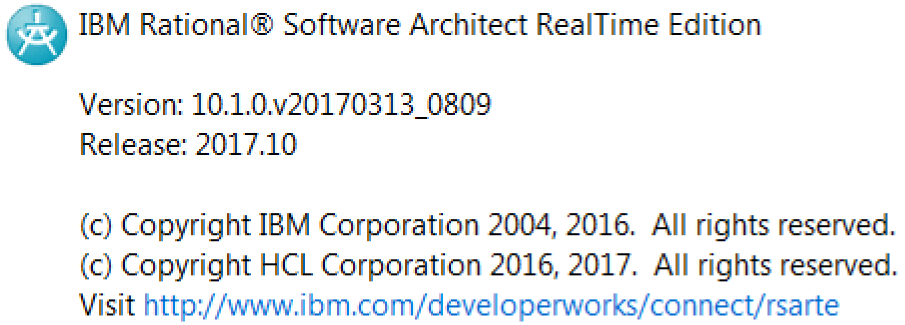
# News in RSA-RTE 10.1

updated for sprint 2017.10

*Mattias Mohlin, March 2017*

## 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

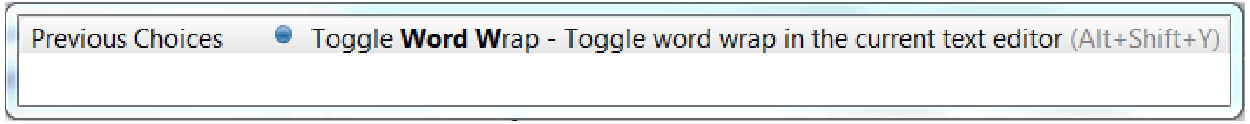

IBM Rational® Software Architect RealTime Edition

Version: 10.1.0.v20170313_0809
Release: 2017.10

(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

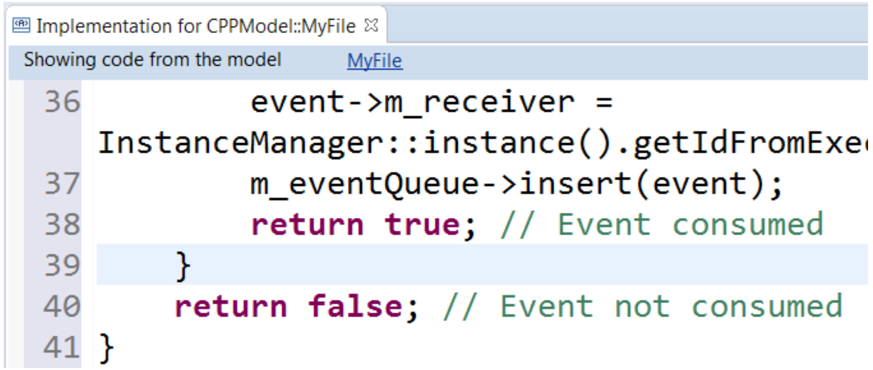
# Eclipse 4.6.1 (Neon)

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



Previous Choices ● Toggle **Word W**rap - Toggle word wrap in the current text editor (Alt+Shift+Y)

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

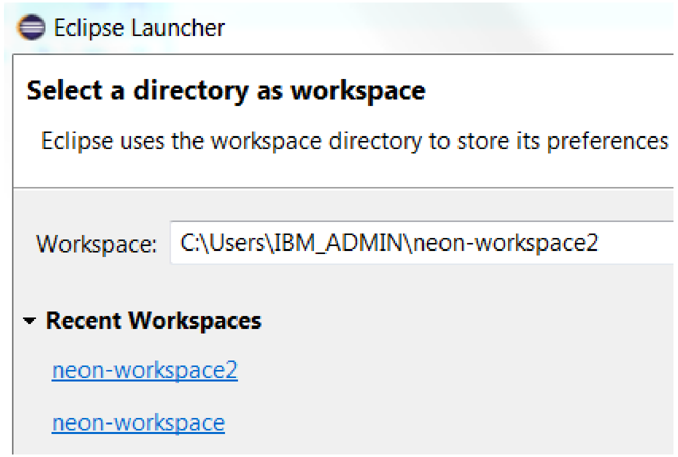


Implementation for CPPModel::MyFile
Showing code from the model
MyFile

```
        event->m_receiver =
InstanceManager::instance().getIdFromExe
        m_eventQueue->insert(event);
        return true; // Event consumed
    }
    return false; // Event not consumed
}
```

# 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 Launcher

**Select a directory as workspace**

Eclipse uses the workspace directory to store its preferences

Workspace: C:\Users\IBM_ADMIN\neon-workspace2

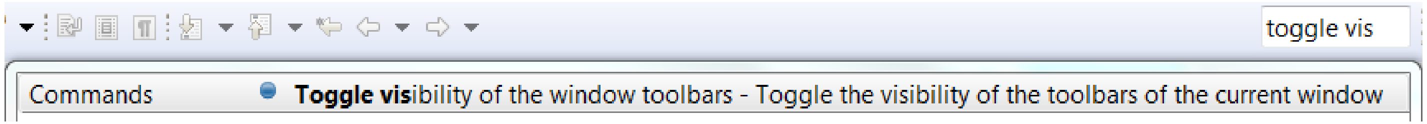
**Recent Workspaces**

neon-workspace2

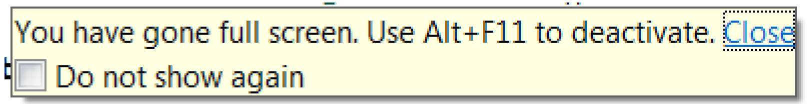
neon-workspace

# 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

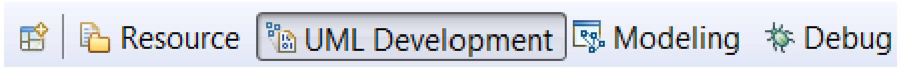

toggle vis
Commands
Toggle visibility of the window toolbars - Toggle the visibility of the toolbars of the current window

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

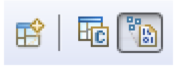


- Perspective names hidden by default (to save space in toolbar)

