

# News in RSA-RTE 10.1

updated for sprint 2017.28

*Mattias Mohlin, July 2017*

# Overview

- **Now based on Eclipse Neon.3 (4.6.3)**
  - Many general improvements since Eclipse Mars
- **Contains everything from RSARTE 10 and also additional features and bug fixes**
  - See the What's New presentation for RSARTE 10 to learn about new features



IBM Rational® Software Architect RealTime Edition

Version: 10.1.0.v20170512\_1332

Release: 2017.19

(c) Copyright IBM Corporation 2004, 2016. All rights reserved.

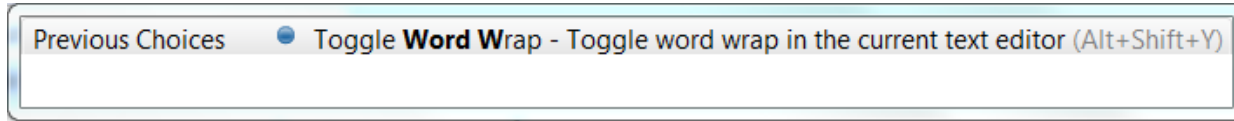
(c) Copyright HCL Corporation 2016, 2017. All rights reserved.

Visit <http://www.ibm.com/developerworks/connect/rsarte>



# Eclipse 4.6.3 (Neon)

- Word wrap in text editors
  - Use the shortcut Alt+Shift+Y or access the "Toggle Word Wrap" command from Quick Access



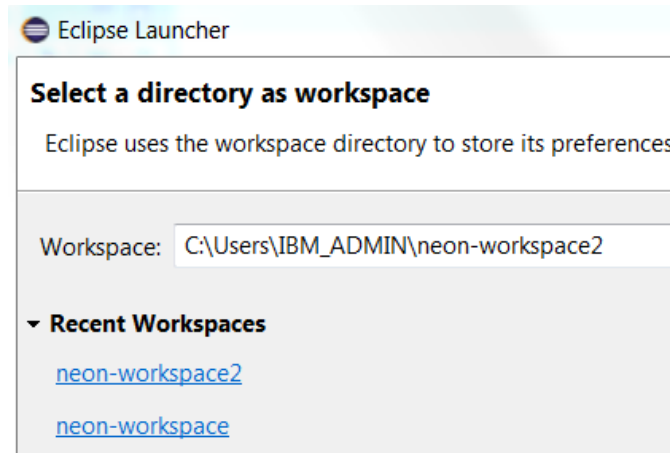
- Commands for "zooming" in text editors by changing the font size
  - Ctrl++ for zooming in and Ctrl+- for zooming out

```
Implementation for CPPModel::MyFile ☒
Showing code from the model MyFile
36     event->m_receiver =
InstanceManager::instance().getIdFromExe
37     m_eventQueue->insert(event);
38     return true; // Event consumed
39 }
40 return false; // Event not consumed
41 }
```



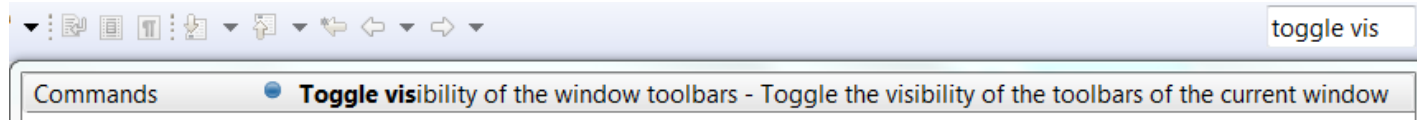
# Eclipse 4.6.3 (Neon)

- Autosave of dirty editors
  - Set a timer to automatically save modified editors after a period of inactivity
- Terminate and relaunch
  - Makes it simpler for users who prefer to only have one launch (e.g. a debug session) active at the same time
- Shortcuts to recently used workspaces when launching RSARTE

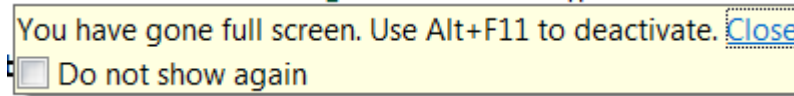


# Eclipse 4.6.3 (Neon)

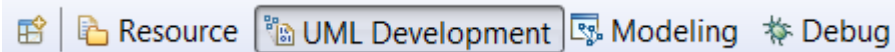
- Command for toggling visibility of window toolbars (to maximize space for editors and views)
  - Assign a key-binding to this command, or access it through Quick Access



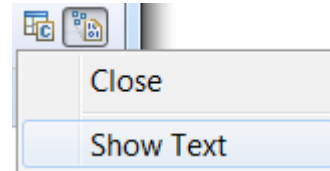
- Full screen support (to maximize RSARTE's usage of the screen)



- Perspective names hidden by default (to save space in toolbar)

*before*  Resource UML Development Modeling Debug

*now* 

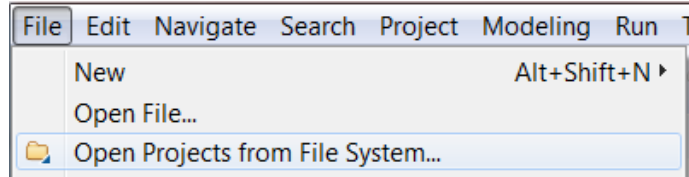


The names can be shown using the context menu



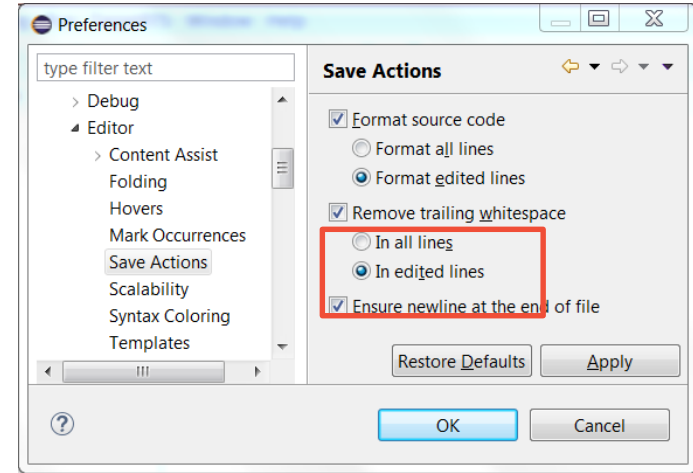
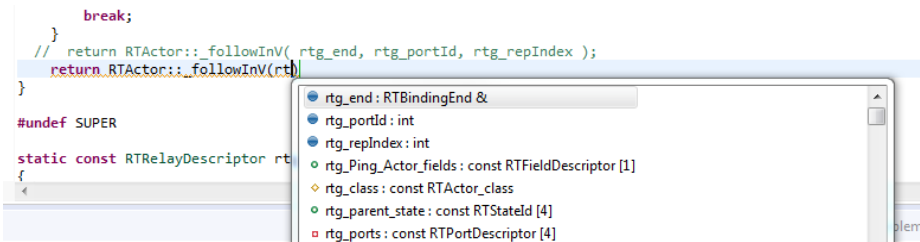
# Eclipse 4.6.3 (Neon)

- New smart wizard for importing projects
  - No longer necessary to use different wizards for different kinds of Eclipse projects
- For more information about Eclipse improvements see
  - News in Eclipse 4.6.3 (Neon) <https://www.eclipse.org/eclipse/news/4.6/>

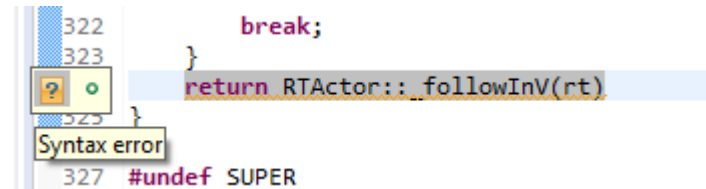


# CDT 9.2.1 (included as part of Eclipse Neon.3)

- Save Action for automatically formatting edited lines when saving a file
- Parameter guessing when typing function calls

