News in RSA-RTE 10.1

updated for sprint 2017.07

Mattias Mohlin, February 2017

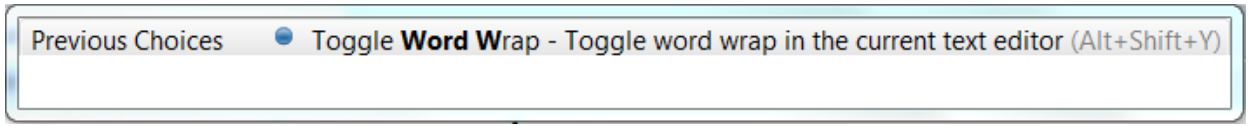
IBM

Overview

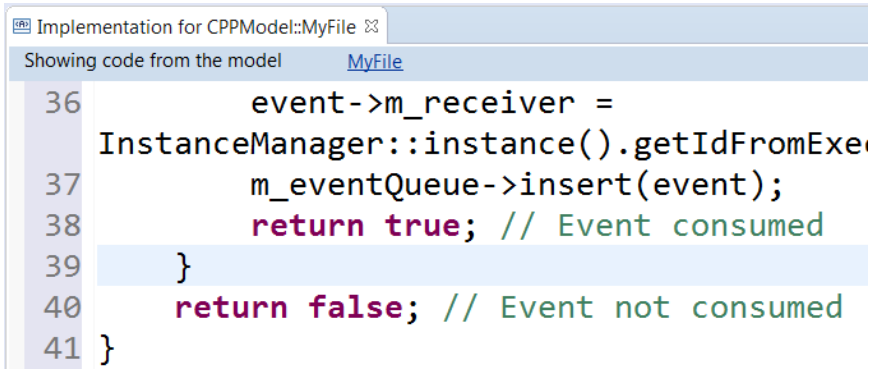
- **Now based on Eclipse Neon-1 (4.6.1)**
  - Many general improvements since Eclipse Mars
  - Note: Neon-2 (4.6.2) is not yet supported!
- **Contains everything from RSARTE 10 and also additional features and bug fixes**
  - See the What's New presentation for RSARTE 10 to learn about new features

Eclipse 4.6.1 (Neon)

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

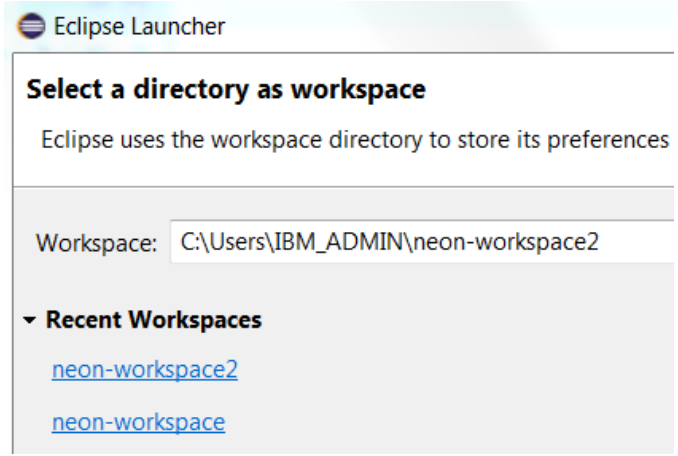


- Commands for ”zooming” in text editors by changing the font size
  - Ctrl++ for zooming in and Ctrl+- for zooming out



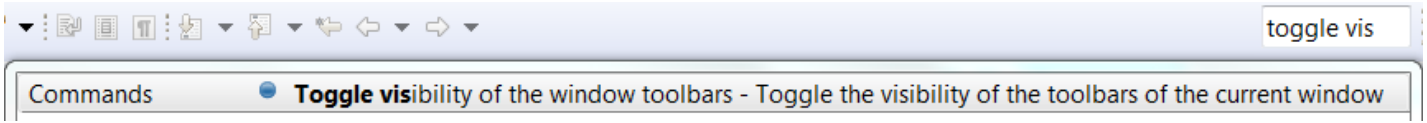
Eclipse 4.6.1 (Neon)

- 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

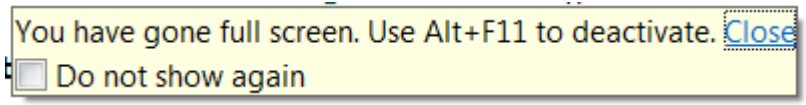


Eclipse 4.6.1 (Neon)

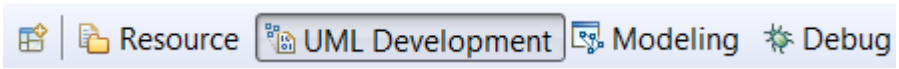
- Command for toggling visibility of window toolbars (to maximize space for editors and views)
  - Assign a key-binding to this command, or access it through Quick Access