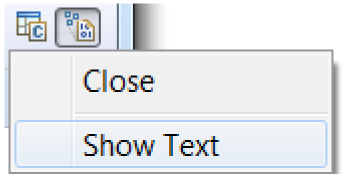
*before*



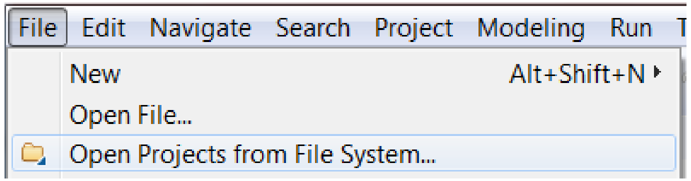
*now*



The names can be shown using the context menu



# Eclipse 4.6.1 (Neon)

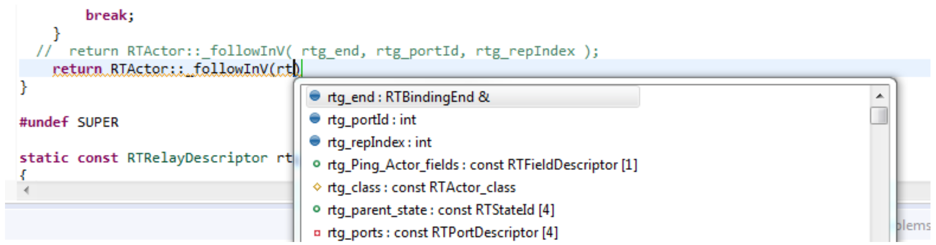


- New smart wizard for importing projects
  - No longer necessary to use different wizards for different kinds of Eclipse projects

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

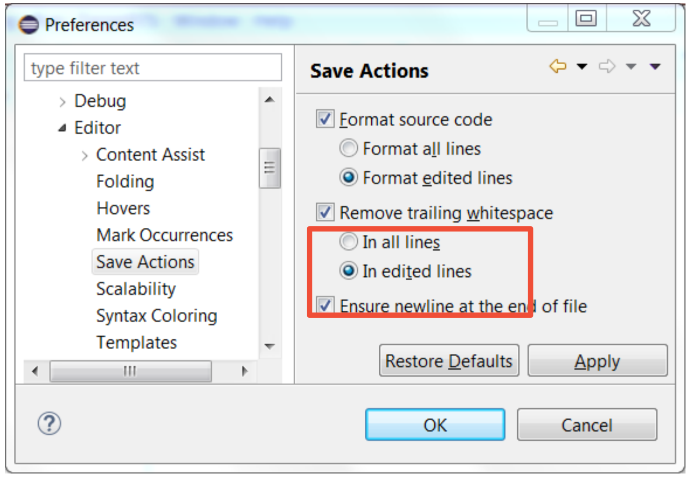
# CDT 9.1 (included as part of Eclipse Neon.1)

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



```
            break;
        }
    //  return RTActor::_followInV( rtg_end, rtg_portId, rtg_repIndex );
        return RTActor::_followInV(rt
}

#undef SUPER

static const RTRelayDescriptor rt
{
```

- rtg_end : RTBindingEnd &
- rtg_portId : int
- rtg_repIndex : int
- rtg_Ping_Actor_fields : const RTFieldDescriptor [1]
- rtg_class : const RTActor_class
- rtg_parent_state : const RTStateId [4]
- rtg_ports : const RTPortDescriptor [4]



Preferences

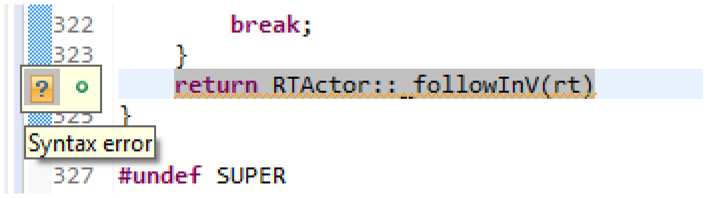
type filter text

- Debug
- Editor
  - Content Assist
  - Folding
  - Hovers
  - Mark Occurrences
  - Save Actions
  - Scalability
  - Syntax Coloring
  - Templates

**Save Actions**

- [✓] Format source code
  - ( ) Format all lines
  - (●) Format edited lines
- [✓] Remove trailing whitespace
  - ( ) In all lines
  - (●) In edited lines
- [✓] Ensure newline at the end of file

- 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

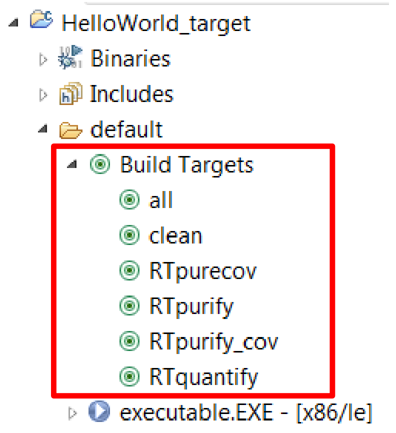


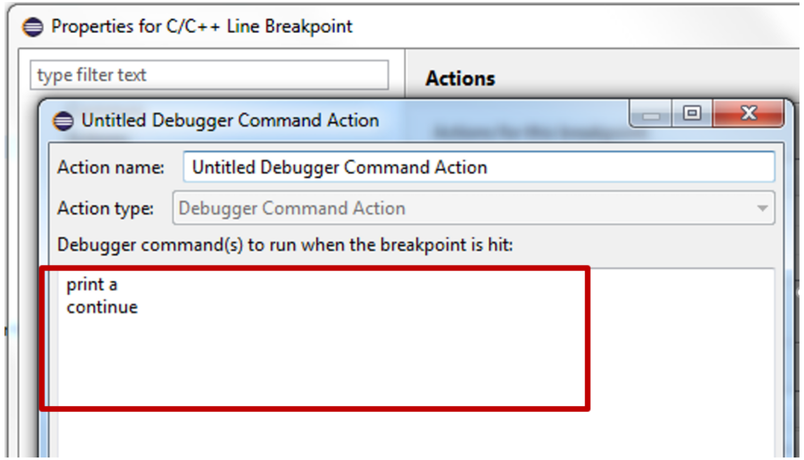
```
        break;
    }
    return RTActor::_followInV(rt)
}

#undef SUPER
```

Syntax error

# 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


HelloWorld_target
Binaries
Includes
default
Build Targets
all
clean
RTpurecov
RTpurify
RTpurify_cov
RTquantify
executable.EXE - [x86/le]