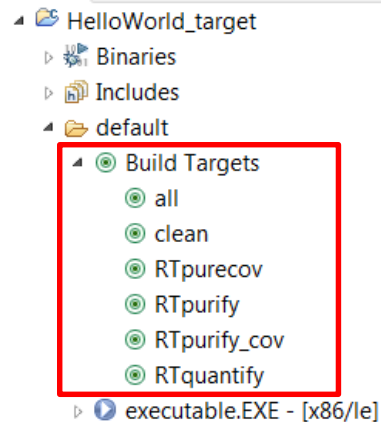
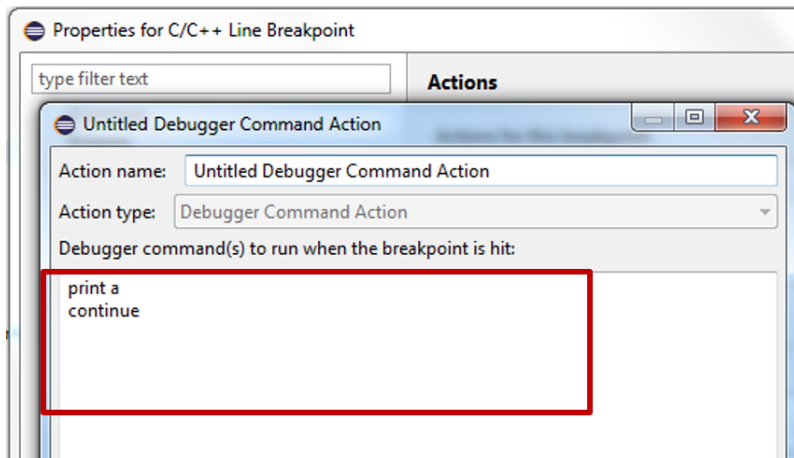


- Support for **decltype(auto)** type-specifiers
- Expansion of icons in the editor ruler
  - Helps when there are multiple icons on the same source code line



# CDT 9.2.1

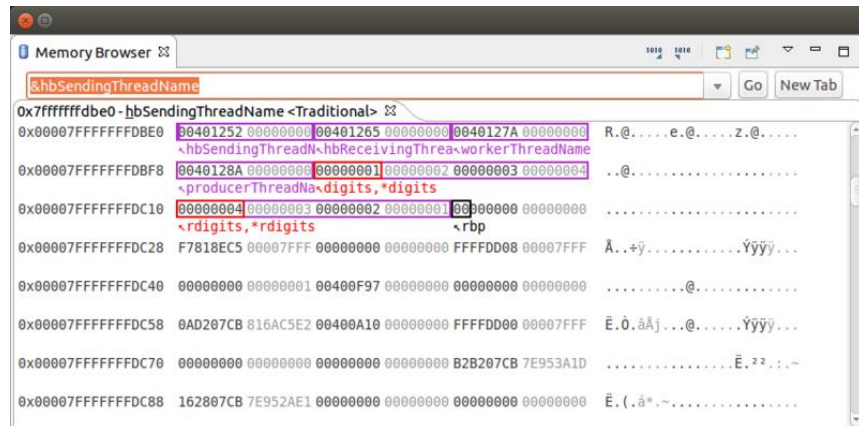
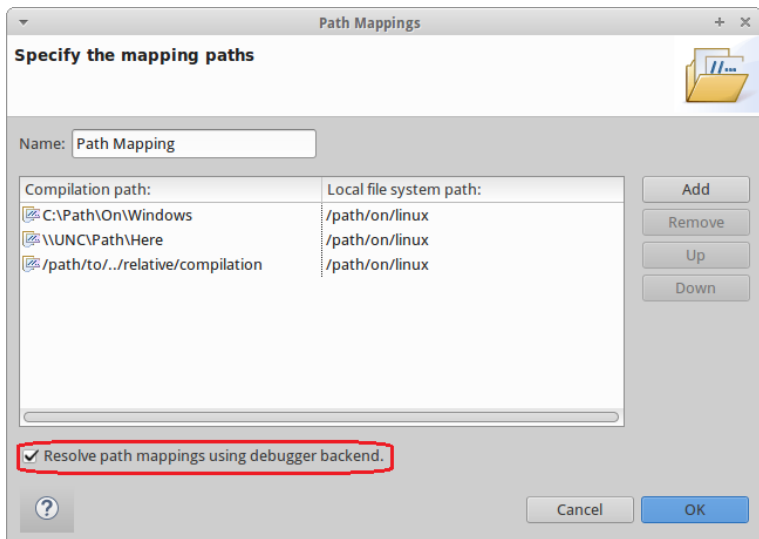
- Command for commenting/uncommenting selected lines in makefile editor (Ctrl + /)
- Build targets (previously called Make targets) now show up in the Project Explorer. They can be run by double-click.
- Support for running commands in the debugger when a breakpoint is hit





# CDT 9.2.1

- Local variables and registers in the Memory Browser
- Improved source lookup when debugging



For more information about news in CDT 9.2.1 see <https://wiki.eclipse.org/CDT/User/NewIn90> and <https://wiki.eclipse.org/CDT/User/NewIn91>



# EGit 4.6.1

- RSARTE now includes an EGit integration that is based on the latest version of EGit
- RSARTE files can now be opened in a text editor from the EGit History view.
  - Previously this command was disabled, but its more useful to at least be able to open the files in a text editor
- See [https://wiki.eclipse.org/EGit/New\\_and\\_Noteworthy/4.6](https://wiki.eclipse.org/EGit/New_and_Noteworthy/4.6) for news in EGit 4.6.x



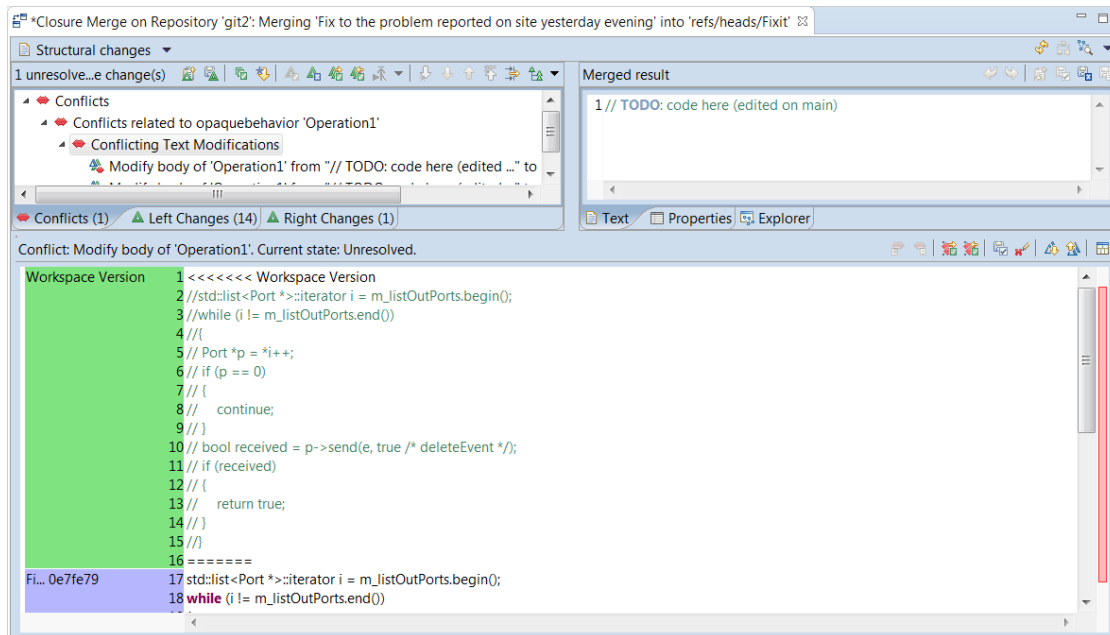
# Codan

- Codan is the Code Analysis feature of CDT. It has been improved in CDT 9.2.1.
- To use it on the CDT projects generated by RSARTE you should enable the preference *UML Development – Real Time C++ Transformations – Generate additional information for Code Analysis*
  - Adds standard include paths to generated CDT projects
  - Makes Codan report much fewer errors, and therefore makes it more stable
  - Use of this feature may increase the build time somewhat



