

What's New in HCL RTist 11.1


updated for release 2021.40

Overview

- ▶ RTist 11.1 is based on Eclipse 2020.06 (4.16)
- ▶ HCL RTist is 100% compatible with IBM RSARTE. All features in IBM RSARTE are also present in HCL RTist. However, HCL RTist contains some features that do not exist in IBM RSARTE.

- Those features are marked in this presentation by



 HCL RTist
Version: 11.1.0.v20211007_1210
Release: 2021.40

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

(c) Copyright HCL Technologies Ltd. 2016, 2021. All rights reserved.

Visit <https://RTist.hcldoc.com/help/topic/com.ibm.xttools.rsarte.webdoc/users-guide/overview.html>

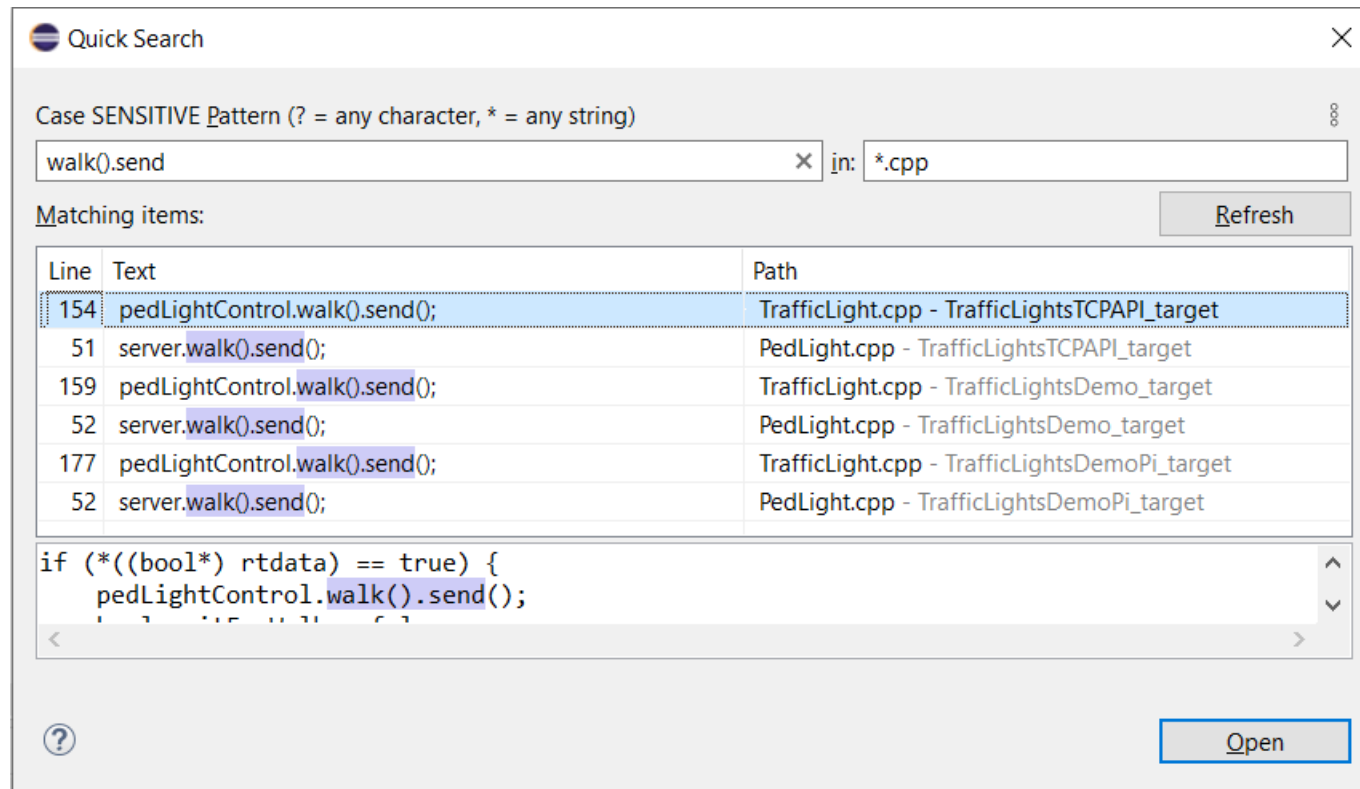


Eclipse 4.16 (2020.06)

- ▶ Compared to RTist 11.0, RTist 11.1 includes new features from 4 quarterly Eclipse releases:
 - 2019.09 (<https://www.eclipse.org/eclipse/news/4.13/platform.php>)
 - 2019.12 (<https://www.eclipse.org/eclipse/news/4.14/platform.php>)
 - 2020.03 (<https://www.eclipse.org/eclipse/news/4.15/platform.php>)
 - 2020.06 (<https://www.eclipse.org/eclipse/news/4.16/platform.php>)
- ▶ For full information about all improvements and changes in these Eclipse releases see the links above
 - Some highlights are listed in the next few slides...