Properties for C/C++ Line Breakpoint

type filter text

Actions

Untitled Debugger Command Action

Action name: Untitled Debugger Command Action

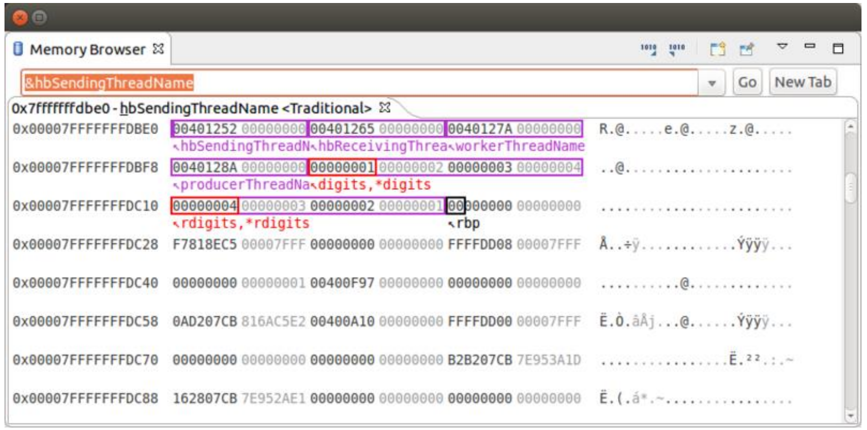
Action type: Debugger Command Action

Debugger command(s) to run when the breakpoint is hit:

```
print a
continue
```

# CDT 9.1

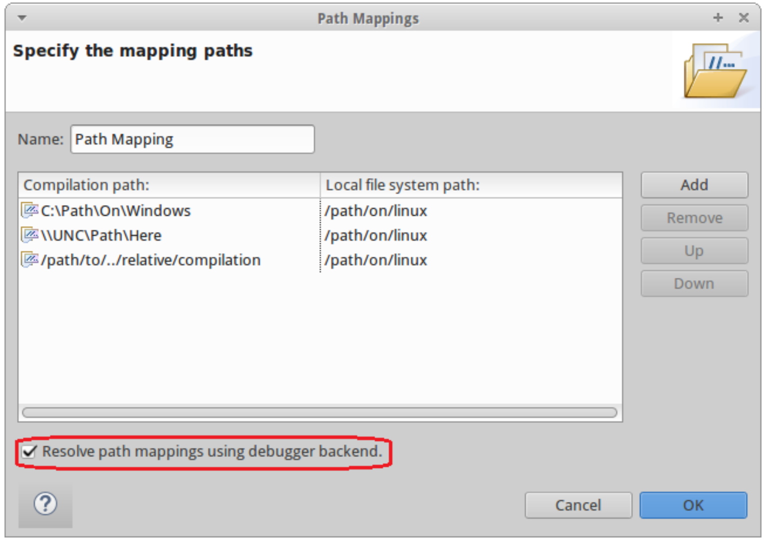
- Local variables and registers in the Memory Browser
- Improved source lookup when debugging



&hbSendingThreadName

0x7fffffffdbe0 - hbSendingThreadName <Traditional>

```
0x00007FFFFFFFDBE0  00401252 00000000 00401265 00000000 0040127A 00000000   R.@.....e.@.....z.@.....
                    ↖hbSendingThreadN↖hbReceivingThrea↖workerThreadName
0x00007FFFFFFFDBF8  0040128A 00000000 00000001 00000002 00000003 00000004   ..@.....................
                    ↖producerThreadNa↖digits,*digits
0x00007FFFFFFFDC10  00000004 00000003 00000002 00000001 00000000 00000000   ........................
                    ↖rdigits,*rdigits                  ↖rbp
0x00007FFFFFFFDC28  F7818EC5 00007FFF 00000000 00000000 FFFFDD08 00007FFF   Å..÷ÿ...........Ýÿÿÿ...
0x00007FFFFFFFDC40  00000000 00000001 00400F97 00000000 00000000 00000000   ..........@.............
0x00007FFFFFFFDC58  0AD207CB 816AC5E2 00400A10 00000000 FFFFDD00 00007FFF   Ë.Ò.âÅj...@.....Ýÿÿÿ...
0x00007FFFFFFFDC70  00000000 00000000 00000000 00000000 B2B207CB 7E953A1D   ................Ë.²².:.~
0x00007FFFFFFFDC88  162807CB 7E952AE1 00000000 00000000 00000000 00000000   Ë.(.á*.~................
```



**Specify the mapping paths**

Name: Path Mapping

| Compilation path: | Local file system path: |
|---|---|
| C:\Path\On\Windows | /path/on/linux |
| \\UNC\Path\Here | /path/on/linux |
| /path/to/../relative/compilation | /path/on/linux |

[✓] Resolve path mappings using debugger backend.

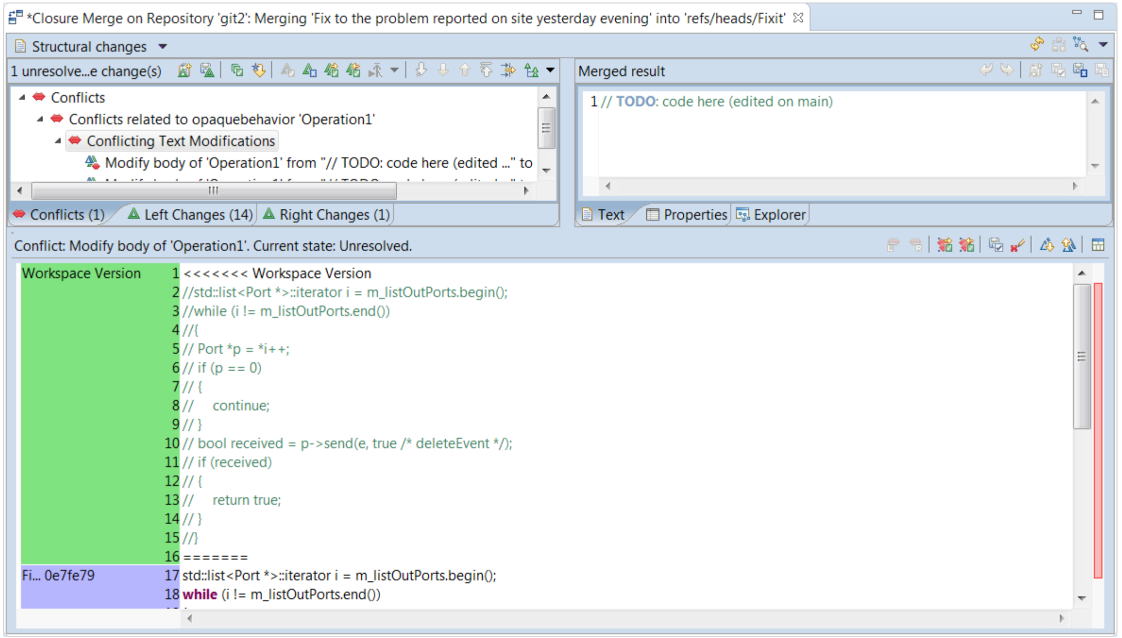
*For more information about news in CDT 9.1 see [https://wiki.eclipse.org/CDT/User/NewIn90](https://wiki.eclipse.org/CDT/User/NewIn90) and [https://wiki.eclipse.org/CDT/User/NewIn91](https://wiki.eclipse.org/CDT/User/NewIn91)*