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

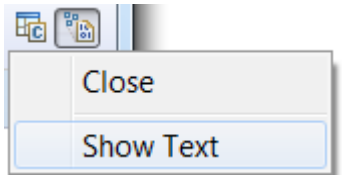


- Perspective names hidden by default (to save space in toolbar)

before 

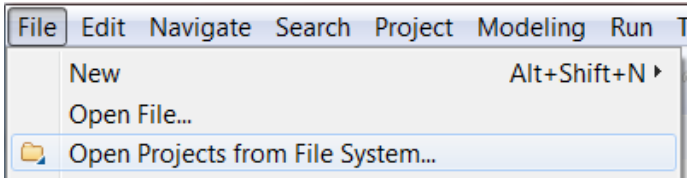
now 

The names can be shown using the context menu



Eclipse 4.6.1 (Neon)

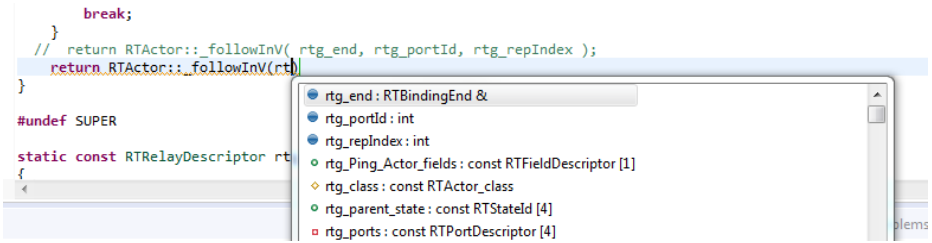
- New smart wizard for importing projects
  - No longer necessary to use different wizards for different kinds of Eclipse projects

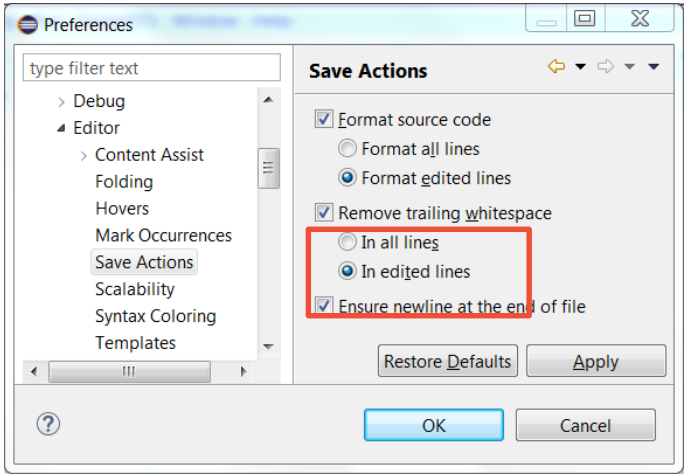


- For more information about Eclipse improvements see
  - News in Eclipse 4.6.1 (Neon) https://www.eclipse.org/eclipse/news/4.6/

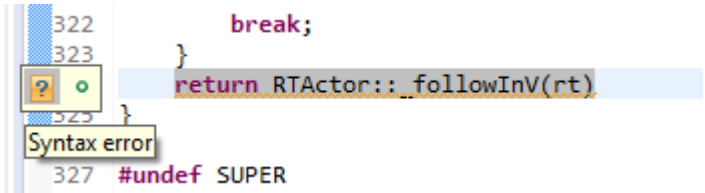
CDT 9.1 (included as part of Eclipse Neon.1)

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



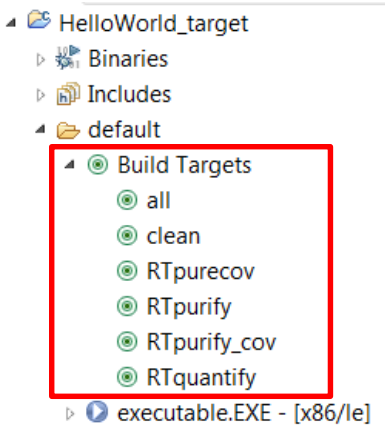


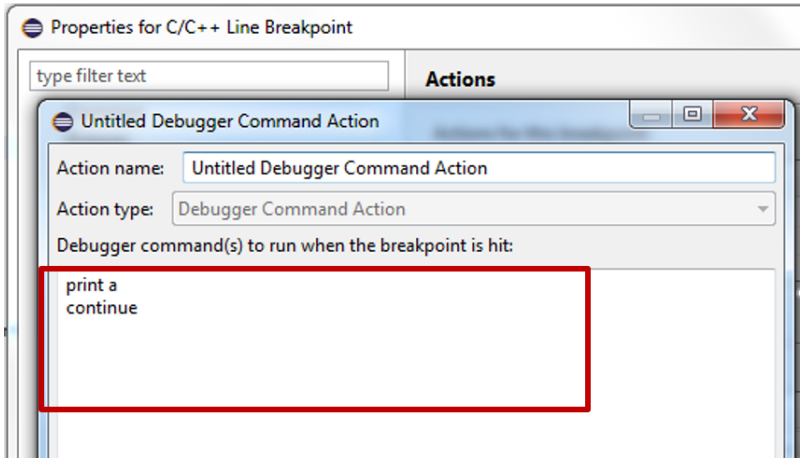
- Support for **decltype**(**auto**) type-specifiers
- Expansion of icons in the editor ruler
  - Helps when there are multiple icons on the same source code line