# Improved Text Merge

- Merging text (code, comments etc.) is now easier thanks to a new text merge editor
  - Replaces the old "sub-merge" (except when merging rich text)
  - Conflicts can be resolved by choosing the contributor versions, but also by direct editing
  - Non-conflicting text blocks are merged automatically but can also be edited if needed
  - The editor has line numbers and a Find command
  - The "Merged result" view is updated when saving or when the text merge editor loses focus
  - Easy to get an overview of all changes made in a merged text compared to the ancestor version
  - The colors used can be customized in preferences at *General – Compare/Patch – Modeling Compare/Patch – UML Compare/Merge*



# Simplified Compare/Merge Editor

- Some commands were removed

- *Edit Merged Result*

- It is recommended to instead use compare/merge tasks to keep track of changes that need to be done after the merge has been completed.

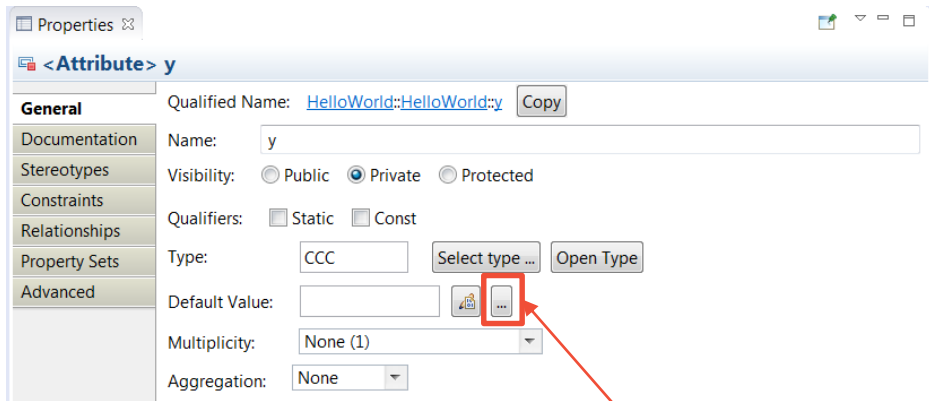
- *Revert Session*

- This command did not work in all compare/merge contexts, and it's better to revert a session by simply closing the Compare/Merge editor and launch it again.



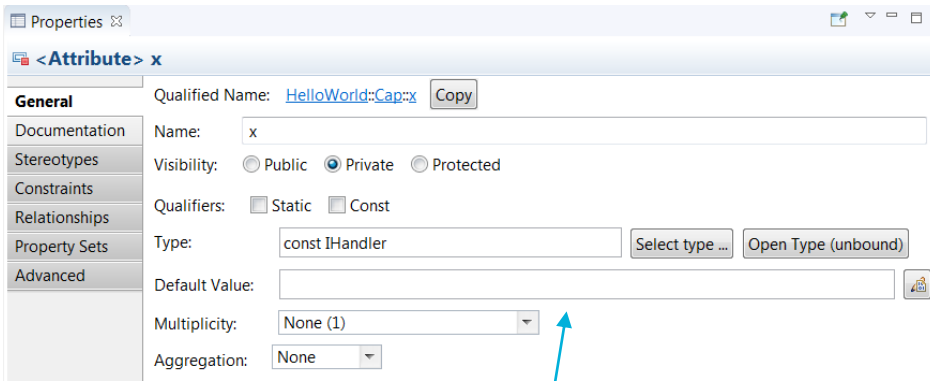
# Improved Layout of Property Pages

- The layout of some property pages have been improved to make them more compact, and to give more space for editing important properties  
For example:



*before*

Removed useless Browse button for default values



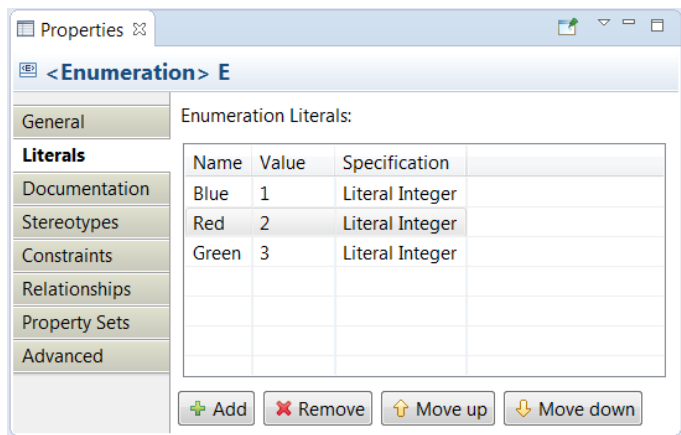
*now*

More space for editing type and default value of an attribute

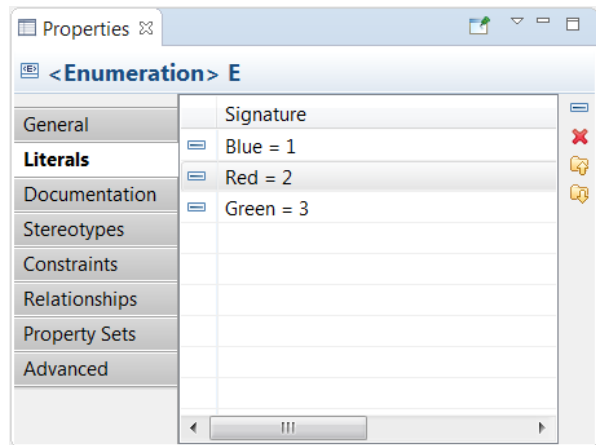


# Improved Editing of Literals and Parameters

- The Properties editor now allows enumeration literals and operation parameters to be edited using textual syntax
  - Similar to how operations and attributes are edited
  - The context menu provides navigation to the literal or parameter



*before*

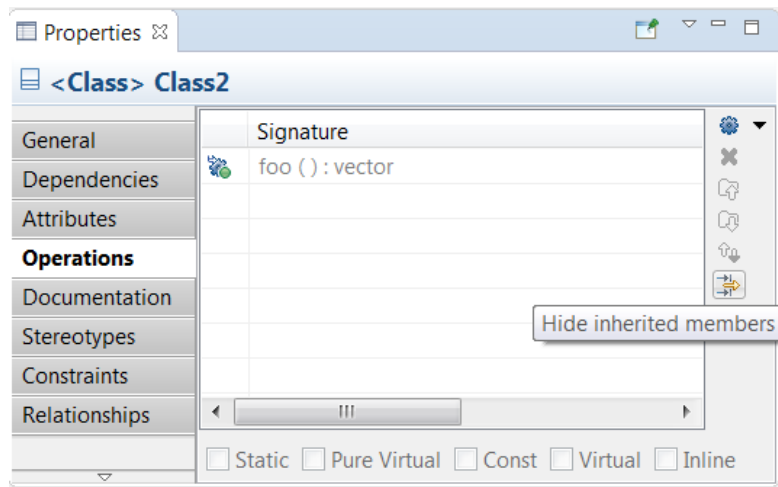


*now*



# Hiding Inherited Members in the Properties View

- The Attributes and Operations tab in the Properties editor now provides a toggle button for hiding/showing inherited attributes or operations
  - Sometimes it's useful to hide inherited members to only see the attributes or operations that are defined in the selected class.