## 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*
- An experimental feature is available for adding standard include paths to generated CDT projects
  - Makes Codan report much fewer errors, and therefore makes it more stable
  - The experimental feature is located in *UML Development* – *Real Time C++ Transformations* – *Generate additional information for Code Analysis*
  - 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 looses 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*



*Closure Merge on Repository 'git2': Merging 'Fix to the problem reported on site yesterday evening' into 'refs/heads/Fixit'

Structural changes

1 unresolve...e change(s)

- Conflicts
  - Conflicts related to opaquebehavior 'Operation1'
    - Conflicting Text Modifications
      - Modify body of 'Operation1' from "// TODO: code here (edited ..." to

Conflicts (1) | Left Changes (14) | Right Changes (1)

Merged result

```
1 // TODO: code here (edited on main)
```

Text | Properties | Explorer

Conflict: Modify body of 'Operation1'. Current state: Unresolved.

Workspace Version

Fi... 0e7fe79

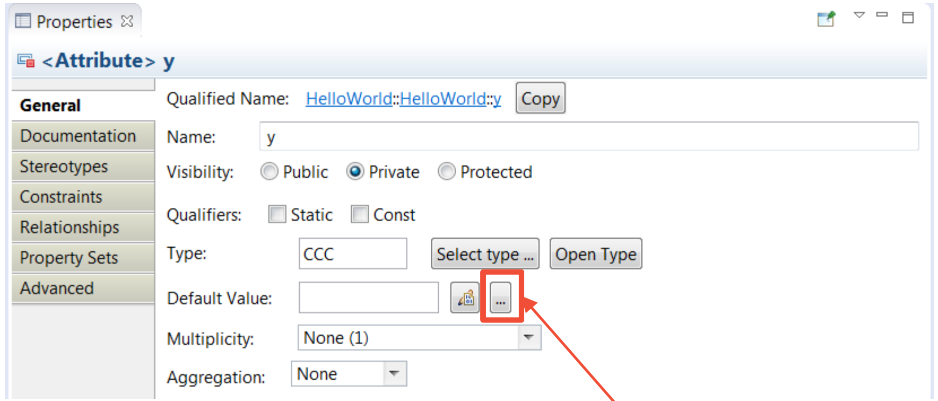
```
 1 <<<<<<< Workspace Version
 2 //std::list<Port *>::iterator i = m_listOutPorts.begin();
 3 //while (i != m_listOutPorts.end())
 4 //{
 5 // Port *p = *i++;
 6 // if (p == 0)
 7 // {
 8 //     continue;
 9 // }
10 // bool received = p->send(e, true /* deleteEvent */);
11 // if (received)
12 // {
13 //     return true;
14 // }
15 //}
16 =======
17 std::list<Port *>::iterator i = m_listOutPorts.begin();
18 while (i != m_listOutPorts.end())
```

## 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:



<Attribute> y

Qualified Name: HelloWorld::HelloWorld::y

Name: y

Visibility: Public [✓ Private] Protected

Qualifiers: ☐ Static ☐ Const

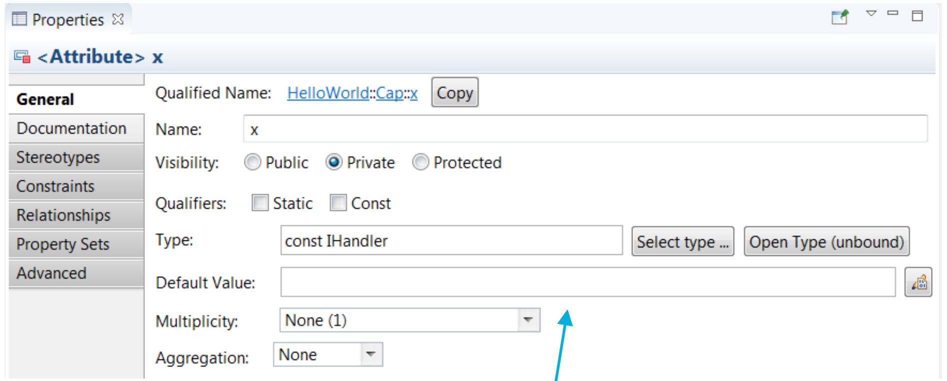
Type: CCC

Default Value:

Multiplicity: None (1)

Aggregation: None

*before*



<Attribute> x

Qualified Name: HelloWorld::Cap::x

Name: x

Visibility: Public [✓ Private] Protected

Qualifiers: ☐ Static ☐ Const

Type: const IHandler

Default Value:

Multiplicity: None (1)

Aggregation: None

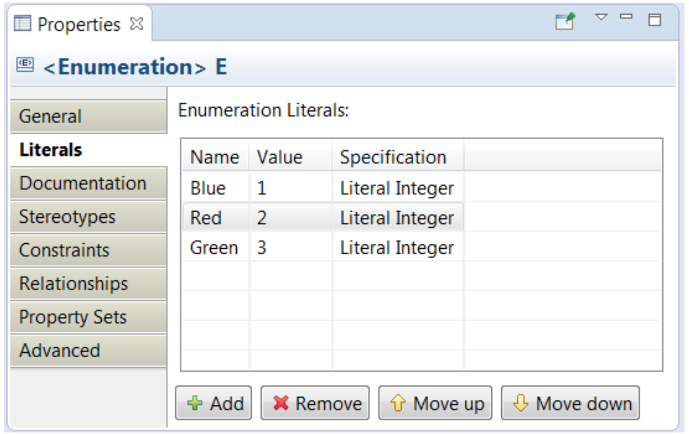
*now*

Removed useless Browse button for default values

More space for editing type and default value of an attribute

# Improved Editing of Literals and Parameters

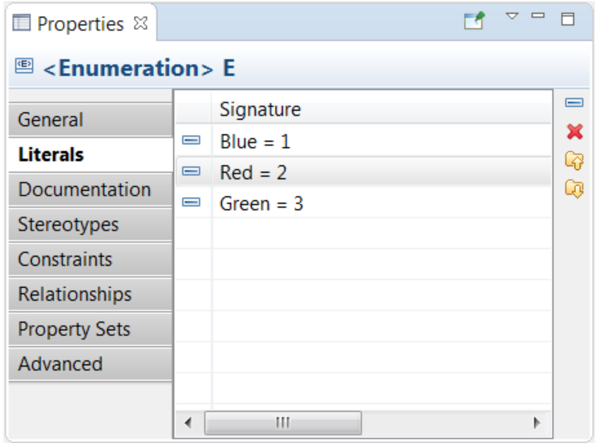
- The Properties editor now allows enumeration literals and operation parameters to be edited using textual syntax
  – Similar to how operations and attributes are edited
  – The context menu provides navigation to the literal or parameter