Eclipse 4.16 (2020.06)

- ▶ A new Quick Search dialog allows you to search the files of your workspace faster (“as-you-type”)
 - For a similar search experience in model files, use the Find Named Element command instead



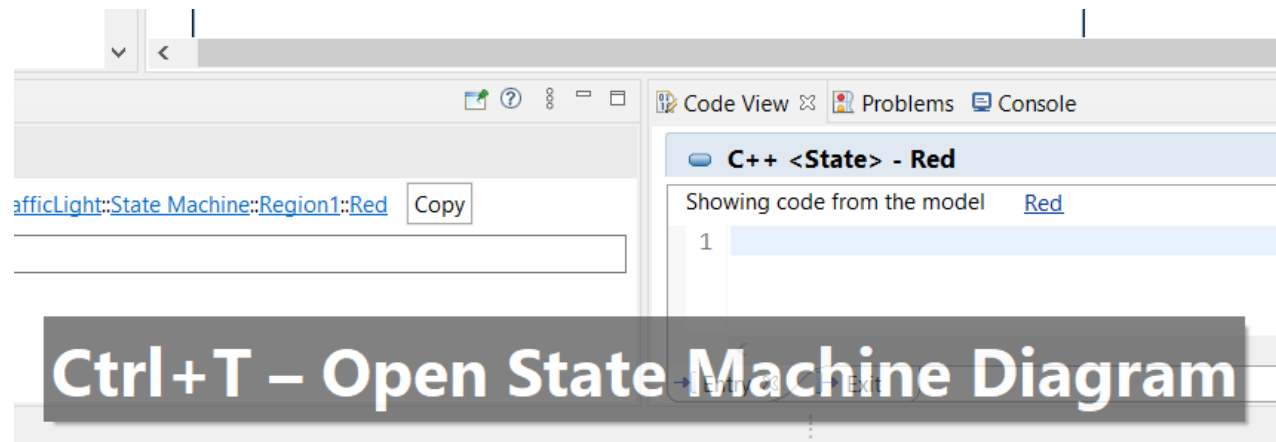
Eclipse 4.16 (2020.06)

- ▶ By default at most 99 editors can now be open at the same time
 - Helps keeping the performance good when working with Eclipse for a long time
 - This can be controlled by the preference **General – Editors – Close editors automatically**
- ▶ Showing key bindings when performing commands
 - New preferences in **General – Keys**
 - This is a good way to learn about key bindings for the commands that are used, and can also help in presentations