# Improved Relationships Tab in the Properties View

- More relationships can now be seen in the Relationships tab without scrolling vertically
- Also, the tables now resize automatically if the Properties view is resized. For example, it's now useful to maximize the Properties view in case there are lots of relationships to look at.
- The tables can now be sorted. For example, by sorting on the Type column it becomes easy to find relationships of a particular kind (such as Generalizations).

The screenshot shows the Properties View for a class named 'AClass'. The Relationships tab is active, displaying two tables of relationships. The first table, 'Relationships originating from this element:', lists relationships where AClass is the source. The second table, 'Relationships targeting this element:', lists relationships where AClass is the target. Both tables have columns for Name, Type, and Other Related Element(s). Buttons for 'Add...', 'Delete', and 'Navigate' are present below each table.

Relationships originating from this element:			
	Name	Type	Other Related Element(s)
↗	(Cap)	<Dependency>	<a href="#">Cap</a>
↗	(Class2)	<Generalization>	<a href="#">Class2</a>

**Relationships**

Add... Delete Navigate

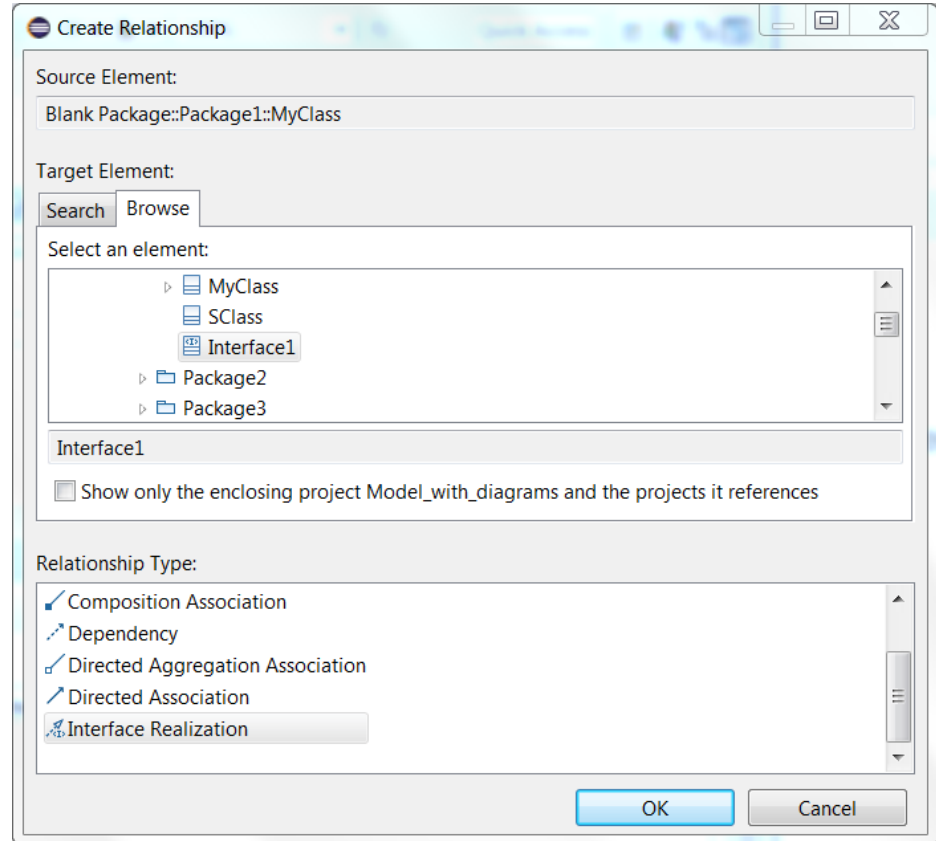
Relationships targeting this element:			
	Name	Type	Other Related Element(s)
↗	(AClass)	<Dependency>	<a href="#">Enumeration1</a>
↗	(AClass)	<Dependency>	<a href="#">Capsule1</a>

Add... Delete Navigate



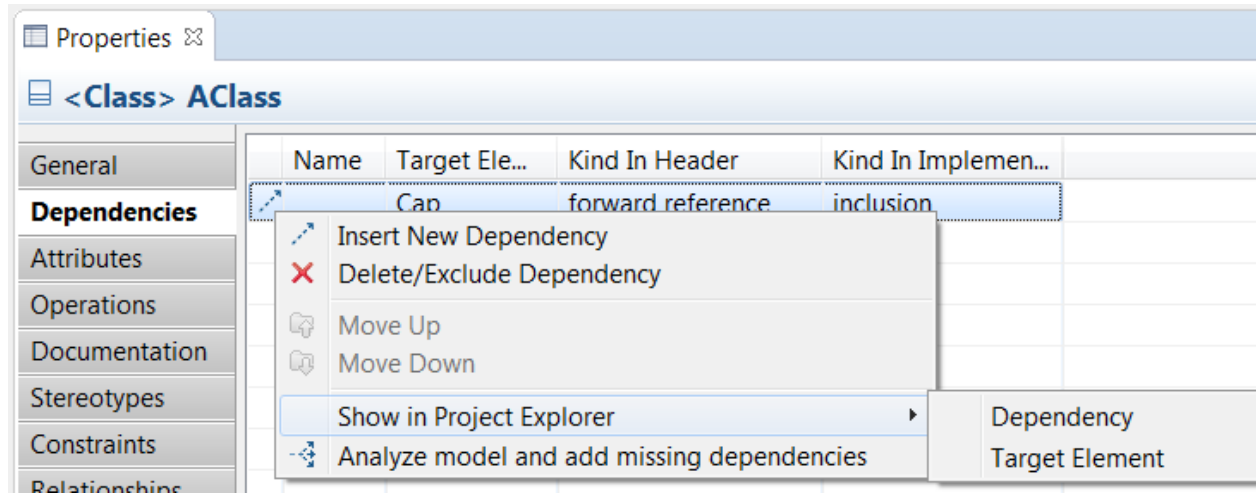
# Improved Create Relationship Dialog

- Relationship types that are not relevant for RSARTE users are now filtered out from this dialog
- This applies when the Capsule Development viewpoint is active



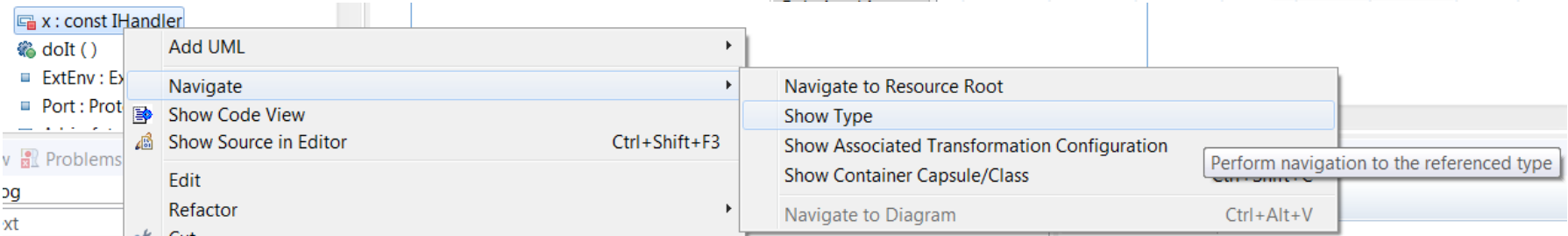
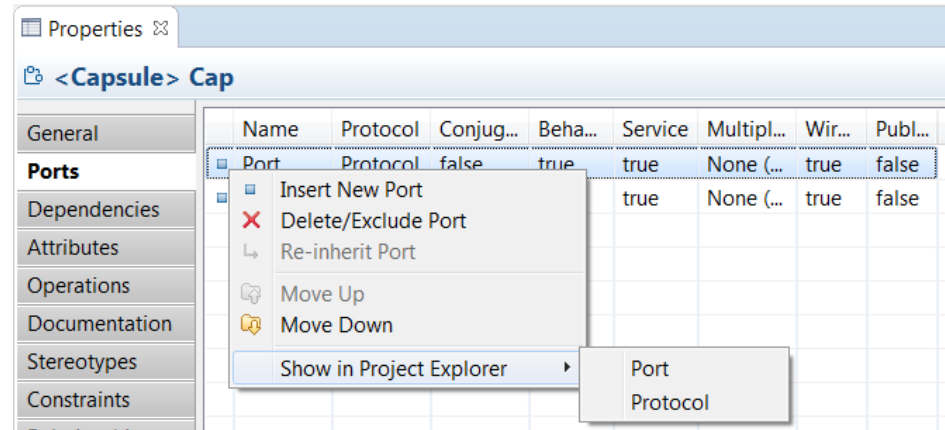
# Navigation from Dependencies