Enumeration Literals:

| Name | Value | Specification |
|---|---|---|
| Blue | 1 | Literal Integer |
| Red | 2 | Literal Integer |
| Green | 3 | Literal Integer |

*before*

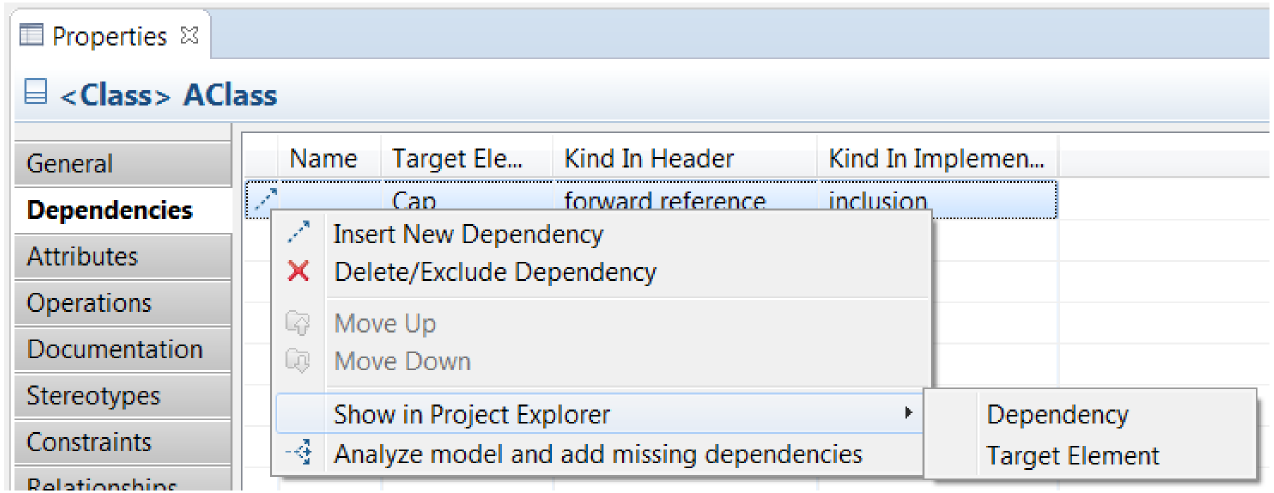


| Signature |
|---|
| Blue = 1 |
| Red = 2 |
| Green = 3 |

*now*

# Navigation from Dependencies

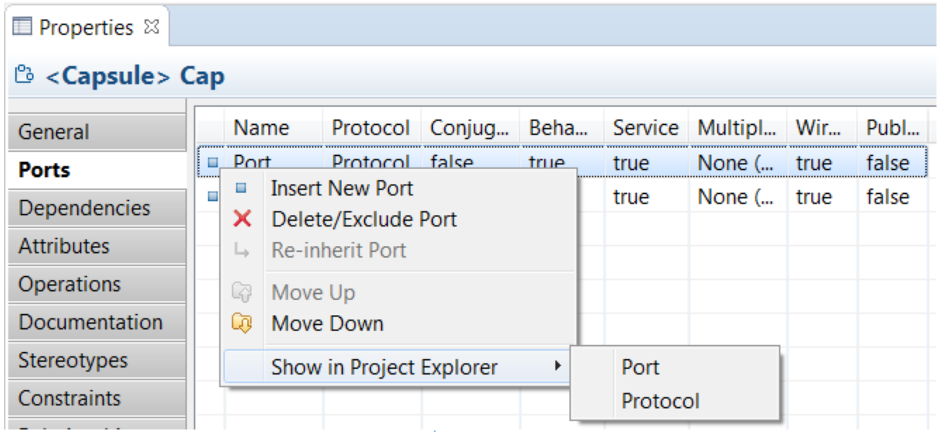
- The Dependencies tab of the Properties view now provides commands for navigating to:
  - the dependency itself
  - the target element of the dependency



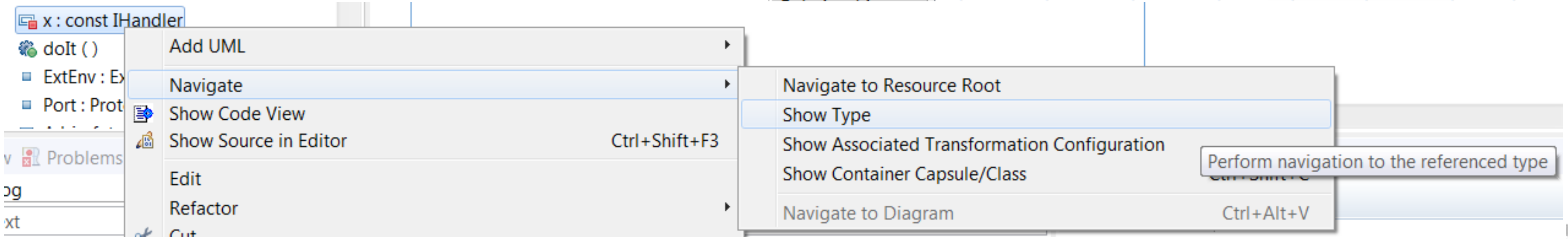
| Name | Target Ele... | Kind In Header | Kind In Implemen... |
|---|---|---|---|
| | Cap | forward reference | inclusion |