Show key binding when command is invoked

Through keyboard

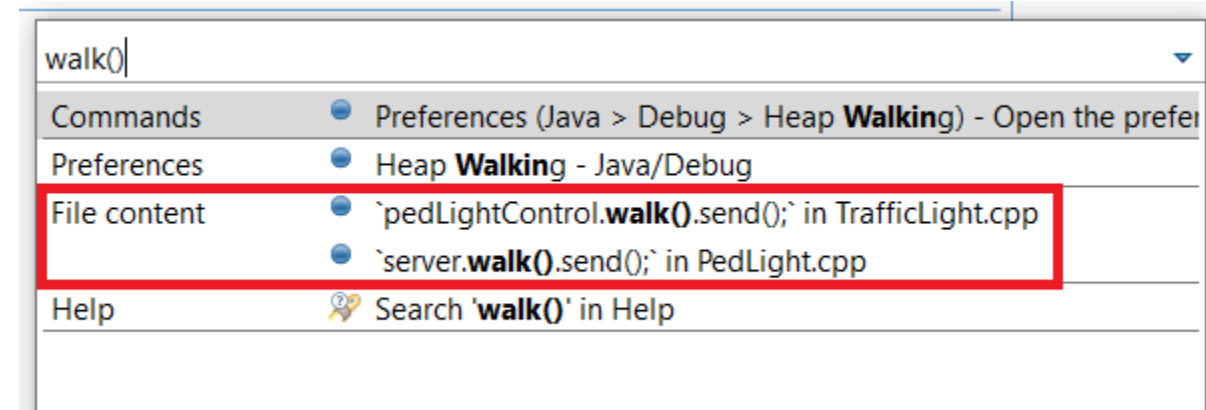
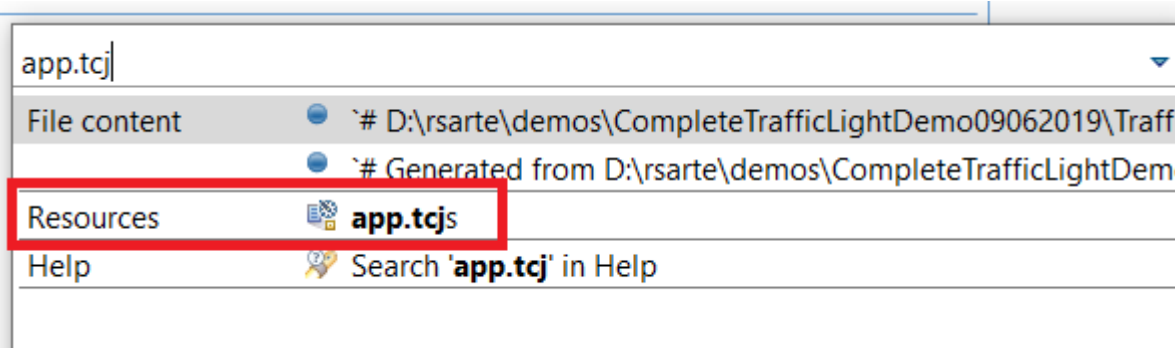
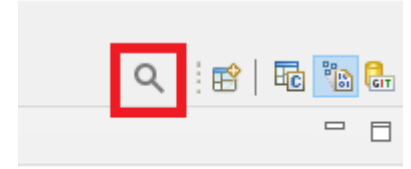
Through mouse click



Eclipse 4.16 (2020.06)

▶ Quick Access field replaced with toolbar button

- Takes less space in the toolbar, and instead uses a normal dialog for typing and showing the results
- Same key binding as before (Ctrl + 3) but the command is now called “Find Actions”
- The results now also include matching files in the workspace, and text matches in files (requires that Quick Search has been used at least once)



Eclipse 4.16 (2020.06)

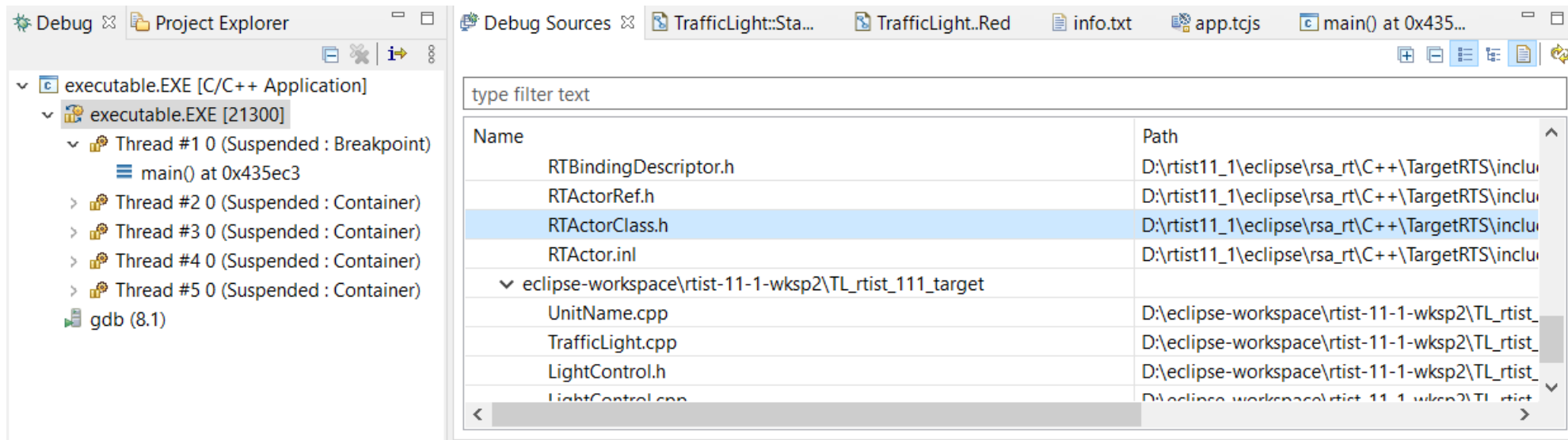
- ▶ Show code problems inline
 - Makes errors/warnings more visible and lets you apply quick fixes without having to go to the Problems view
 - Enable this feature in preferences at **General – Editors – Text Editors – Show code minings for problem annotations**
- ▶ There were several improvements in SWT and GTK
 - The minimal supported GTK version is now 3.20

```
34 {  
35 //{{{USR platform:/resource/TL_rtist_111/Tr  
36 log.log("Red -> Green");  
   ⓧ 'cout' is not a member of 'std'  
37 std::cout << "test";  
38 //}}}USR  
39 }  
40
```