CDT 9.1

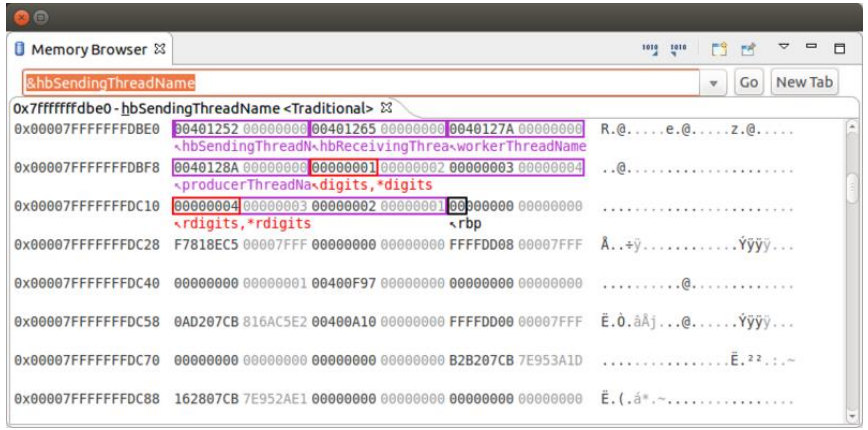
- Command for commenting/uncommenting selected lines in makefile editor (Ctrl + /)
- Build targets (previously called Make targets) now show up in the Project Explorer. They can be run by double-click.
- Support for running commands in the debugger when a breakpoint is hit

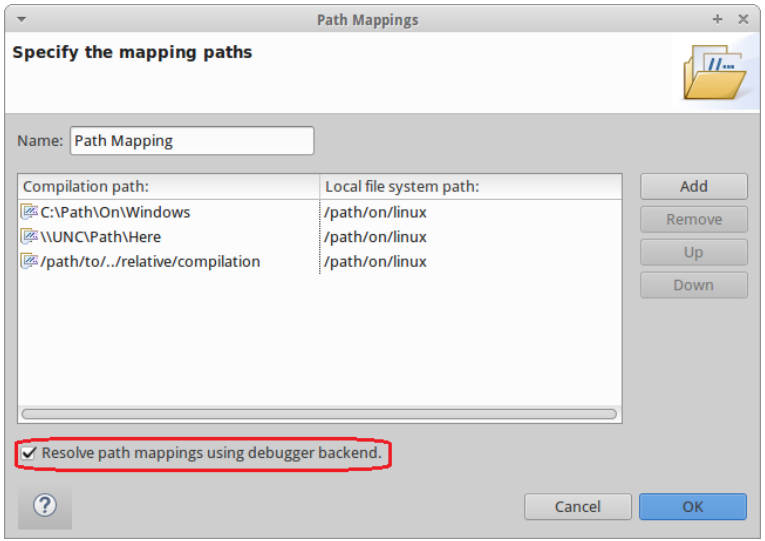




CDT 9.1

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





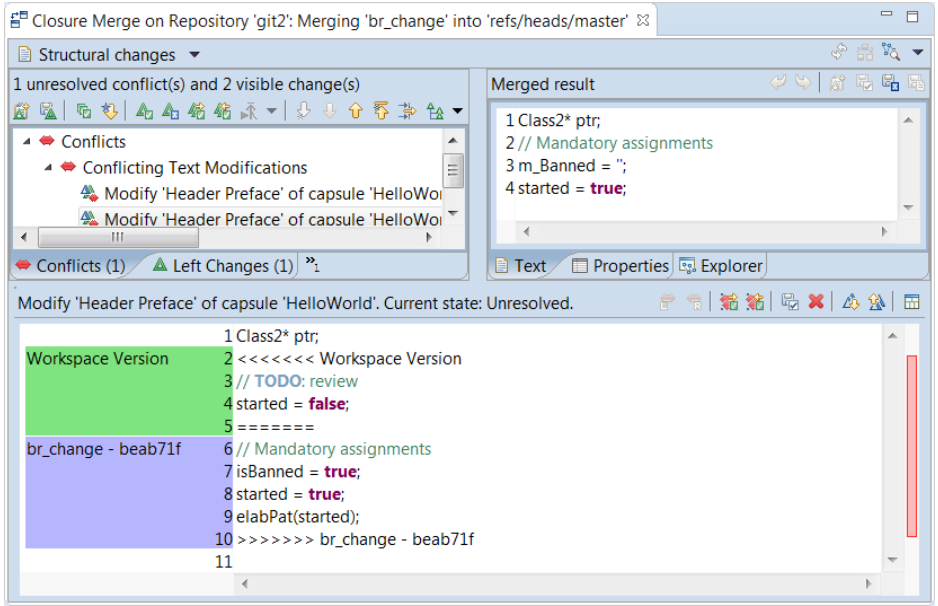
For more information about news in CDT 9.1 see https://wiki.eclipse.org/CDT/User/NewIn90 and https://wiki.eclipse.org/CDT/User/NewIn91