# 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

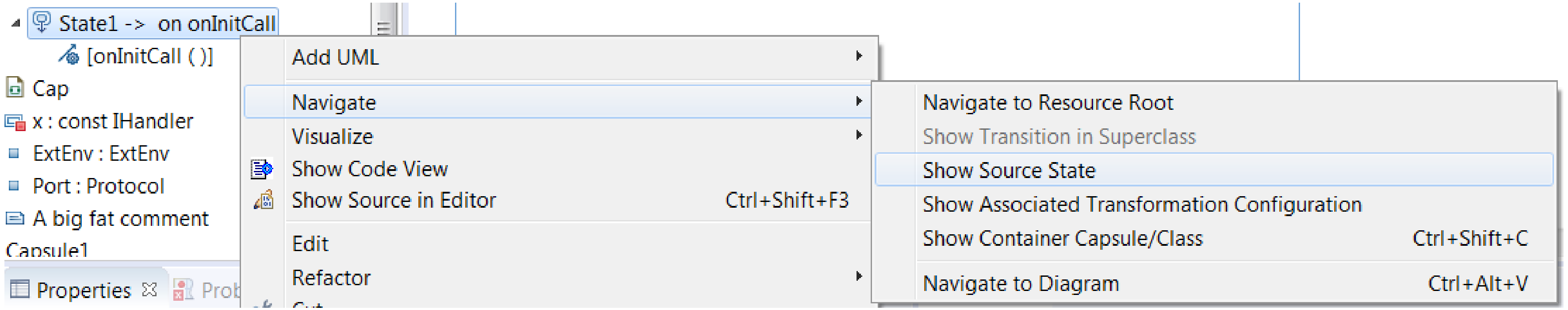


| Name | Protocol | Conjug... | Beha... | Service | Multipl... | Wir... | Publ... |
|---|---|---|---|---|---|---|---|
| Port | Protocol | false | true | true | None (... | true | false |
|  |  |  |  | true | None (... | true | false |


x : const IHandler
doIt ( )
ExtEnv : Ex
Port : Prot
Problems
Add UML
Navigate
Show Code View
Show Source in Editor Ctrl+Shift+F3
Edit
Refactor
Cut
Navigate to Resource Root
Show Type
Show Associated Transformation Configuration
Show Container Capsule/Class
Navigate to Diagram Ctrl+Alt+V
Perform navigation to the referenced type

# Navigation to Source State of a Transition

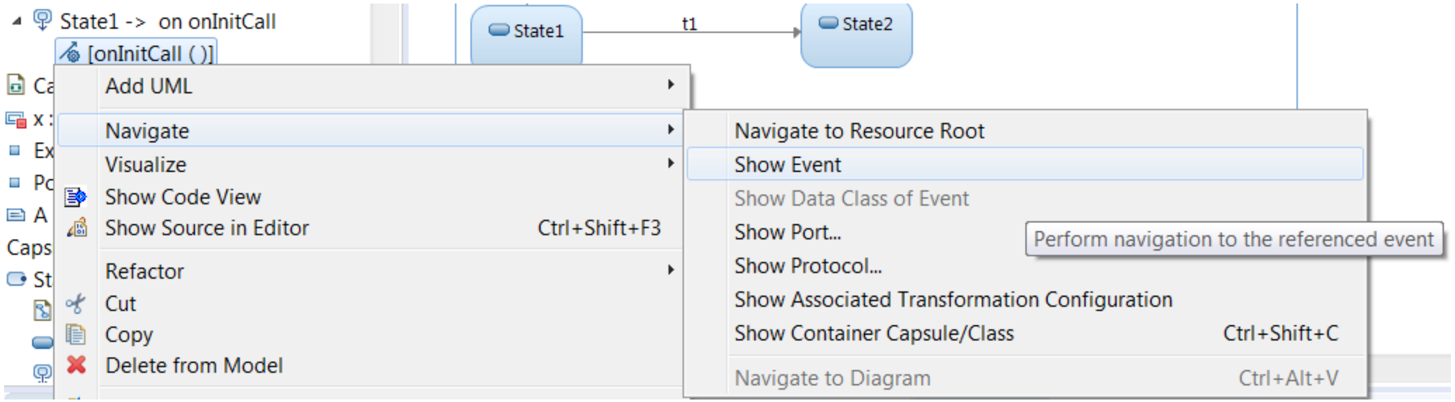
- The Project Explorer now supports navigation from a transition to its source state (i.e. the state from which the transition originates)


State1 -> on onInitCall
[onInitCall ( )]
Cap
x : const IHandler
ExtEnv : ExtEnv
Port : Protocol
A big fat comment
Capsule1
Properties
Prob
Add UML
Navigate
Visualize
Show Code View
Show Source in Editor Ctrl+Shift+F3
Edit
Refactor
Navigate to Resource Root
Show Transition in Superclass
Show Source State
Show Associated Transformation Configuration
Show Container Capsule/Class Ctrl+Shift+C
Navigate to Diagram Ctrl+Alt+V

# 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

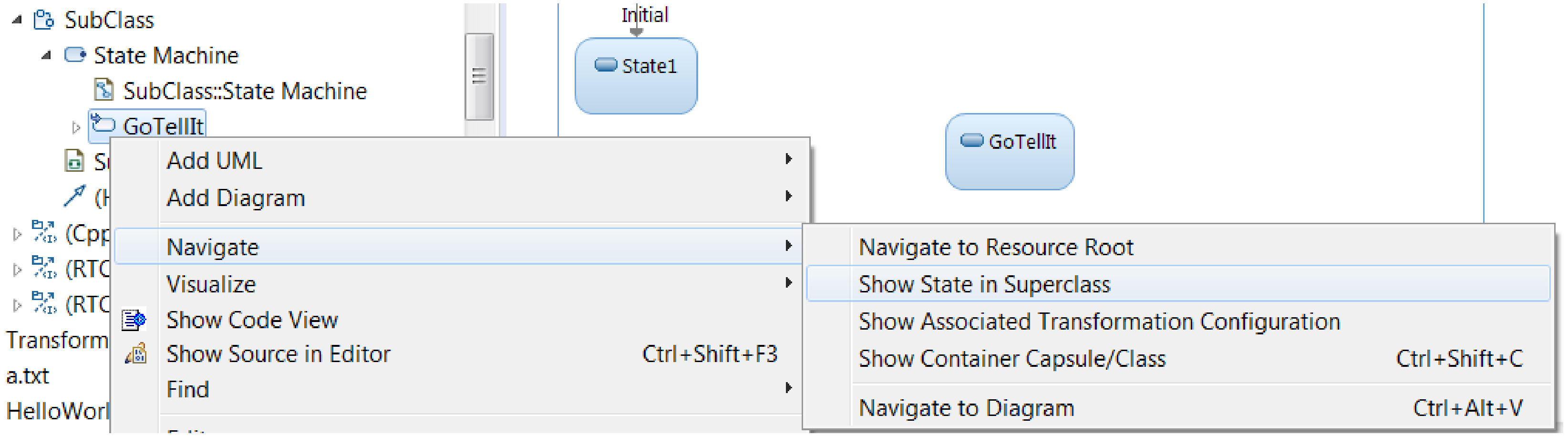
  For passive class triggers navigation to the trigger operation is provided instead.