CDT 9.11 (included as part of Eclipse 2020.06)

▶ New Debug Sources view

- Shows source files the C++ debugger knows about when debugging an application
- Useful in particular when the application contains source files that are not present in the Eclipse workspace
- Source files can be found by searching (filtering) and opened by double-click



The screenshot shows the Eclipse IDE interface during a debug session. The Project Explorer on the left shows the application structure, including the executable and several threads. The Debug Sources view on the right displays a table of source files found by the debugger. The table has two columns: Name and Path. The file RTActorClass.h is highlighted in blue.

Name	Path
RTBindingDescriptor.h	D:\rtist11_1\eclipse\rta_rt\C++\TargetRTS\inclu
RTActorRef.h	D:\rtist11_1\eclipse\rta_rt\C++\TargetRTS\inclu
RTActorClass.h	D:\rtist11_1\eclipse\rta_rt\C++\TargetRTS\inclu
RTActor.inl	D:\rtist11_1\eclipse\rta_rt\C++\TargetRTS\inclu
▼ eclipse-workspace\rtist-11-1-wksp2\TL_rtist_111_target	
UnitName.cpp	D:\eclipse-workspace\rtist-11-1-wksp2\TL_rtist_
TrafficLight.cpp	D:\eclipse-workspace\rtist-11-1-wksp2\TL_rtist_
LightControl.h	D:\eclipse-workspace\rtist-11-1-wksp2\TL_rtist_
LightControl.cpp	D:\eclipse-workspace\rtist-11-1-wksp2\TL_rtist_

CDT 9.11 (included as part of Eclipse 2020.06)

- ▶ CODAN improvements
 - Several additional checks implemented
- ▶ For more information about CDT improvements see
 - <https://wiki.eclipse.org/CDT/User/NewIn99>
 - <https://wiki.eclipse.org/CDT/User/NewIn910>
 - <https://wiki.eclipse.org/CDT/User/NewIn911>

Newer EGit Version in the EGit Integration

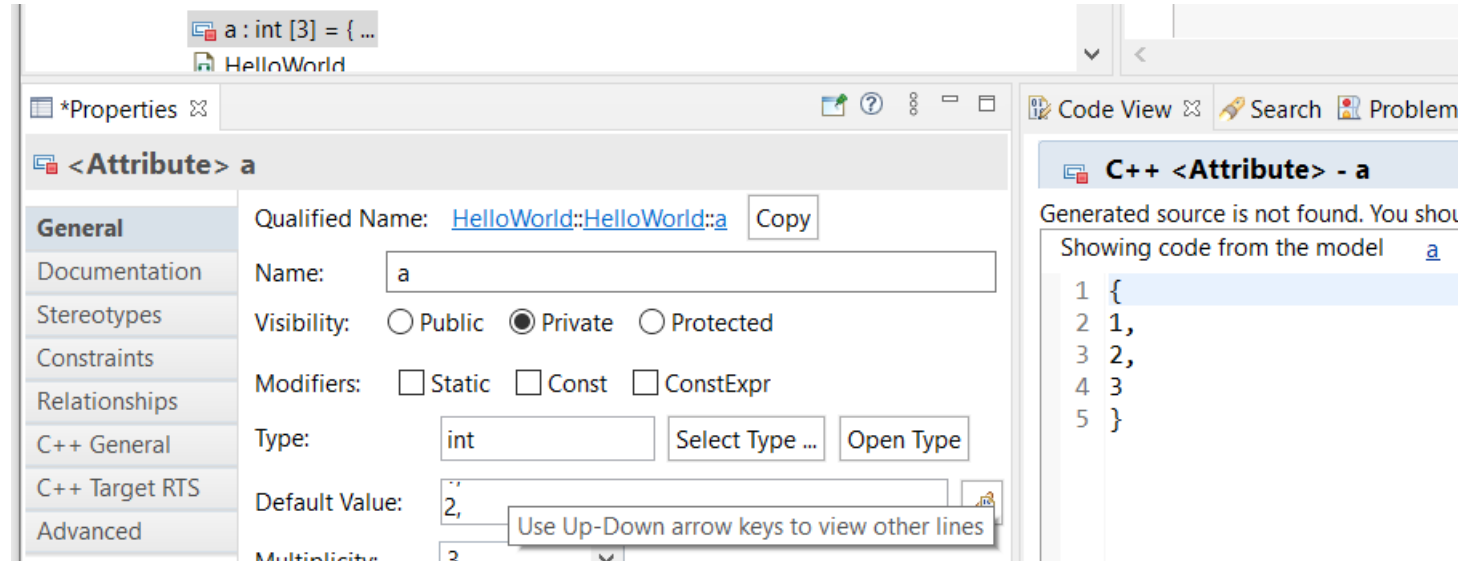
- ▶ The EGit integration in RTist has upgraded EGit from 5.4 to 5.8
 - This is the recommended and latest version for Eclipse 2020.06
- ▶ This upgrade provides several new features, performance improvements and bug fixes
 - For detailed information about the changes see
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.5
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.6
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.7
 - https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.8

