

# Cameo Concept Modeler Documentation

Cameo Concept Modeler (Concept Modeler) is a plugin for MagicDraw, the award-winning software modeling tool. It is designed to model real-world concepts, import/export a model from/to an OWL ontology, and generate glossaries in plain English for clearer and more informative source of knowledge for any domain.

## 19.0 LTR SP2 Version News

## 19.0 LTR SP1 Version News

## 19.0 LTR Version News

## Cameo Concept Modeler (CCM) Quick Start Guide

### Introduction

- MDA
- Concept modeling purpose
- The role of ontologies and reasoners
- Open-world assumption vs. closed-world assumption
- Information modeling purpose

### Concept Modeler Capabilities

#### Concept Modeling Semantics

- Classes
- Property ownership
- Global properties
- Subproperty
- Generalization
  - Overlapping and incomplete subclasses
  - Disjoint subclasses
  - Complete subclasses
  - Disjoint and complete subclasses
- Anonymous union classes
- Inverse properties
- Property restrictions
  - Existential quantification constraint
  - Universal quantification constraint
  - Cascading restrictions
- Annotation and annotation properties
- Preferred Annotation Property
- Advanced Modeling Patterns
  - Association Class
  - Facets
    - Role
    - Phase
- Property chain
- Equivalent properties
- Equivalent classes
- Multiplicities
- IRI tagged value
  - Effective IRI meta-property
  - Synchronize UML Package URI and Resource IRI
- Cardinality restrictions
- Complement Of
- Concept model export URI style
- OWL export folder
- Intersection
- UPCM library in CCM
- Conditions
- Importing OWL
- Working with superclass intersection

#### UML to Equivalent OWL in OWL Functional Syntax

- Class
- Class generalization
- Generalization with disjoint subclasses
- Generalization with subclass completeness
- Anonymous union class
- Class with Datatype property
- Class with Self-Referential Object Property

- Class with object property
- Property holder with datatype property
- Property holder with self-referential object property
- Property holder with object property
- Class with object property without range
- Class with subproperty
- Class with universal quantification constraint on property I
- Class with universal quantification constraint on property II
- Class with existential quantification constraint on property
- Property holder with self-referential subproperty
- Property holder with subproperty
- Class with subproperty without a range
- Class with necessary and sufficient property
- Class with property having unspecified multiplicity
- Class with inverse property
- Annotation and annotation property
- Asymmetrical inverse property
- Disjoint classes
- Property chains
- Equivalent property
- Equivalent class
- Class with Asymmetric Object Property
- Class with Functional Object Property
- Class with Inverse Functional Object Property
- Class with Irreflexive Object Property
- Class with Reflexive Object Property
- Class with Transitive Object Property
- Property with a maximum but no minimum cardinality
- Property with multiple domains and ranges
- Property restriction from a different namespace
- Necessary and sufficient conditions of anonymous subclasses
  - Intersection subset of Union
  - Intersection equivalent to Union
  - Intersection subset of a Restriction
  - Intersection equivalent to a Restriction
  - Intersection disjoint with a Restriction
  - Intersection subset of an Intersection
  - Intersection equivalent to an Intersection
  - Intersection subset of Complement
  - Intersection equivalent to Complement
  - Intersection disjoint with Complement
  - Intersection subclass of Class
  - Union subset of Union
  - Union equivalent to Union
  - Union subset of Restriction
  - Union equivalent to Restriction
  - Union disjoint with Restriction
  - Union subset of Intersection
  - Union subset of Complement
  - Union equivalent to Complement
  - Union has member Complement
  - Union disjoint with Complement
  - Union subset of Class
  - Complement subset of Union
  - Complement disjoint with Union
  - Complement subset of Restriction
  - Complement subset of Intersection
  - Complement disjoint with Restriction
  - Complement has member Restriction
  - Complement equivalent to Restriction
  - Complement disjoint with Intersection
  - Complement subset of Complement
  - Complement equivalent to Complement
  - Complement has member Complement
  - Complement subset of Class
  - Restriction subset of Union
  - Restriction disjoint with Union
  - Restriction subset of Restriction
  - Restriction equivalent Restriction
  - Restriction disjoint with Restriction
  - Restriction subset of Intersection
  - Restriction disjoint with Class
  - Restriction subset of Class
  - Restriction disjoint with Complement
  - Restriction subset of Complement
  - Restriction disjoint with Intersection
  - Class subset of Union
  - Class subset of Restriction
  - Class equivalent to Restriction