Note about Codan

- Codan is the Code Analysis feature of CDT. It has been improved in CDT 9.1.
- However, it cannot correctly analyze the CDT projects generated by RSARTE and also currently interfers with some RSARTE UI features
  - You may notice unexpected behaviors if working for example in the Code Editor or Code View when Codan is enabled
- For now, it is therefore recommended to disable Codan
  - Unmark all checkboxes in *Preferences – C/C++ - Code Analysis*

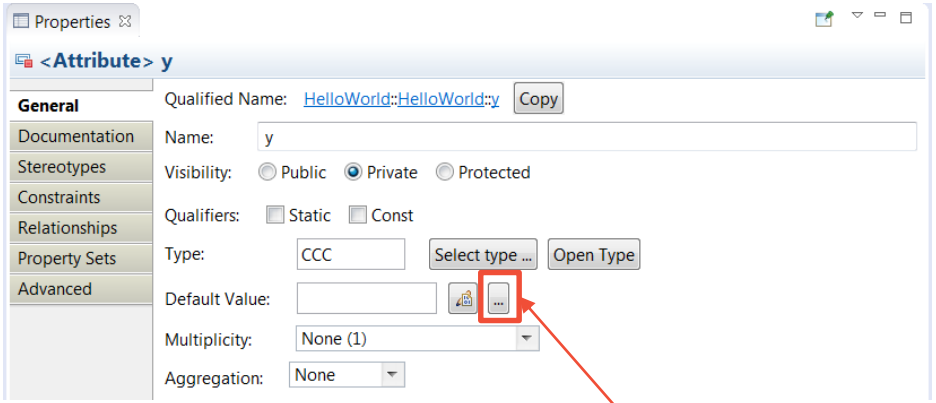
Improved Text Merge

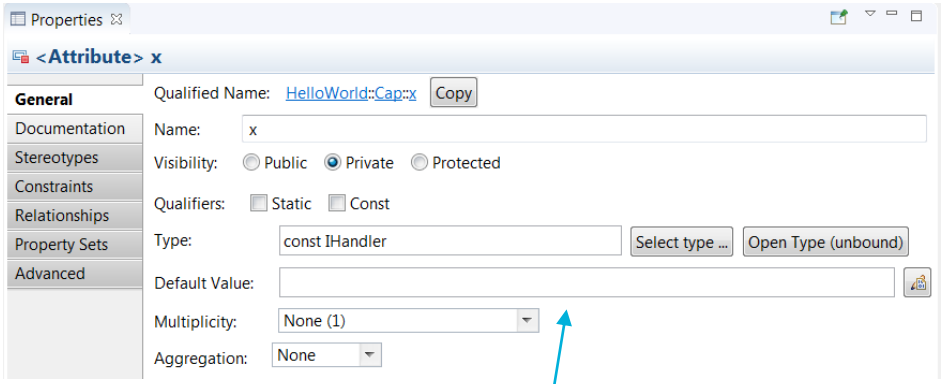
- Merging text (code, comments etc.) is now easier thanks to a new single-view text merge editor
  - Replaces the old ”sub-merge” (except when merging rich text)
  - Conflicts can be resolved by choosing the contributor versions, but also by direct editing
  - Non-conflicting text blocks are merged automatically but can also be edited if needed
  - The editor has line number and a Find command
  - The ”Merged result” view is updated when saving or when the text merge editor looses focus
- This is currently an experimental feature
  - Enable it in *Preferences – General – Compare/Patch – Modeling Compare/Patch – UML Compare/Mer*ge
  - There you can also customize the colors used