Installation Script

- ▶ A bash script is now available which helps automating the installation of RTist
 - Download it from the [Info Center](#)
 - Works on both Windows and Linux
- ▶ In particular useful for installing RTist 11.1 (due to the requirement of using Java 11 for the installation)
 - Choose whether you want to then run RTist with either Java 8 or Java 11
- ▶ For documentation on how to configure and use the script see the [Info Center](#).

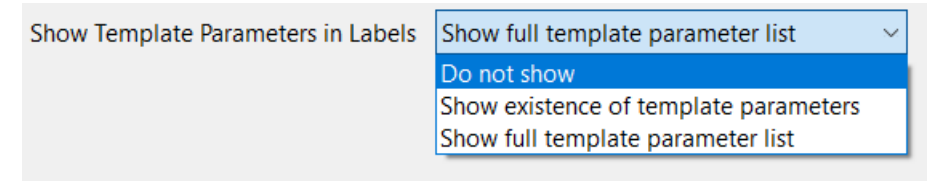
Properties View Improvements

- ▶ The Default Value field now supports multi-line values
 - To create a multi-line default value you still need to use the Code View or Code Editor
 - For editing a multi-line default value you can now use the Properties view, but it's still often more convenient with the Code View or Code Editor
 - For quickly viewing a multi-line default value the Properties view can be handy



Project Explorer Improvements

- ▶ The Project Explorer can now show template information after the name of an element that has template parameters
 - Makes it easier to see if an element is a template without having to expand it in the Project Explorer, or look in the Properties view
 - A new preference **RealTime Development – Project Explorer – Show Template Parameters in Labels** controls what to show



▼ List
typename Element
size : unsigned int

*Do not show
template parameters*

▼ List<T>
typename Element
size : unsigned int

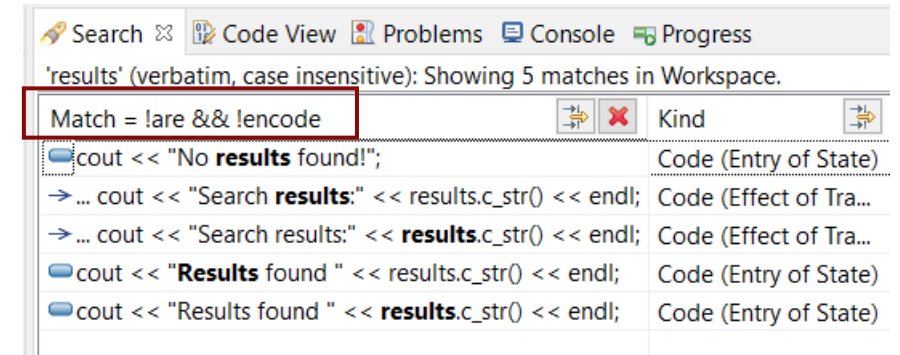
*Show existence of
template parameters*

▼ List<typename Element, size : unsigned int>
typename Element
size : unsigned int

*Show full template
parameter list*

Search Filtering

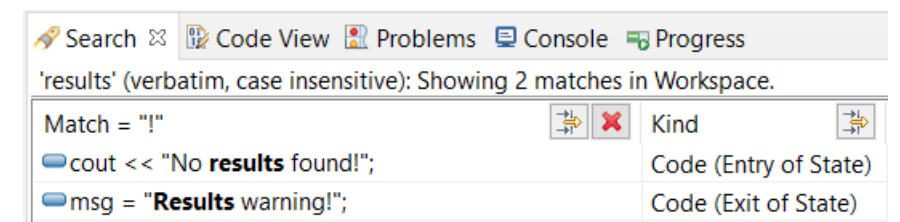
- ▶ It's now possible to filter search results using Boolean operators NOT (!) and AND (&&)
 - Useful if a search returns too many matches
 - Use a filter on the form
!A && !B && ... !X to hide matches where certain words are not present
 - Use a filter on the form
A && B && ... X to only show matches where certain words are present
 - ...or any combination, where some words are present and others not
- ▶ Enclose the filter string in double quotes to apply the filter verbatimly
 - Needed if the filter string contains the characters ! or &&



Search Code View Problems Console Progress

'results' (verbatim, case insensitive): Showing 5 matches in Workspace.