- The Dependencies tab of the Properties view now provides commands for navigating to:
  - the dependency itself
  - the target element of the dependency



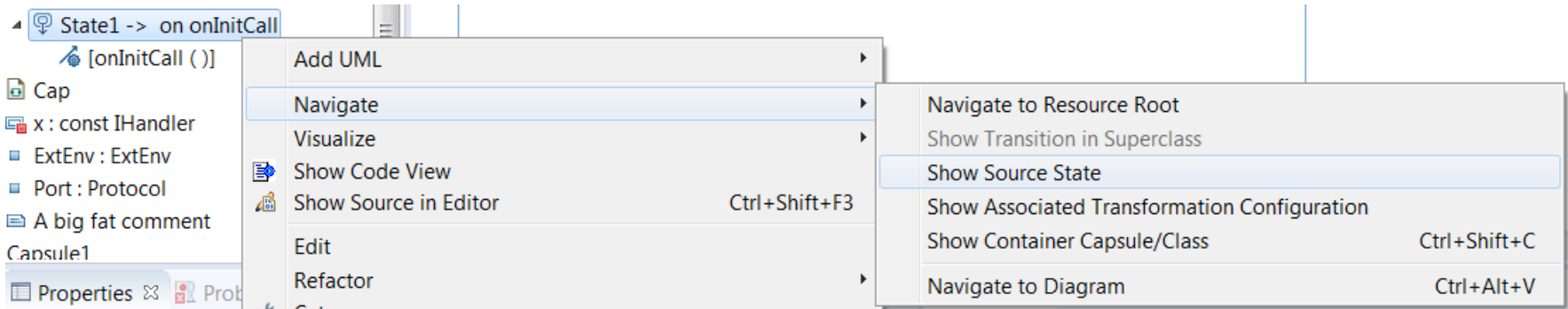
# Navigation from Attributes, Parameters and Ports

- The Attributes, Parameters and Ports tabs of the Properties view now provides commands for navigating to:
  - the attribute, parameter or port itself
  - the type of attributes and parameters, and the protocol of ports
- These commands are also available in the Project Explorer context menu



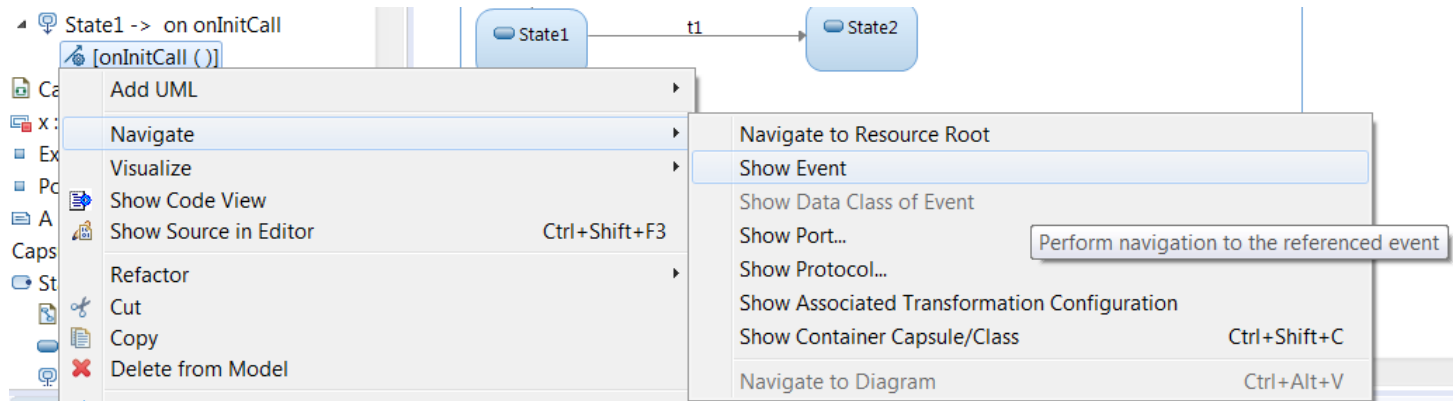
# Navigation to Source State of a Transition

- The Project Explorer now supports navigation from a transition to its source state (i.e. the state from which the transition originates)



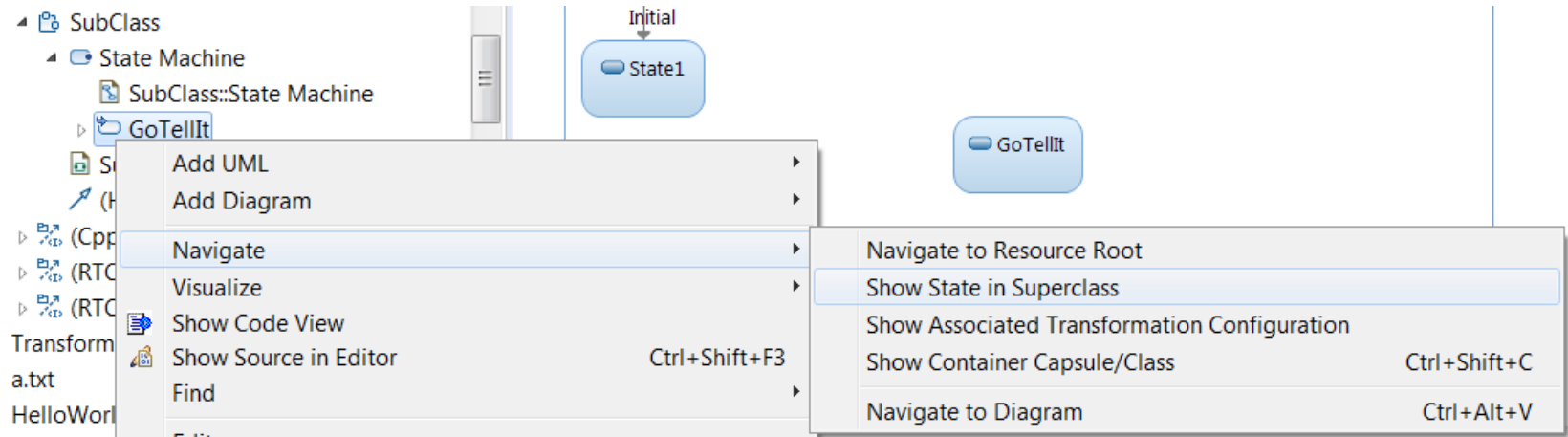
# Navigation from Transition Triggers

- Useful navigation commands have been added for transition triggers shown in the Project Explorer. You can now navigate to:
    - the event
    - the data class of the event (if any)
    - the port
    - the protocol of the port
- For passive class triggers navigation to the trigger operation is provided instead.