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

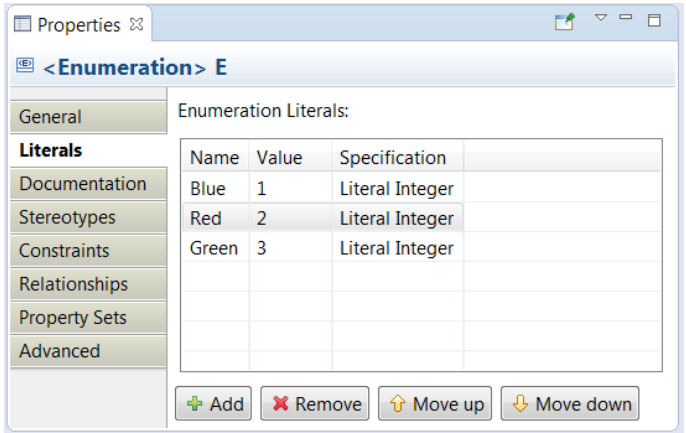
now

Removed useless Browse button for default values

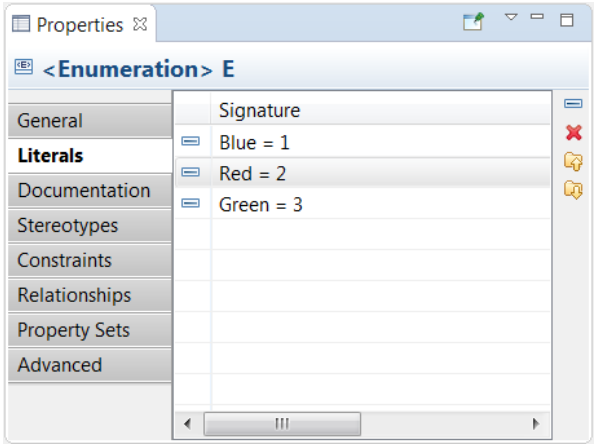
More space for editing type and default value of an attribute

