magicdraw™
Architecture Made Simple

UML 2

migration manual

No Magic, Inc.
2011

CONTENTS

MIGRATION TO UML 2

How to use this manual

As of MagicDraw 10.0 we have moved from UML 1.4 metamodel to UML 2. It is a significant change so it is very important that you get familiar with these changes before migrating your projects to UML 2.

We've put all our best effort in order to implement UML 1.4 model migration to the version UML 2. However, the new standard is not backward compatible. There might be situations when the migration facilities may not import 100% of the model and data loss may occur.

Please review this information, so you can prepare your projects before loading them with MagicDraw.

- **"Basic concepts and instructions" on page 4** describes techniques and main ideas of migration from UML 1.4 to UML 2.
- **"Migration of modules and profiles" on page 5** provides instructions for preparing modules/ profiles migration to UML 2.
- **"Migration of Teamwork Projects" on page 6** provides instructions for Teamwork Server administrator for moving projects into MagicDraw Teamwork Server.
- **"Problems and solutions of data mapping" on page 7** describes problematic data mapping cases, possible data loss, and possible workarounds to avoid them in migration process. You may prepare your project for migration redesigning potentially problematic parts using older version of MagicDraw.
- **"Tracing migration problems" on page 10** describes ways how to find changes that were done in the migration process.
- **"Autoupdate issues" on page 11** specifies possible autoupdate problems.

Basic concepts and instructions

MagicDraw is based on UML 2 metamodel while previous MagicDraw versions were based on UML 1.4. One of the biggest issues with UML 2 is that it is not backward compatible with UML 1.4. UML 2 also defines new model element types, introduces new properties. Some elements and properties were removed.

Projects that are based on UML 1.4 have to be converted to UML 2 in order to work with MagicDraw . Project load mechanism in MagicDraw will automatically detect UML 1.4 projects and will convert it to UML 2.

We strongly recommend backing up all the projects before conversion.

We also do not recommend migrating to UML 2 in the middle of intensive project.

NOTES	You will not be able to open the converted project in previous MagicDraw versions. Only native MagicDraw files with UML 1.4 model can be loaded correctly into MagicDraw . It is recommended to load third-party XMI files with UML 1.4 into previous MagicDraw (up to 9.5 version) first, then save into native MagicDraw XML file and try to open (migrate to UML 2) using MagicDraw .

Migration of modules and profiles

MagicDraw projects may use multiple modules or profiles.

Project based on UML 1.4 cannot be loaded into MagicDraw , if it uses UML 1.4 modules. You have to convert all used modules to UML 2 before converting your main project to UML 2.

Conversion of Modules

You should start from the independent modules, i.e. ones that are not using other modules.

1. Back up the module.
2. Open module as normal project. It will be converted into UML 2.
3. Save it into the same file.

NOTE	It will be not possible to load converted module with previous MagicDraw versions or use them in projects based on UML 1.4.

After all modules are converted into UML 2, you will be able to load projects that are using those modules.

Solving Cyclic Usages

Cyclic usage is a situation when module A is using module B and module B is using module A. In this case, neither module A nor module B can be loaded with MagicDraw . The solution to this problem is:

1. Open one of the modules with previous MagicDraw version.
2. Try to solve cyclic usage by importing (merging) one module into another or remove cyclic usage.

Converting Stereotypes/Profiles

Stereotypes in UML 2 metamodel must be contained by a Profile model element, and all tags have to be defined as properties of stereotype. In UML 1.4 metamodel it was allowed to store stereotypes and tags anywhere in the model with no restrictions.

To fix this problem during migration process, MagicDraw creates a Profile named "**Profile_for_<project_name>**" and moves all stereotypes from various packages into it.

For tags that are not owned by stereotype, a new owning stereotypes is created. It will have the same name as tag.

For example, if you had a tag definition called "duration", after project conversion to UML 2 you will have a stereotype *<<duration>>,* which will have a tag definition "duration". Stereotype will be placed in a Profile called "**Profile_for_<project_name>**". All elements that had this tag in UML 1.4, in UML 2 will also have the stereotype assigned.

UML Standard Profile was redesigned as well. Base classes for some stereotypes have been changed; some stereotypes were removed. Stereotypes with modified base classes will be still assigned to elements, even if the element cannot have this stereotype assigned. These elements will be marked as problematic. For more details, see “Tracing migration problems” on page 10.

All stereotypes that were used in old projects but were removed from UML Standard Profile, will be added into new Profile "**Profile_for_<project_name>**".

For more information about stereotypes mapping, see "Stereotypes and tags", on page 8.

