder • Log data from MapleMBSE itself • You can specify a log level in **<InstallDir>\OSGiBridge\log4j.properties**. • The MapleMBSE.log file will be overwritten whenever starting MapleMBSE.

A.2 Using a Log File to Identify a Problem

In the MapleMBSE.log file, the values in cells updated in the worksheet are recorded.

Note that your computer will run slow due to the large volume of data logged and the size of MapleMBSE.log file will be tens of MB or more.

1. Rename **<InstallDir>\OSGiBridge\log4j.properties**. Name it **log4j-orig.properties**.
2. Rename **<InstallDir>\OSGiBridge\log4j-debug.properties**. Name it **log4j.properties**.
3. Restart MapleMBSE.
4. Reproduce your problem in MapleMBSE.
5. After you have completed the above steps, copy **MapleMBSE.log** located in %USERPROFILE%\Documents in an appropriate location.

6. Return the logging properties to the orignal settings by renaming **<InstallDir>\OS-GiBridge\log4j-orig.properties** to **log4j.properties**.

Appendix B Error Messages and Troubleshooting

B.1 Error Messages When Starting MapleMBSE

Error Message	Description	Resolution
MapleMBSE.exe is already running. Close all Excel Windows, and then confirm MapleMBSE.exe process is terminated.	An attempt was made to start another instance of MapleMBSE but failed because only one instance of MapleMBSE can run at a time.	Close all Excel windows, then confirm that the Excel process (MapleMBSE.exe) has termininated before attempting to launch MapleMBSE again.
File Open Error.	A model file or SyncModel file cannot be opened.	Follow the instructions in the error message dialog.

B.2 Error Messages Displayed When Attempting to Alter the SyncView Table

Messages that will be displayed in a message box when editing are as follows.
These errors were handled in MapleMBSE. If these errors were caused by user, the operation will be cancelled.

Error Message	Description	Resolution
Can’t overwrite in the leftmost column across several rows.	Attempted to overwrite in the leftmost column across several rows but ID is specified in the property of the leftmost cells so that MapleMBSE can manage data - you cannot overwrite in the leftmost column across several rows.	The editing opeation that resulted in the displayed error message will be cancelled. Restart MapleMBSE without saving a model.
Insertion to the top row is currently prohibited.	Attempted to insert a row above the top row of the SyncView table but the operation is not permitted.	Same as above
Can’t delete the insertion row or column.	Attempted to delete the insertion row or column but the operation is not permitted.	Same as above
Can’t move the table.	Attempted to move SyncView table but the operation is not permitted.	Same as above
Can’t update across several rectangular areas at one time.	Attempted to update across several rectangular areas. For	Same as above

	example, copying content into several, nonadjacent areas selected in the table. This operation is not permitted.	
Can’t delete overall editing area	Attempted to delete entire SyncView table but the operation is not permitted.	Same as above
Can’t delete the column.	Attempted to delete a column in the SyncView table but the operation is not permitted.	Same as above
Can’t insert the column.	Attempted to insert the column in SyncView but the operation is not permitted.	Same as above

B.3 Error Messages Displayed When Editing the Model Spreadsheet

Messages that will be displayed by a comment in cell when editing are as follows.

Error Message	**Description**	**Resolution**
The key is duplicated.	An existing key name was specified.	Specify other key name or specify including an end key if the key is not end key.
The key has not been input.	The key has not been input.	Input the key.
No link can be found.	No link shown in the link field exists.	Input a correct value in the link field.
The value is invalid.	The value in the cell is invalid.	Input a correct value.
The cell cannot be changed.	Attempted to change a cell which cannot be changed.	When this error occurs after inserting a row, you may have inserted an invalid feature to the model so delete the row inserted. In addition, when this error occurs after changing the cell value, you may change a feature which cannot be changed in the model so return to the original value. In either case, the change is not reflected on the model.

B.4 Error Messages When Editing (Unhandled Exception)

When the error of Unhandled Exception occurs, name the model and save, and then close MapleMBSE.

B.5 Warning Messages for Remaining Memory

MapleMBSE checks the remaining memory usage whenever you operate. When the available memory becomes insufficient, a warning message will appear. Follow the instructions in the message.

If the warning about the remaining memory reappears frequently, your memory (the heap size of Java) may be insufficient for the model you are editing. Consider expanding the heap size of Java by following the instructions in "Changing the Java Heap Size”.

B.6 Troubleshooting

Changing the Java Heap Size

An error dialog related to insufficient memory may appear when starting or editing MapleMBSE. If this problem occurs adjust the java heap size:

1. Right-click on the MapleMBSE shortcut on your desktop (or the folder of the application) and select Properties.
2. In the **Target:** field add the command line option, /R <memsize> to the end of the shortcut link target (for example, "C:\Program Files\MapleMBSE 2017\MapleMBSE.exe"), where <memsize> is a value, between 800MB to 1100MB, of continuous memory reserved for the Java Virtual Machine, some of this memory will be used for the Java heap, some for the JVM itself. For more information, including other options you can specify, see *Launching from Command Line (page 6)*.
3. Click **OK** or **Apply**.
4. Double-click the shortcut to start MapleMBSE.
5. If an error message appears, increase the value of <memsize> specified in step 2 above. Repeat this until you find a value large enough so that the error message is no longer displayed when starting MapleMBSE.
6. Open OSGiBridge.ini, located in the MapleMBSE installation folder, in Notepad to change the maximum heap size of JVM.
7. Find the entry “-Xmx512m” in this file. Replace "512" with the value of <memsize> from step 2.
8. If MapleMBSE fails to launch, replace -Xmx<memsize>m, set in step 7 with a smaller value for the Java heap, between 150MB to 200MB smaller than <memsize>. This will leave some memory space required for the JVM itself.