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



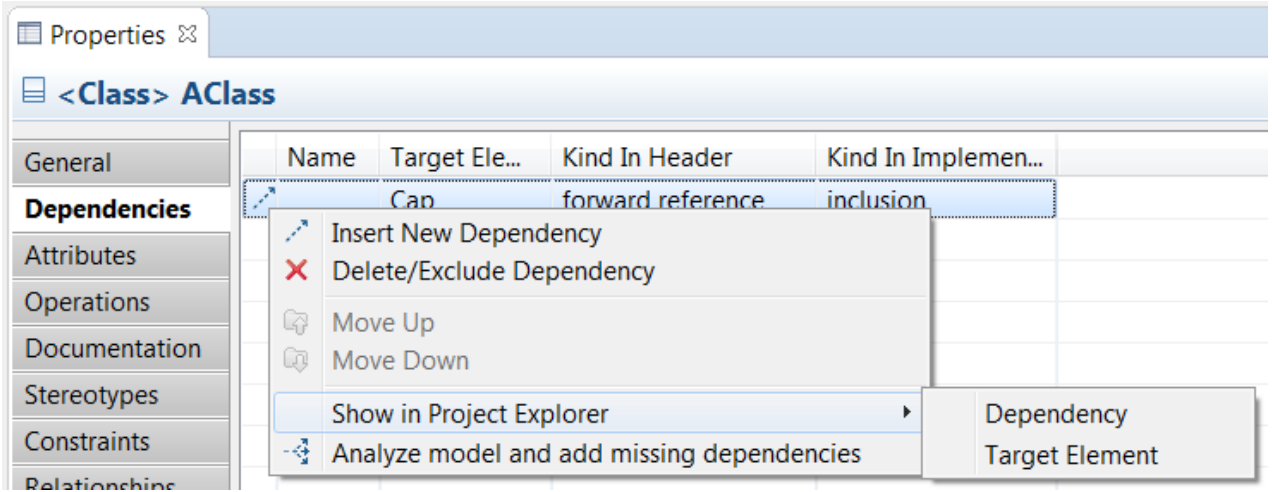
before



now

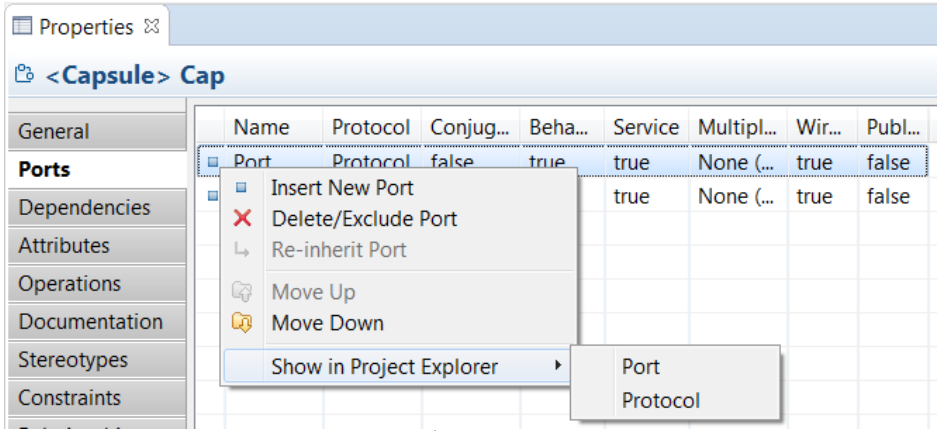
Navigation from Dependencies

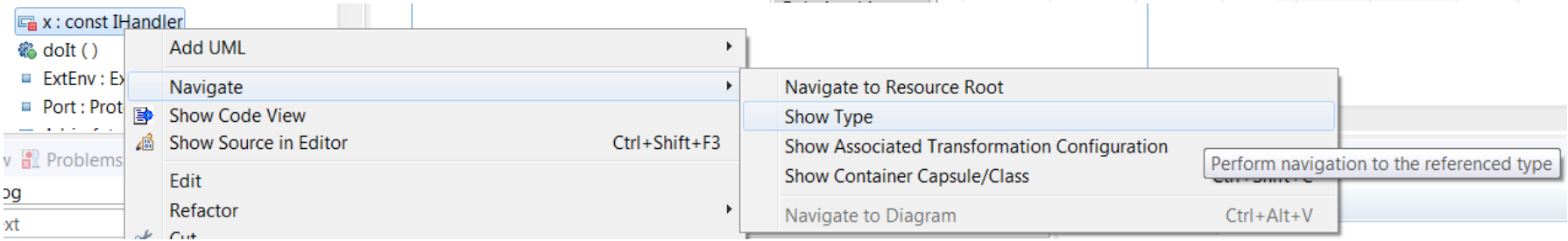
- The Dependencies tab of the Properties view now provides commands for navigating to:
  - the dependency itself
  - the target element of the dependency



Navigation from Attributes, Parameters and Ports

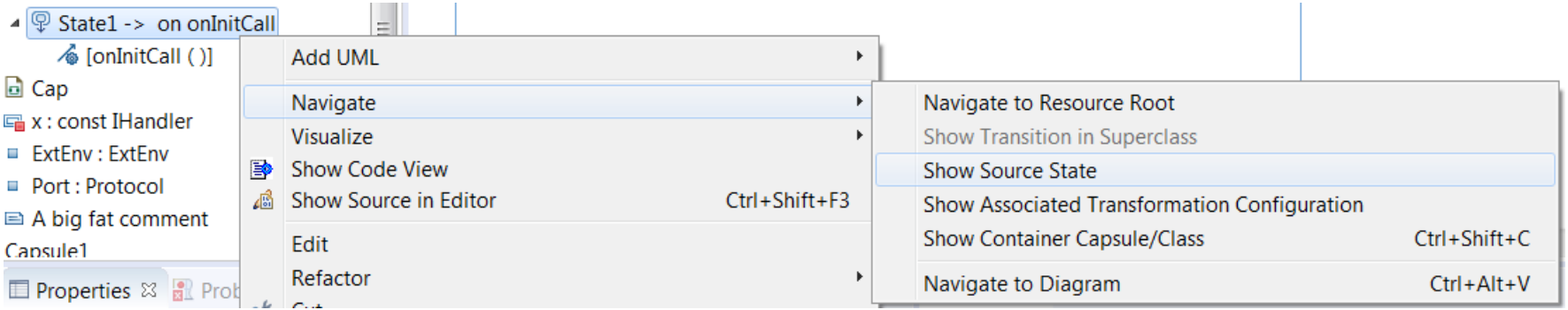
- The Attributes, Parameters and Ports tabs of the Properties view now provides commands for navigating to:
  - the attribute, parameter or port itself
  - the type of attributes and parameters, and the protocol of ports
- These commands are also available in the Project Explorer context menu





Navigation to Source State of a Transition

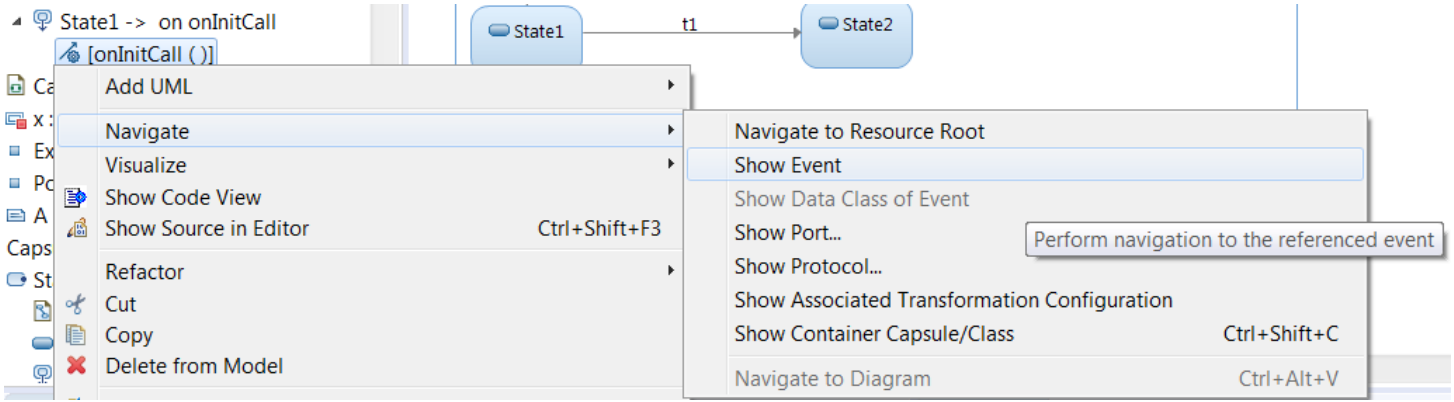
- The Project Explorer now supports navigation from a transition to its source state (i.e. the state from which the transition originates)