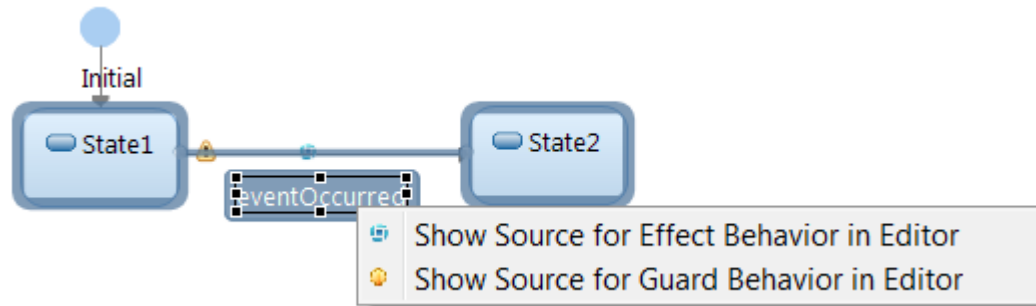
# Navigation from Redefined or Excluded Elements

- Navigation commands are now available for navigating from redefined or excluded states, transitions or ports. They navigate to the corresponding element in the super class (capsule).
- These commands are available both in the Project Explorer and in diagrams.



# Open Code Editor for a Transition

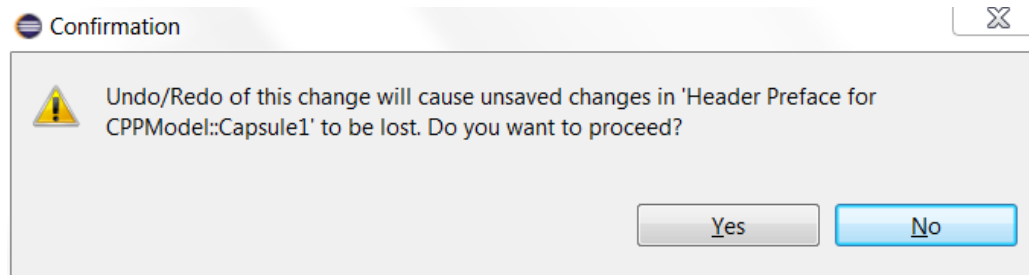
- It's now possible to double-click on the text label for a transition line in order to bring up the pop-up for opening the code editor
- This is sometimes easier than to double-click on the transition line itself (especially if the label was moved a bit away from the line)





# Undo/Redo Improvements

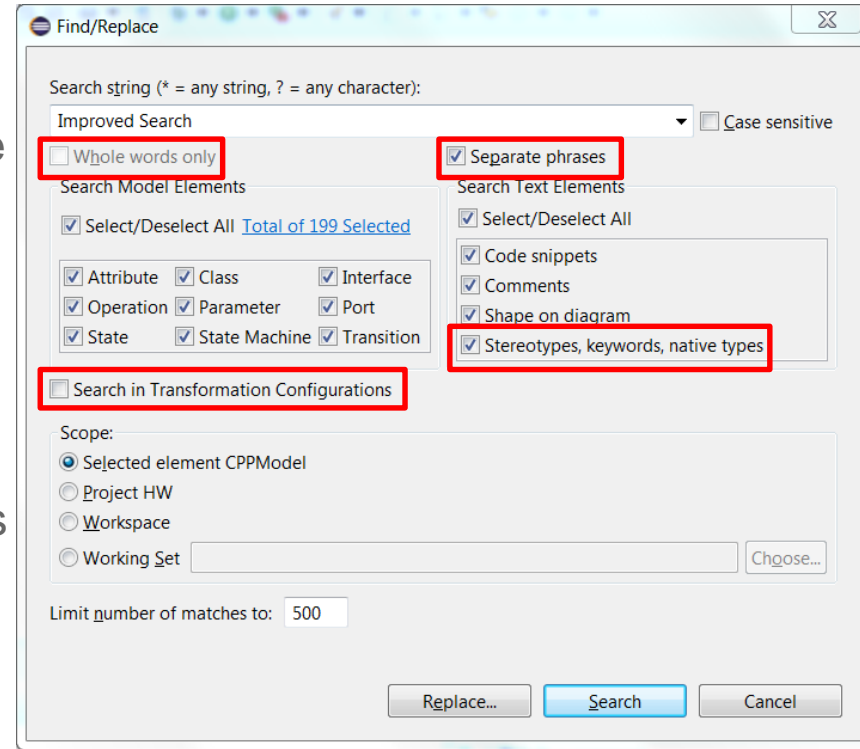
- A confirmation dialog now appears if you perform an Undo/Redo operation that would overwrite unsaved changes in the Code Editor:



# Improved Search Dialogs

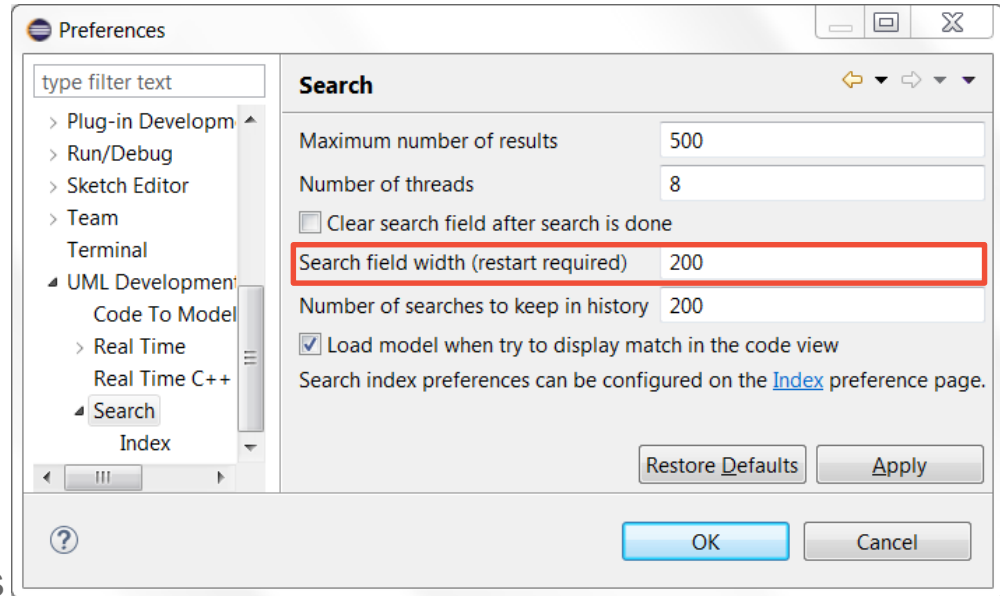
The Find/Replace and Model Search dialogs have several new options for searching to make them equally powerful as the Search field:

- **Whole words only.** When this option is set, the search behaves very similar to how searching in 9.x worked (it had a similar checkbox)
- **Separate phrases.** Search for multiple words separately.
- **Search in Transformation Configurations.**
- **Stereotypes, keywords, native types.**



# Miscellaneous Search Improvements

- New preference for making search field wider
- Search field is no longer disabled while a search is running
  - Makes it possible to start a new search at any time
- Progress bar for Search operations now reports actual progress of the running job so user can estimate completion time for long-running tasks