- Class disjoint with Restriction
- Class subset of Complement
- Class disjoint with Class
- Class disjoint with Complement
- Class disjoint with Intersection
- Class disjoint with Union
- Class equivalent to Class
- Class subset of Class
- Class subset of Intersection
- Complement disjoint with Class
- Complement disjoint with Complement
- Complement equivalent to Class
- Complement has member Class
- Complement has member Intersection
- Complement has member Union
- Intersection disjoint with Class
- Intersection disjoint with Intersection
- Intersection disjoint with Union
- Intersection equivalent to Class
- Intersection has member Class
- Intersection has member Complement
- Intersection has member Intersection
- Intersection has member Restriction
- Intersection has member Union
- Restriction has member Class
- Restriction has member Complement
- Restriction has member Intersection
- Restriction has member Restriction
- Restriction has member Union
- Union disjoint with Class
- Union disjoint with Intersection
- Union disjoint with Union
- Union equivalent to Class
- Union has member Class
- Union has member Intersection
- Union has member Restriction
- Union has member Union
- Association classes

## Natural Language Glossary

- Equivalent classes in NLG

## Usage

- Creating a concept modeling project
- Creating a concept model
  - Converting a UML model into a concept model
  - Creating a property chain
    - Deleting a property chain
    - Editing a property chain
  - Creating equivalent properties
    - Deleting an equivalent property
    - Editing an equivalent property
  - Creating equivalent classes
- Creating an XML catalog file
- Importing an OWL ontology to a concept model
  - Updating the XML catalog file
  - Setting the OWL import catalog in MagicDraw
  - Creating a path variable to share OWL import catalog files
  - Using a path variable to share OWL import catalog files
  - Importing an OWL ontology file
- Displaying and hiding IRI
- Exporting your concept model to an OWL ontology
  - Setting destination for the OWL export folder
  - Setting the concept model export syntax
  - Setting the concept model export URI style
  - Setting the concept model URI
  - Specifying file export paths
  - Specifying IRI ontology versions
  - Use Path Variables to Export a Concept Model to an OWL Ontology
  - Exporting your concept model
  - Exporting models and concept models at any level in package hierarchy
  - View the CCM watermark in an exported OWL
- Logging during OWL importing and exporting
- Adding a concept model to Teamwork Cloud and export it as an OWL ontology
  - Adding a concept model to Teamwork Cloud
  - Exporting a concept model from TWCloud to an OWL ontology
- Automatically Generating Glossaries

- [Creating a glossary table](#)
- [Rebuilding a glossary table](#)
- [Viewing a glossary](#)
- [Creating a property holder](#)
- [Restriction](#)
  - [Creating Restrictions](#)
  - [Removing Restrictions](#)
- [Working with subproperties](#)
- [Redefined property](#)
- [Subsetted property](#)
- [Creating a necessary and sufficient condition](#)
- [Working with subclasses](#)
  - [Making subclasses disjoint](#)
  - [Making subclasses complete](#)
  - [Making subclasses overlapping](#)
  - [Making subclasses incomplete](#)
- [Working with the natural language glossary](#)
  - [Generating a natural language glossary](#)
  - [Exploring the natural language glossary](#)
  - [Customizing your natural language glossary](#)
    - [Variables of the natural language glossary](#)
    - [Selecting an ordered list of annotation properties](#)
    - [Including property definitions in the Natural Language Glossary](#)
- [Existing Project Migration](#)
  - [Migrate Older Models to Use Relative IRIs](#)
  - [Sufficient constraint](#)
  - [Updating symbol styles in older projects](#)
- [Adding property subsetting](#)
- [Removing property subsetting](#)
- [Using Concept Modeling Capabilities with non-CCM projects](#)
- [Working with Complement Of](#)
- [Importing an OWL ontology into a non-CCM project](#)
- [Generalizations with read-only subclasses](#)
- [Working with annotations](#)
  - [Creating annotations](#)
  - [Showing annotations on the diagram](#)
  - [Showing an annotation in the Documentation pane](#)
  - [Working with annotation properties](#)
    - [Applying an annotation stereotype to a comment](#)
    - [Associating an annotation property with an annotation](#)
    - [Defining an annotation property](#)
    - [Importing an ontology that defines annotation properties](#)
    - [Selecting a Preferred Annotation Property for a UML Comment or Annotation](#)
- [Experimental Features](#)
- [Working with association classes](#)
- [Working with roles](#)
- [Working with phases](#)
- [Creating a datatype property](#)
- [Using the log file to track changes during import](#)
- [Working with intersection](#)

## References