Match = !are && !encode	Kind
cout << "No results found!";	Code (Entry of State)
→ ... cout << "Search results :" << results.c_str() << endl;	Code (Effect of Tra...
→ ... cout << "Search results:" << results.c_str() << endl;	Code (Effect of Tra...
cout << " Results found " << results.c_str() << endl;	Code (Entry of State)
cout << "Results found " << results.c_str() << endl;	Code (Entry of State)



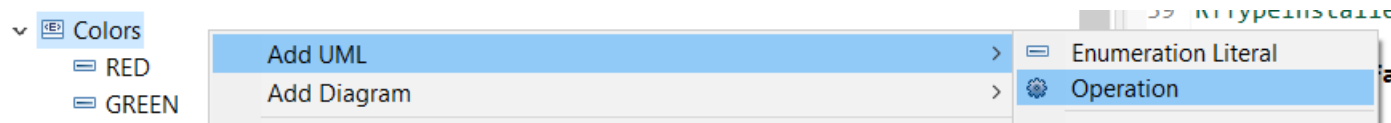
Search Code View Problems Console Progress

'results' (verbatim, case insensitive): Showing 2 matches in Workspace.

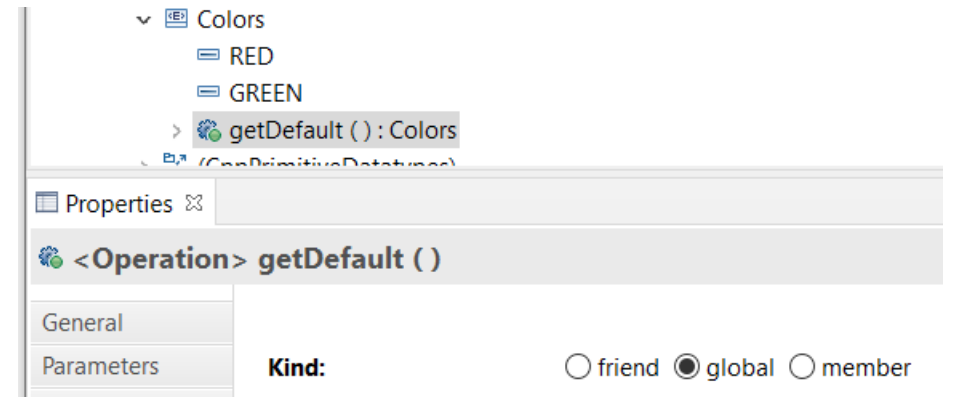
Match = "!"	Kind
cout << "No results found!";	Code (Entry of State)
msg = " Results warning!";	Code (Exit of State)

Enums with Operations

- ▶ Enumerations can now have operations
 - Create them as usual with **Add UML - Operation**



- ▶ Such operations will be translated to global functions
 - C++ enums cannot have member functions, but it's sometimes useful to have functions that operate on or return enum literals
 - Using global functions can then be an alternative to wrapping the enum inside a class