State1 -> on onInitCall
[onInitCall ( )]
State1
t1
State2
Add UML
Navigate
Visualize
Show Code View
Show Source in Editor Ctrl+Shift+F3
Refactor
Cut
Copy
Delete from Model
Navigate to Resource Root
Show Event
Show Data Class of Event
Show Port...
Show Protocol...
Show Associated Transformation Configuration
Show Container Capsule/Class Ctrl+Shift+C
Navigate to Diagram Ctrl+Alt+V
Perform navigation to the referenced event

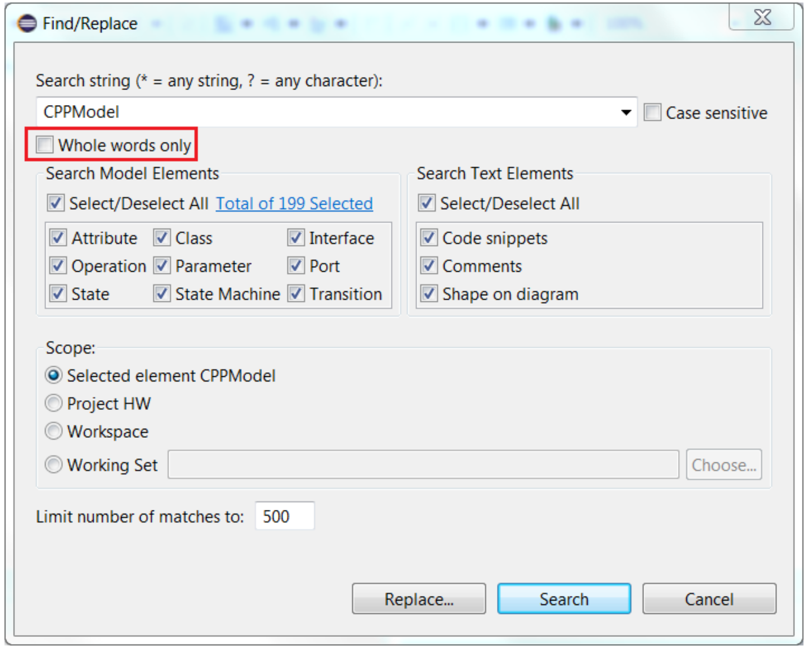
# Navigation from Redefined or Excluded Elements

- 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.


SubClass
State Machine
SubClass::State Machine
GoTellIt
Initial
State1
GoTellIt
Add UML
Add Diagram
Navigate
Visualize
Show Code View
Show Source in Editor Ctrl+Shift+F3
Find
Navigate to Resource Root
Show State in Superclass
Show Associated Transformation Configuration
Show Container Capsule/Class Ctrl+Shift+C
Navigate to Diagram Ctrl+Alt+V

# Whole Word Search

- The Find/Replace and Model Search dialogs now has an option for whole word search
- It roughly corresponds to enclosing the search string in double quotes when searching from the search field
- When this option is set, the search behaves very similar to how searching in 9.x worked (it had a similar checkbox)



Find/Replace

Search string (* = any string, ? = any character):

CPPModel

[ ] Case sensitive

[ ] Whole words only

Search Model Elements

[✓] Select/Deselect All Total of 199 Selected

| | | |
|---|---|---|
| [✓] Attribute | [✓] Class | [✓] Interface |
| [✓] Operation | [✓] Parameter | [✓] Port |
| [✓] State | [✓] State Machine | [✓] Transition |

Search Text Elements

[✓] Select/Deselect All

[✓] Code snippets

[✓] Comments

[✓] Shape on diagram

Scope:

[●] Selected element CPPModel

( ) Project HW

( ) Workspace

( ) Working Set

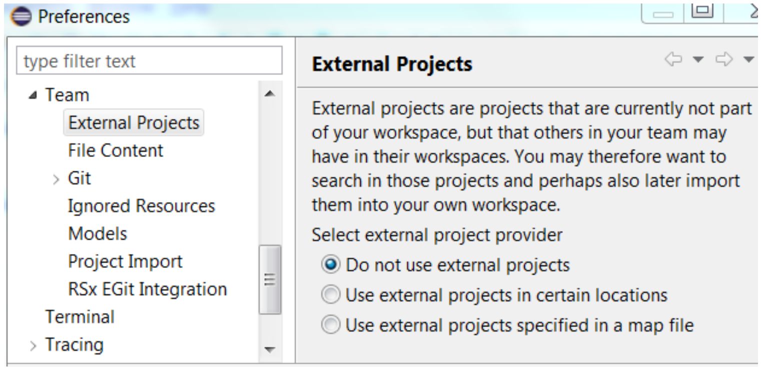
Limit number of matches to: 500

# 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

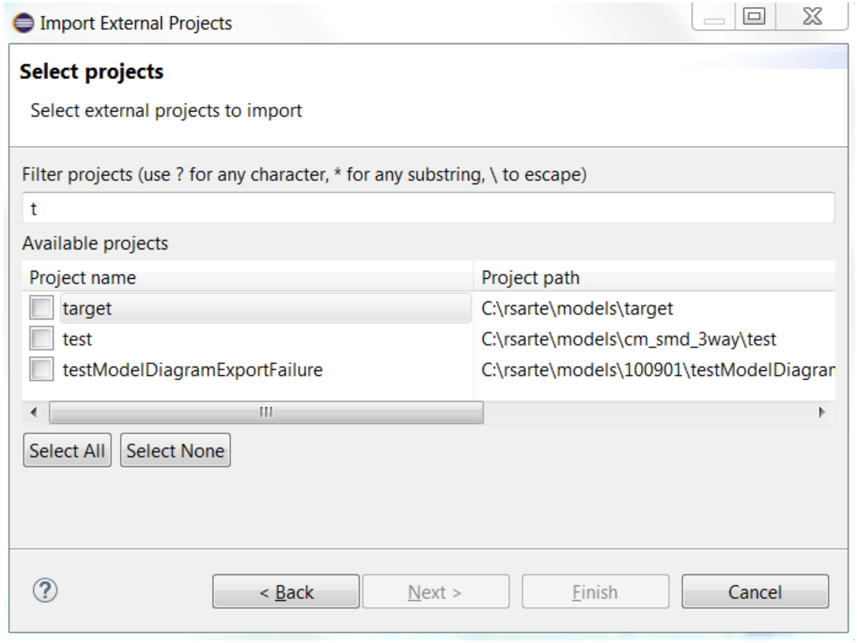


**External Projects**

External projects are projects that are currently not part of your workspace, but that others in your team may have in their workspaces. You may therefore want to search in those projects and perhaps also later import them into your own workspace.