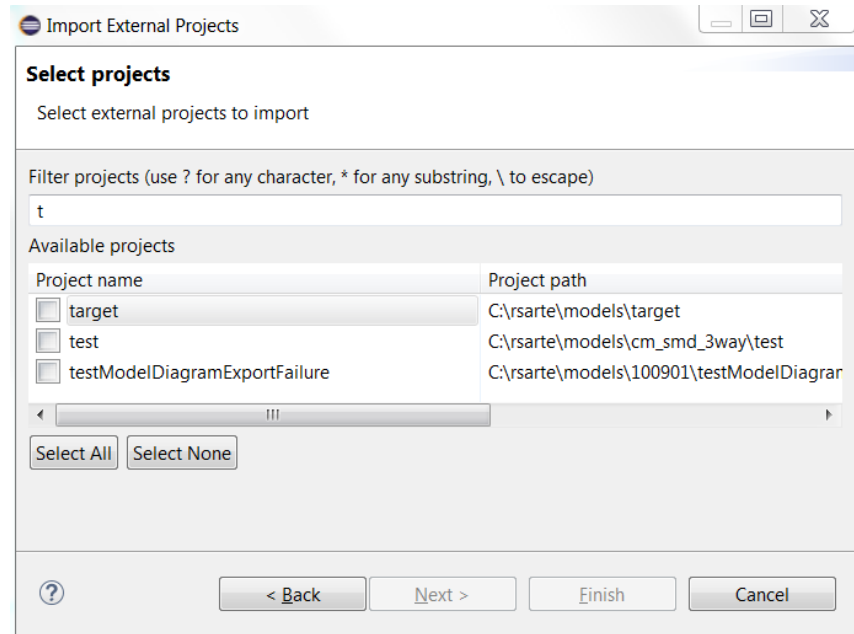
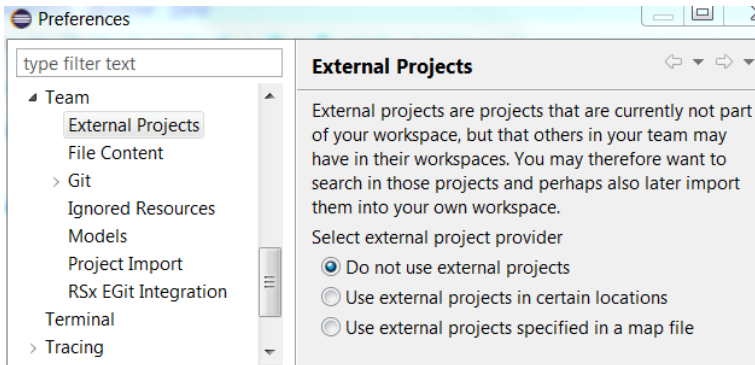
# External Projects

- These are projects that you currently don't have in your workspace, but your team members may have them in their workspaces.
- You can now search in external projects (and import the ones you need from the search result)
- You can also import them directly from a new Import External Projects wizard
- When an external project is imported, dependent projects are automatically imported too
  - Guarantees a consistent workspace for all team members
  - Not necessary for each user to keep project dependencies in mind, when deciding which projects to import



# Import External Projects Wizard

- *Import – Other – External Projects*
- Before using the wizard you must specify where to look for external projects:  
*Preferences – Team – External Projects*
  - Certain locations in the file system, or
  - A map file

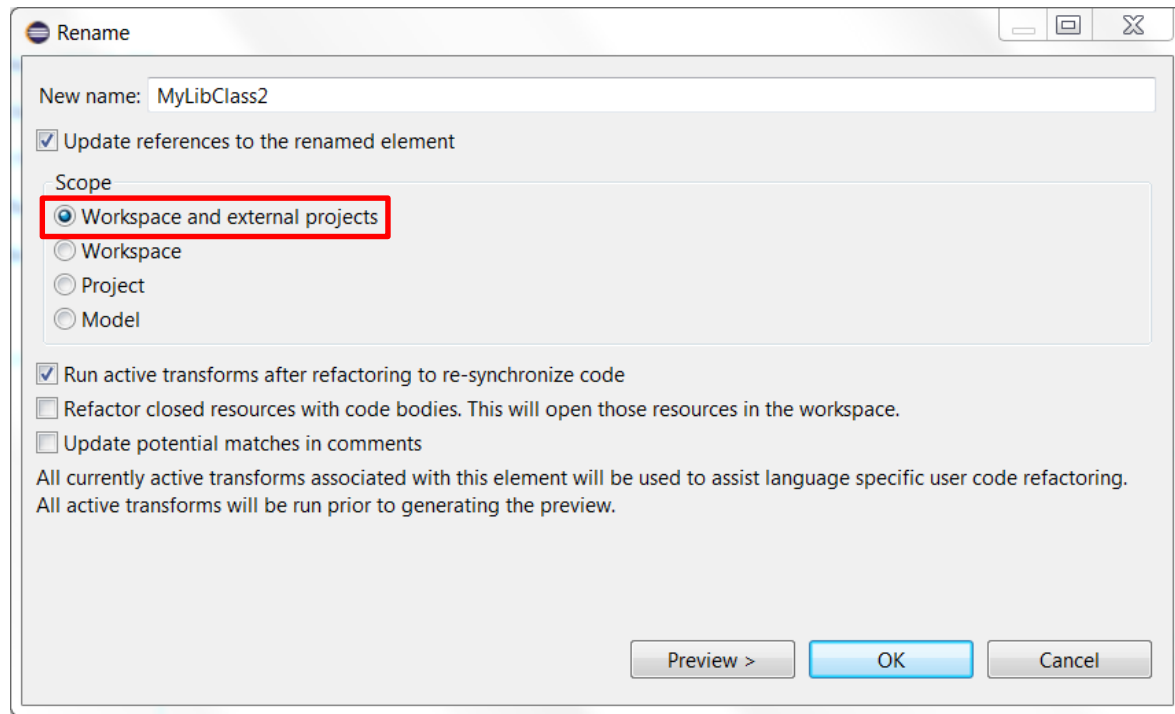


- In case of ambiguities where to find dependent projects, these can be resolved on the second wizard page
- This way of importing projects is more convenient than importing by means of the general Eclipse Project Import wizard (which does not take project dependencies into account)



# Importing External Projects when Refactoring

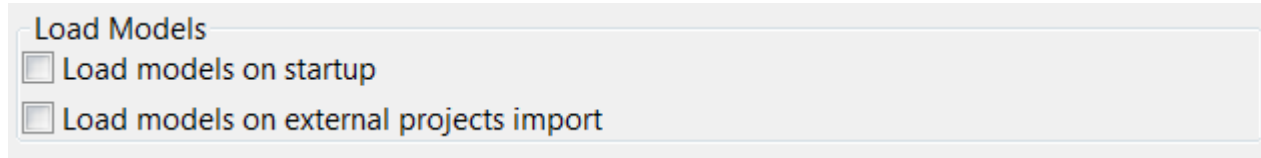
- When performing a refactoring you want to have in your workspace all projects that may be affected by it
  - For example, renaming an element should rename all references to the element. This means that all projects that contain such references must be present in the workspace when running the refactoring.
- It's now possible to automatically import external projects when refactoring
  - References in the refactored model are analyzed to determine which projects that need to be imported



# Loading of Models Imported from External Projects

- It's now possible to automatically load models that are imported from an external project. Set the preference:

*Preferences – UML Development – Load models on external projects import*

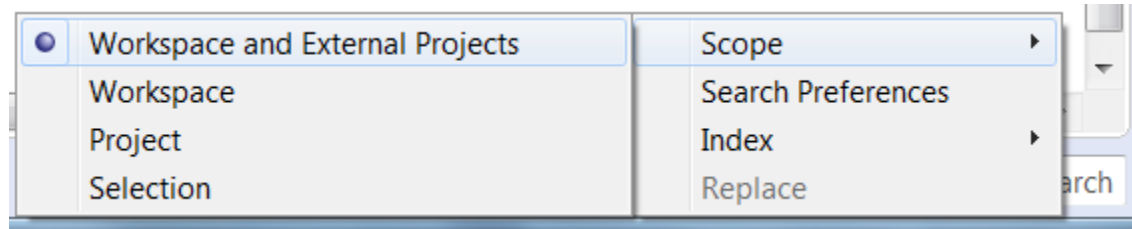


- Note that the preference for automatic loading of models on startup also was moved to this preference page



# New Search Scope for Including External Projects

- A new search scope "Workspace and External Projects" is now available in the Scope context menu of the Search field



- This makes it possible to decide whether to search in external projects or not, without having to disable the External Projects preference





# Load UML Models

- This command was changed to always load all models in the workspace
  - Users rarely used the possibility to only load some of the models
  - Populating the "Load UML Models" dialog could take a significant amount of time in large workspaces
- In the following API the 'prompt' parameter is now ignored (but kept to avoid API incompatibility):

`com.ibm.xtools.modeler.ui.internal.ui.actions.LoadModelsActionDelegate.loadUMLModels(boolean prompt)`

Additional API methods were added and are now recommended to be used instead:

`loadUMLModels()`

`loadUMLModels(List<IFile>)`

`loadUMLModels(List<IFile>, Consumer<Boolean>)`

*Load everything*

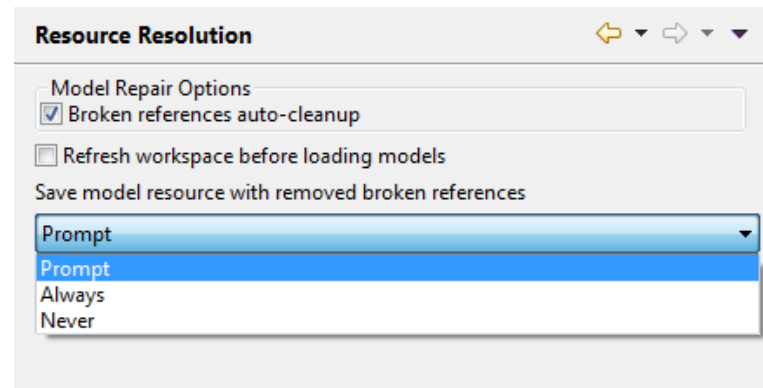
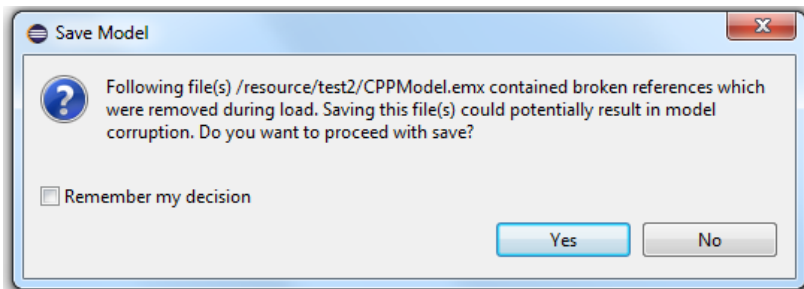
*Load specified model files*

*Load specified model files with callback when done*



# Preventing Model Corruption

- During model loading, certain kind of broken references could be deleted (for example, ones that define redefinitions).
  - Each deletion is marked with the message in “Rational Modeling” console. If users modifies such model and then tries to save it, it could lead to the corruption.
- To prevent saving models with removed broken references, warning message dialog appears now for each model file during save.



- To control action on save, new preference “Save model resource with removed broken references” is introduced at the *UML Development - Real Time - Resource Resolution* page.



# Model Compiler

- Use of the Model Compiler is now the recommended way to build models into C++ applications
  - No longer necessary to have a Display when running command-line builds
- Most features from the classic builder are now supported
  - Inheritance of TC prerequisites
  - "Save before build"
  - Generation of makefiles for external library TCs
  - Only generate code without building it

*N.B. Some limitations in the Model Compiler still exists (e.g. support for C code), so if necessary you can still use the classic builder*
- Makefiles can now be generated with a single rule that will perform all transformations in one step.
  - This can be useful if your build environment does not support parallel processing of make rules, but you still want to drive the entire build from make

Generated make file

Include make rules for model-to-code transformation

Separate. One make rule for each source file.

Joint. One make rule for all source files corresponding to the same model file.

Full. One make rule for all sources.



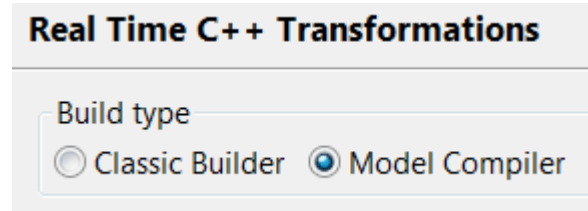
# Command-Line Usage of the Model Compiler

- The model compiler requires a license (one per build)
  - Use the command-line option `--license` to specify the location of the license file or server
  - Alternatively, the environment variable `RSARTE_MODEL_COMPILER_LICENSE` can be used instead
- Generating build information for a TC
  - Use the command-line options `--generate=makefile --genBuildInfoRules --build=build_info`
  - A special make target ("build\_info") will be built which will print useful information about the build, such as the compiler and linker commands that will be used, full paths to source files, etc.
  - The build information file is generated in the same location as the makefile and has the suffix "build\_info.js".



# Model Compiler Preferences

- The "Real Time C++ Transformations" preference page allows you to choose if you want to use the model compiler or the old C++ code generator ("Classic Builder") for building your model.



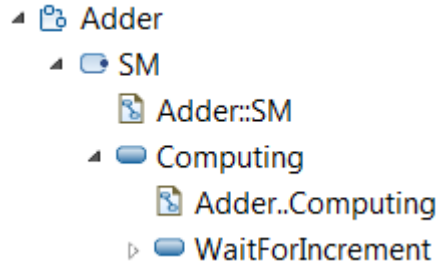
The image shows a screenshot of a software preference dialog box titled "Real Time C++ Transformations". Below the title bar, there is a section labeled "Build type" which contains two radio button options: "Classic Builder" and "Model Compiler". The "Model Compiler" option is selected, indicated by a blue dot inside its radio button.

- Depending on your choice the applicable preferences will be shown below  
The model compiler supports an extended subset of the preferences supported by the classic builder.



# Improved Readability of Generated C++ Code

- C++ code generated by the model compiler now contains comments for states
  - State name (fully qualified name within parenthesis)
  - Generated both for capsules and passive class state machines
  - Helps when debugging generated C++ code



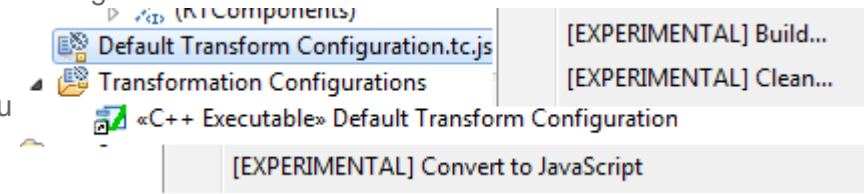
```
}
break;
case 4 /* WaitForIncrement (SM::Computing::WaitForIncrement) */:
    switch( portIndex )
    {
    case 0 /*RTControlPort*/:
        switch( signalIndex )
        {
        case 1 /*RTInitSignal*/:
            return ;
        }
    }
}
```



# Improved TC File Format

- A new TC file format based on JavaScript is now supported by RSARTE

- Files with the extension .tc.js are considered to be Transformation Configurations
- .tc.js files are visible in the Project Explorer and have TC icon
- .tc.js files can be viewed by Eclipse Java Script or external editor
- Command "Convert to Java Script" is available in the context menu of the Project Explorer on the .tc files
- It is possible to build and clean .tc.js files using corresponding context menu command on the .tc.js files. To have these commands visible, Model Compiler should be selected as an active builder
- TC files in the new format can be set as Active TCs



- Support of a new TC file format for the command line Model Compiler tool

- You can convert an existing TC with the following command:  
`java <JVM options> -jar modelcompiler.jar --xml2js --root <.map> <files>`

- The new format is easy for humans to read and write and allows TCs to become more dynamic through the use of JavaScript statements.

- This will make it possible to build multiple variants of a single TC without having to create physical TC files for all variants.

- A JavaScript API is now available for working with TCs programmatically



# Product News Shown on Welcome Page

- There is a newsletter for RSARTE with new posts approximately every second week
- Now you can view the latest news about the product on the Welcome page (in the *What's New* section)
- There is also a link for subscribing to the newsletter (by sending a mail to [rsarte@se.ibm.com](mailto:rsarte@se.ibm.com))



## UML Real Time Development

Learn about what is new in this release of Rational Software Architect RealTime Edition.

### Latest News ▾



**Merge models in the most efficient way**



**On the way to C++ simplification**



**Do you know about navigation history of the Project Explorer**



**Have you noticed the New Search**



**RSARTE 10.1 for Eclipse Neon and CDT 9.1**

**Subscribe** to the RSARTE newsletter





THANK YOU!