- ▶ This works the same both for scoped and non-scoped enumerations

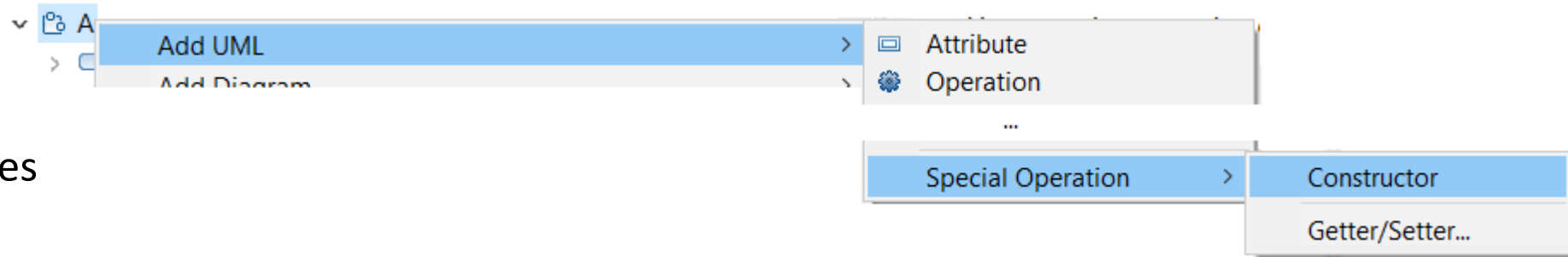
Generic Type Descriptors

- ▶ The model compiler now supports generating type descriptors for type aliases with template parameters
 - For example: `template<typename T, unsigned int N > using StdArray = std::array<T, N>;`
 - If type descriptor functions are defined for the type alias, they will be generated as template functions with the same template parameters
 - Allows to implement generic type descriptors that work for all (or many) instantiations of the template
 - A new `RTObject_class::fromType<T>()` template function can be used for looking up the type descriptor of a type at compile time. Useful for example when implementing generic encode or decode functions. Specialize it for the types that you use (specializations for built-in types are available in the TargetRTS). For example:

```
template <> inline const RTObject_class* RTObject_class::fromType<RTString>() {  
    return &RTType_RTString;  
}
```
- ▶ You can specify a unique name for the type descriptor of a specific template instantiation
 - For example: `template <> const char* RTName_StdArray<StdString, 4>::name = "StdArray<StdString, 4>";`
 - The TargetRTS now prints a warning if two type descriptors with the same name exists. Helps troubleshooting missing template specializations for the name attribute.

Custom Capsule Constructors

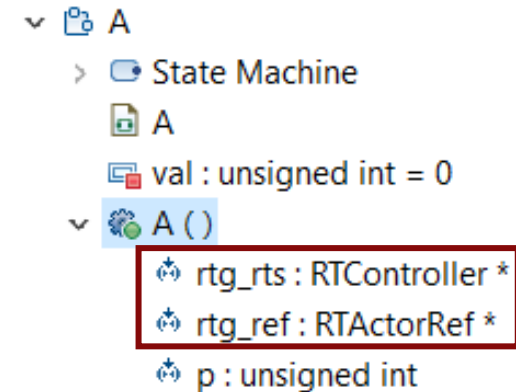
- ▶ It's now possible to create custom constructors for capsules



- ▶ Each capsule constructor has two mandatory parameters:
 - **rtg_rts** Controller (i.e. thread) that will run the created capsule instance
 - **rtg_ref** Capsule part where the created capsule instance will be inserted

- ▶ In addition you can add any number of user-defined parameters

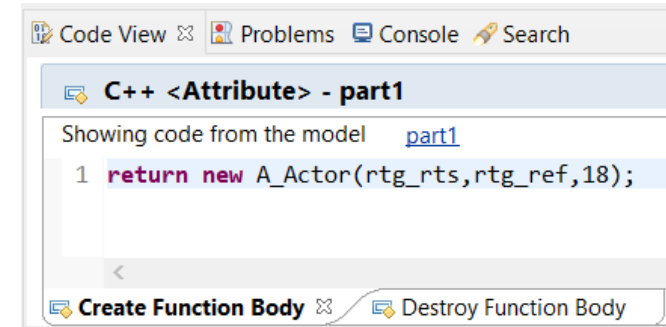
- ▶ This feature makes it possible to pass initialization data to a capsule instance already when it's created



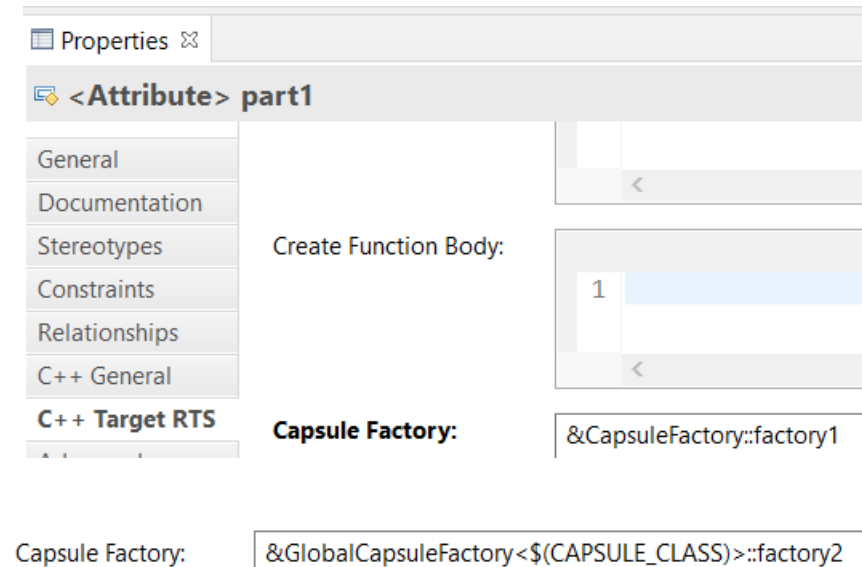
- Previously this could only be done by sending such data with the initialization event (which is not possible for fixed capsule parts)
- Custom capsule constructors work for all capsules regardless of the capsule part they are incarnated into

