Problems and solutions of data maping

To ensure that your migration process is as fluent as possible, we recommend reading the following suggestions to modify your projects based on UML 1.4 before conversion to UML 2 to avoid mapping problems.

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

On this page

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 a dependency contains important information, try to connect it to another semantically close element 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 must be contained by Profile model element, and all tags must be defined as properties of stereotype. The UML 1.4 metamodel allowed you to store stereotypes and tags anywhere in the model with no restrictions.	As described in Converting Stereotypes /Profiles, 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 the same stereotype may be a good option as well. If you have a UML 1.4 based project with all stereotypes placed in a package, assign a «profile» stereotype to this package. In this case the 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 the stereotype «profile» assigned.

Sequence Diagram

	Issue	Solution
Concurrent lifelines and message branching	Concurrent lifelines and message branching are no longer supported in a sequence diagram. They will be loaded into MagicDraw, but the user will not be allowed to draw these elements.	UML 2 provides new notation for concurrency and branching using combined fragments. We recommend redrawing 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 the parameters of Operation when using CallOperationEvent. Add additional parameters to an operation if you are using more CallOperationEvent parameters than the Operation has. Assign stereotypes and tags to Operation parameters instead of CallEvent parameters.

The same rules should be applied to the parameters of the SendSignalEvent and the attributes of the 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 no longer be used in the activity diagram.	Redraw your activity diagrams without using ConcurrentStates, move internal elements from ConcurrentState to the diagram and show concurrent flows by using synchronization bars or other notation. CompositeState in activity diagrams should be modified in a similar manner by moving internal elements into the 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 to these components should be deployed instead.	During migration, MagicDraw will automatically create artifacts (named as components) deployed in Nodes and will manifest Components (Manifestation relationship between artifacts and components will be created). The 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 created automatically from the browser tree directly to diagrams, use “Display paths” to display relationships.