Select external project provider

- [✓ Do not use external projects]
- Use external projects in certain locations
- Use external projects specified in a map file



Import External Projects

**Select projects**

Select external projects to import

Filter projects (use ? for any character, * for any substring, \ to escape)

t

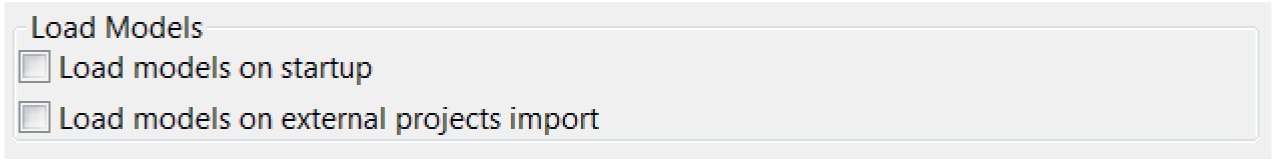
Available projects

| Project name | Project path |
|---|---|
| target | C:\rsarte\models\target |
| test | C:\rsarte\models\cm_smd_3way\test |
| testModelDiagramExportFailure | C:\rsarte\models\100901\testModelDiagrar |

- 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*



Load Models

☐ Load models on startup

☐ Load models on external projects import

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

# Load UML Models

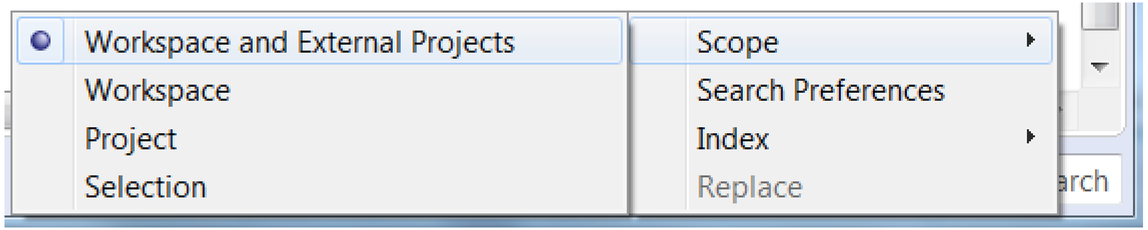
- This command was changed to always load all models in the workspace
  - Users rarely used the possibility to only load some of the models
  - Populating the ”Load UML Models” dialog could take a significant amount of time in large workspaces
- In the following API the ’prompt’ parameter is now ignored (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()** | *Load everything* |
| **loadUMLModels(List<IFile>)** | *Load specified model files* |
| **loadUMLModels(List<IFile>, Consumer<Boolean>)** | *Load specified model files with callback when done* |

# 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

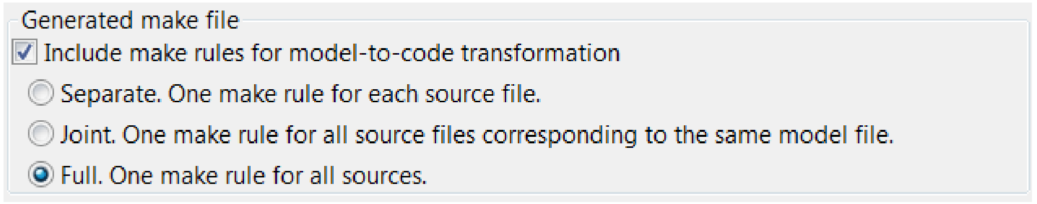


- This makes it possible to decide whether to search in external projects or not, without having to disable the External Projects preference

# Model Compiler

- 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

  *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

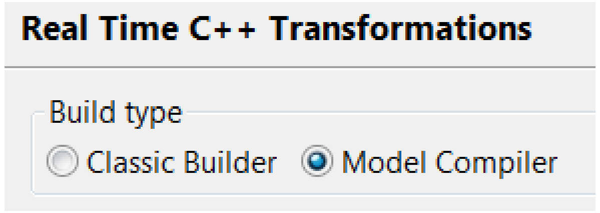
○ Separate. One make rule for each source file.

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

● Full. One make rule for all sources.

# 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.



**Real Time C++ Transformations**

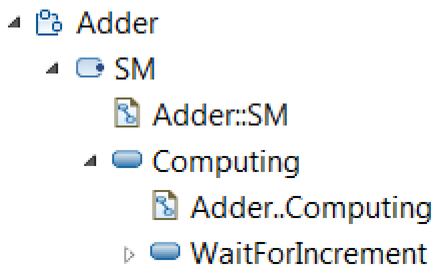
Build type

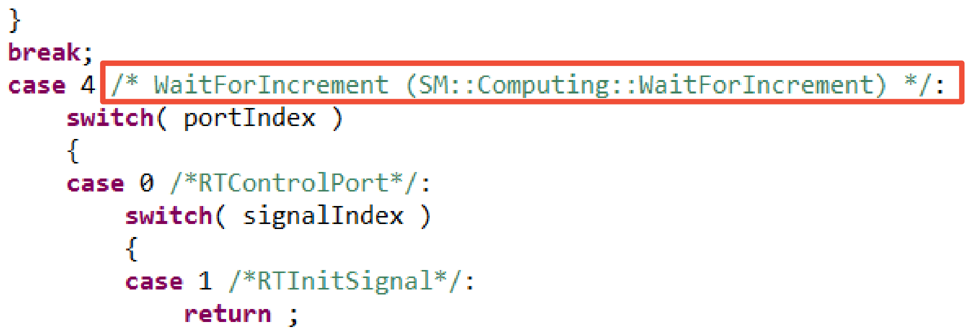
Classic Builder [✓ Model Compiler]

- 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 ;
```

THANK YOU!