Navigation from Transition Triggers

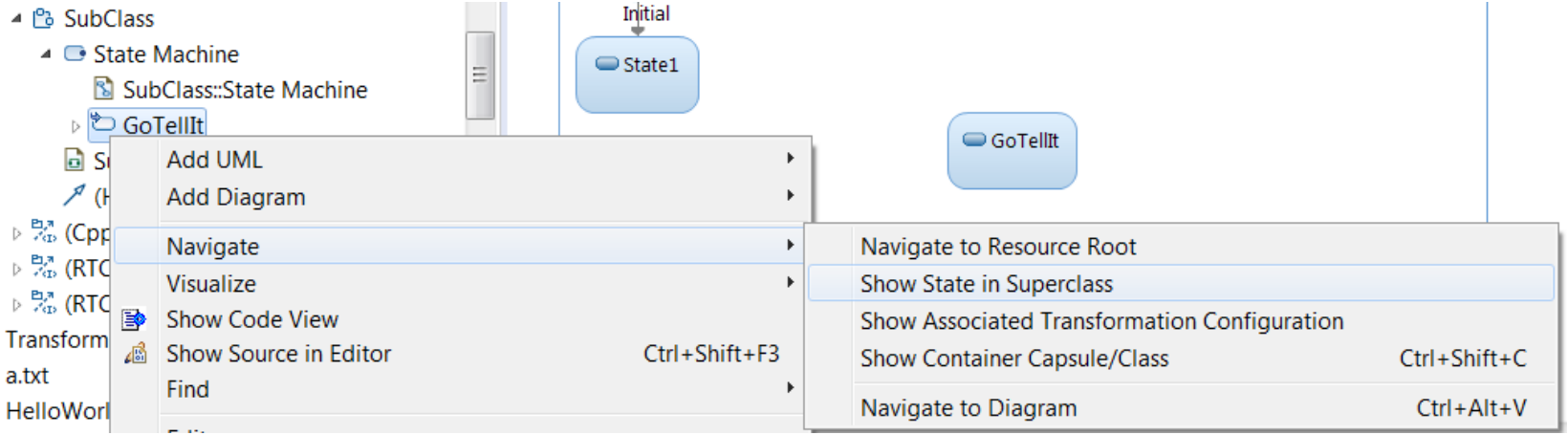
- Useful navigation commands have been added for transition triggers shown in the Project Explorer. You can now navigate to:
  - the event
  - the data class of the event (if any)
  - the port
  - the protocol of the port

  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.