Migration of Teamwork Projects

It is strongly recommended to install MagicDraw Teamwork Server into new location and prepare old Teamwork Server projects for migration.

Preparing old Teamwork projects

1. Make sure that all changes are saved into old Teamwork Server. Notify all teamwork users to commit all changes and unlock all data.
2. Please review the project to avoid possible migration problems (see “Problems and solutions of data mapping” on page 7) and make appropriate modifications.
3. Install MagicDraw Teamwork Server to a new location.
4. During the first launch of Teamwork Server, select the **Import all projects** checkbox in the opened **Import Configuration** dialog box.

Converting Teamwork Projects

All teamwork projects have to be converted to UML 2 data following the same instructions as for regular MagicDraw projects (See “Migration of modules and profiles” on page 5).

It is recommended to convert all teamwork modules before converting projects. All standard profiles, bundled with MagicDraw installations, will be updated automatically on the first Teamwork Server launch.

Steps for project or module converting:

1. Open the latest project version from the Teamwork Server.
2. Commit it to the Server.

NOTE	Only latest project version will be converted into UML 2, all old versions (history) will remain based on UML 1.4. It will be possible to open any old version only in read-only mode.

Special Cases Of Teamwork Modules

When fixed version of teamwork module is used

If old project was using a fixed version of teamwork module (fixed version), this option will be automatically turned off after conversion. Your project will be using the latest version of teamwork module.

When you will load a project that is using a fixed version of a teamwork project, you will be warned, that MagicDraw will automatically switch to the latest version of a teamwork module. If usage of a particular version of teamwork module is very important, you should not commit loaded project to teamwork server.

Follow the steps to preserve usage of a particular version of teamwork module:

1. Login into Teamwork Server with MagicDraw . From the **Teamwork** menu, choose **Projects**.
2. Select your project, expand modules tree and check, which version of a teamwork module is used.
3. Find this module in the list of projects (select the **Show Modules** checkbox) and click the **Versions** button to open **Versions** dialog box.
4. Select version that is used by your project and set it as the latest one (click **Set As Latest**).
5. Click the **Open** button to open this version. During this step you will convert the latest version of teamwork module to UML 2.
6. Commit converted module into Teamwork Server and close it.

7. Open project that uses a fixed version of this module. The last module version will be used.
8. Commit converted project into Teamwork Server and close it.
9. From the **Teamwork** menu, choose **Projects** again.
10. Select your project in the list of projects; expand tree of modules and change module version from **Latest** to a latest version number.
11. Find this module in projects list (select the **Show Modules** checkbox) and click the **Versions** button to open **Versions** dialog box.
12. Select a version that was latest before these manipulations and set it as latest one (click **Set As Latest**).
13. Open this module and commit to Teamwork Server. During this step you will convert the latest version of teamwork module to UML 2.

After these steps your project will be using a fixed version of a teamwork module that is converted to UML 2, but other projects will be able to use the latest version of this module.

If multiple projects are using various versions of the same module, repeat these steps for every project.

When a module/profile is used by a custom diagram

It is recommended to convert your teamwork modules, which are used in custom diagrams, before converting your projects. All standard profiles, bundled with MagicDraw installations, will be converted automatically on the first Teamwork Server launch

Converting Local Versions of Teamwork Projects

Locally saved teamwork projects based on UML 2 can be committed into Teamwork Server only when there are no newer versions on the server and user is connected to the Teamwork Server.

NOTE	After Teamwork project is converted to UML 2, it will not be possible to commit local versions based on UML 1.4.

Converting Teamwokr Projects on IDE Integration

1. Make sure that all changes are saved into old Teamwork Server. Notify all teamwork users to commit all changes and unlock all data.
2. Launch standalone MagicDraw client and login into Teamwork Server.
3. Convert teamwork projects to UML 2. For more details, see “Converting Teamwork Projects” on page 6.
4. Save integration project locally by overriding old project file.

Problems and solutions of data mapping

If you want to be sure that your migration process is as fluent as possible, we recommend reading suggestions how to modify your projects based on UML 1.4 before conversion to UML 2 to avoid mapping problems.

NOTE	All modifications based on these recommendations should be applied using MagicDraw 9.5 or older.

Class Diagram

	Issue	**Solution**
Dependencies	Dependency can be connected only between NamedElements in UML 2. Dependencies between other elements will be deleted after load.	Try to avoid dependencies between generalizations, merge, import and other relationships that are not NamedElements in UML 2. If dependency holds important information, try to connect it to other semantically close element that is supported in UML 2.
Subsystem	Subsystem is no longer subtype of package.	Subsystem will be automatically converted to a component with stereotype <<*subsystem*>> during conversion. If the subsystem had inner elements, they will be moved to a package that now owns the component.
Stereotypes and tags	Stereotypes in UML 2 metamodel have to be contained by Profile model element, and all tags have to be defined as properties of stereotype. UML 1.4 metamodel allowed to store stereotypes and tags anywhere in the model with no restrictions.	As it was described in “Converting Stereotypes/ Profiles” on page 5, MagicDraw will create new profile and owner stereotypes for all tags, however you may want to do some changes manually. We recommend creating stereotypes for "loose" tags. Grouping multiple tags into same stereotype could be a good option as well. If you have UML 1.4 based project where all your stereotypes placed in a package, assign <<*profile*>> stereotype to this package. In this case package will be converted to a Profile model element. If you had many packages with stereotypes, you can move them into a new package with stereotype <<*profile*>> assigned.

Sequence Diagram

	Issue	**Solution**
Concurrent lifelines and message branching	Concurrent lifelines and message branching are no longer supported in sequence diagram. They will be loaded into MagicDraw , but user will not be allowed to draw such elements.	UML 2 provides new notation for concurrency and branching using combined fragments. It is recommended to redraw sequence diagrams before or after migration.

State Diagram

	Issue	**Solution**
StubState and SynchState	StubState and SynchState elements are no longer supported in UML 2. StubStates and SynchStates will be removed together with connected transitions.	Please try to remove StubStates and SynchStates from state diagrams before migration. We suggest redesigning your model in alternative ways, trying to connect important transitions to other states.
Parameters of event	Event can't have parameters in UML 2. All parameters will be lost during the migration.	Use parameters of Operation when using CallOperationEvent. Add additional parameters to operation if you are using more CallOperationEvent parameters than Operation has. Assign stereotypes and tags to Operation parameters instead of CallEvent parameters.

The same rules should be applied to parameters of SendSignalEvent and attributes of Signal.

Activity Diagram

	Issue	Solution
States	UML 2 Activity diagram is no longer based on StateMachine, so State model elements (SimpleState, CompositeState, and ConcurrentState) can not be used in activity diagram anymore.	Redraw your activity diagrams not using ConcurrentStates, move internal elements from ConcurrentState to diagram and show concurrent flows by using synchronization bars or other alternative notation. CompositeState in activity diagrams should be modified in a similar manner, by moving internal elements into diagram. SimpleState will be mapped to ObjectNode with this state assigned to the InState property.

Implementation Diagram

	Issue	Solution
Deployment	Components cannot be deployed in Nodes in UML 2; artifacts related with these components should be deployed instead.	During migration, MagicDraw will automatically create artifacts (named as components) that are deployed in Nodes and will manifest Components (Manifestation relationship between artifacts and components will be created). Diagram view will not change - Components will still be located over Nodes, however these components are not added into Nodes. You should redraw your diagrams, showing artifacts and relations between components, artifacts and nodes. Drag artifacts that were created automatically from browser tree directly to diagrams, use "Display paths" to display relationships.

Tracing migration problems

You are able to select automatic marking of problematic elements during conversion process.

All recognized data mapping or other migration problems will be marked by adding **ToDo** tagged value with problem description into the problematic element.

To trace marked elements

First of all, try to avoid cases, described above of this document.

If data loss cannot be avoided, all problems, conflicts, and data loss occurrences are registered to the model element's **ToDo** tags on project load:

1. Load your project. The warning message appears.
2. Click **Yes**. Warnings are displayed in **ToDo** tags.

3. Double-click the element or, from the shortcut menu, choose **Specification**. Information about changes is added to the **ToDo** field in the **General** tab.

NOTE	Use the **Find ToDo** feature to search for elements containing **ToDo** information. From the **Tools** main menu, choose **Find ToDo**.

Autoupdate issues

WARNING!	Be cautious with MagicDraw autoupdate.

We do not recommend to use autoupdate, because you may need previous MagicDraw version installation for migrating your projects to UML 2.

MagicDraw is able to convert UML 1.4 to UML 2, but some mapping problems may appear. Previous MagicDraw version should be used for solving these problems before conversion. For more details about mapping solutions, see "Problems and solutions of data mapping" on page 7.

It's recommended to download MagicDraw installation manually and install it into new location (not the location suggested by default).

If you have read this document carefully and you think you will not have problems in all your projects with elements migration, you may use autoupdate function.

Autoupdate for MagicDraw Teamwork Server is disabled.