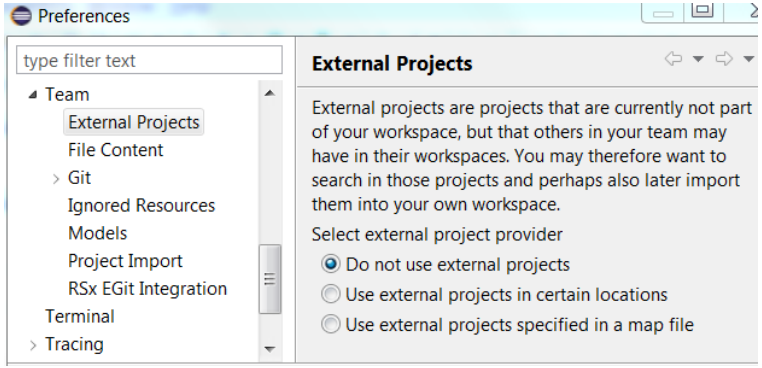


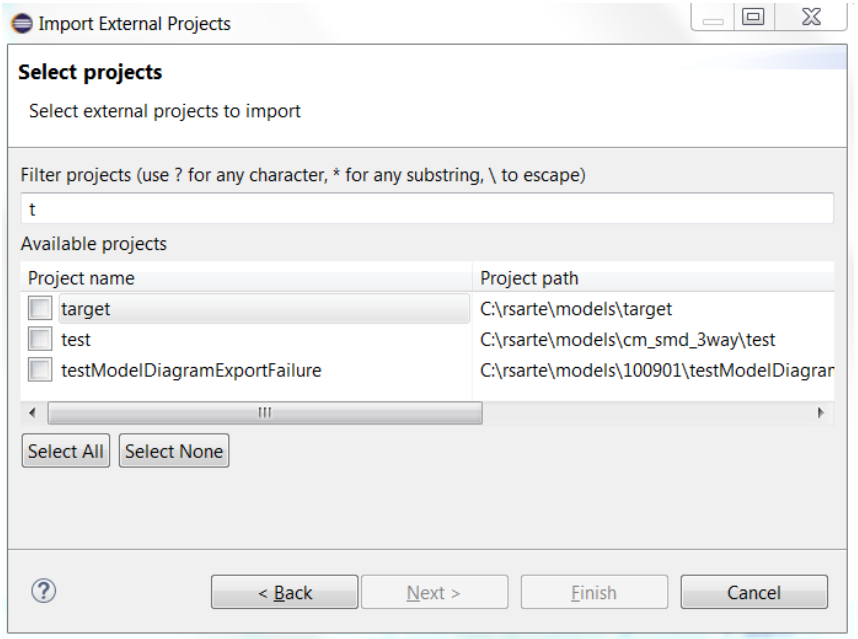
External Projects

- 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

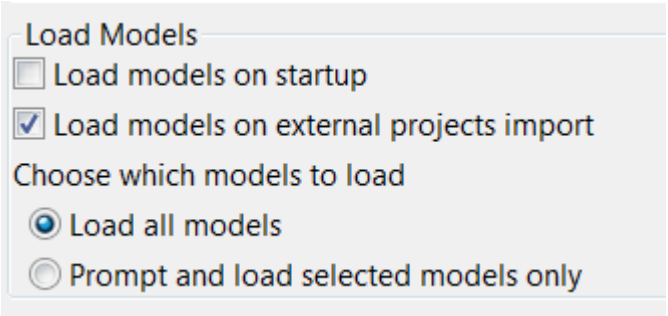




- In case of ambiguities where to find dependent projects, these can be resolved on the second wizard page
- This way of importing projects is more convenient than importing by means of the general Eclipse Project Import wizard (which does not take project dependencies into account)

Loading of Models Imported from External Projects

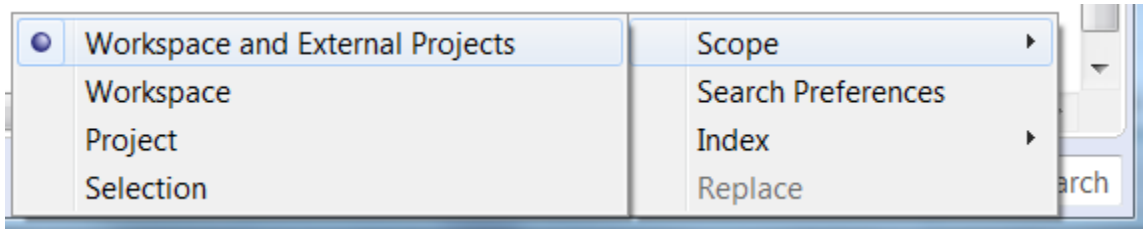
- It's now possible to automatically load models that are imported from an external project. Set the preference:
  Preferences – UML Development – Load models on external projects import



- 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

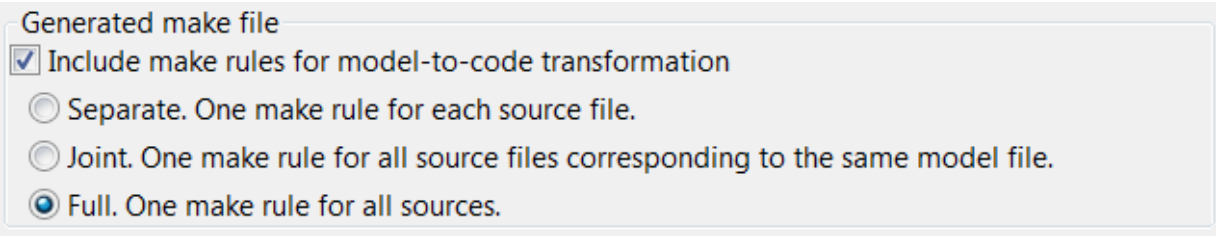
- A new search scope ”Workspace and External Projects” is now available in the Scope context menu of the Search field



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

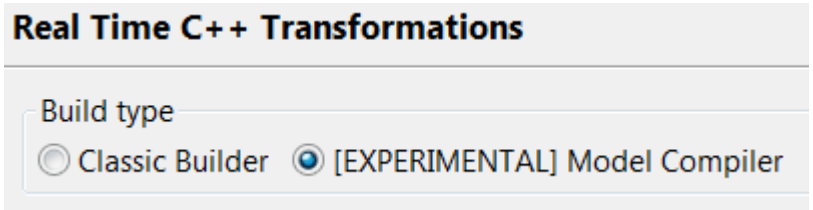
Model Compiler

- The Model Compiler user interface is still experimental, but the feature now has production quality for command-line usage
  - No longer necessary to have a Display when running command-line builds
- Inheritance of TC prerequisites is now supported
- ”Save before build” is now supported
- 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



Model Compiler Preferences

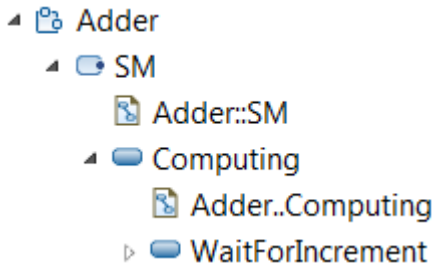
- The ”Real Time C++ Transformations” preference page allows you to choose if you want to use the model compiler or the old C++ code generator (”classic builder”) for building your model.



- 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 code generator.

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 ;
```

THANK YOU!