Capsule Factories (1/2)

- ▶ The concept of a capsule factory was introduced to allow incarnating capsules with custom constructors
 - Specifies how a capsule instance is created and destroyed
 - Can be provided in various ways (in a hierarchical manner)
- ▶ New capsule factory code snippets for capsule parts
 - All capsule instances incarnated in that capsule part will use the specified Create/Destroy code
- ▶ New capsule factory property for capsule parts
 - Will be used if no Create/Destroy code is provided for that capsule part
- ▶ New capsule factory property in the TC
 - Will be used if none of the above are provided
 - Allows specifying a default (global) capsule factory
 - A variable $\$(CAPSULE_CLASS)$ can be used in this TC property (expands to the name of the class that is generated from the type of the capsule part)



The screenshot shows a code editor window titled "C++ <Attribute> - part1". It displays a single line of C++ code: `1 return new A_Actor(rtg_rts, rtg_ref, 18);`. The editor has tabs for "Code View", "Problems", "Console", and "Search". Below the code editor, there are two buttons: "Create Function Body" and "Destroy Function Body".



The screenshot shows the Eclipse IDE's Properties window for a capsule part named "<Attribute> part1". The window has a sidebar with tabs for "General", "Documentation", "Stereotypes", "Constraints", "Relationships", "C++ General", and "C++ Target RTS". The "C++ Target RTS" tab is selected. The main area shows the "Create Function Body" property with a code snippet: `1`. Below it, the "Capsule Factory" property is set to `&CapsuleFactory::factory1`. At the bottom, there is a "Capsule Factory:" label with a text box containing `&GlobalCapsuleFactory<$(CAPSULE_CLASS)>::factory2`.

Capsule Factories (2/2)

- ▶ For optional capsule parts, it's also possible to provide the capsule factory using a new TargetRTS

function RTFrame::incarnateCustom()

```
RTActorId incarnateCustom( RTActorRef & cp,  
RTActorFactory& factory,  
int index = -1);
```

- In this case, only the Create code can be provided (the regular delete operator will be used for destroying such capsule instances)
- Example usage:

```
RTActorId id = frame.incarnateCustom(part1,  
RTActorFactory([this](RTController * c, RTActorRef * a, int index) {  
    return new A_Actor(c, a, 444); // User-defined constructor  
}))  
);
```

- ▶ If multiple capsule factories are provided, they will be picked in this priority order:
 1. The capsule factory provided in a call to RTFrame::incarnateCustom()
 2. The capsule factory specified by means of Create and/or Destroy code snippets on a capsule part
 3. The capsule factory specified by the "Capsule Factory" property on a capsule part
 4. The capsule factory specified in the "Capsule Factory" property on the TC

Dependency Injection

- ▶ A capsule normally depends on many things at run-time for its execution
 - Examples: Other capsules typing its capsule parts, the thread that will run the capsule, initialization data to pass to the capsule constructor, etc.
- ▶ Spreading out such dependencies in a hard-coded way in an application can make it hard to change them to configure different variants of an application
 - E.g. mocking out dependent capsules when unit testing a capsule
- ▶ The TargetRTS now provides a new dependency injection service realized by the RTInjector class
 - Register the dependencies to configure the application (typically early, e.g. in the top capsule constructor)
 - A create function can be registered for a capsule part (identified by its qualified path name)
 - A capsule factory can delegate to `RTInjector::create()` for creating capsule instances
 - If necessary, registered dependencies can be changed at run-time

```
C++ <Operation> TOP ( rtg_rts : RTController *, rtg_ref : RTActorRef * )  
Showing code from the model TOP \(...\)  
1 RTInjector::getInstance().registerCreateFunction("/logger:0/logger",  
2     [this](RTController * c, RTActorRef * a, int index) {  
3         //return new SimpleLogger_Actor(c, a);  
4         return new TimestampLogger_Actor(LoggerThread, a);  
5     }  
6 );
```

Code Compliance

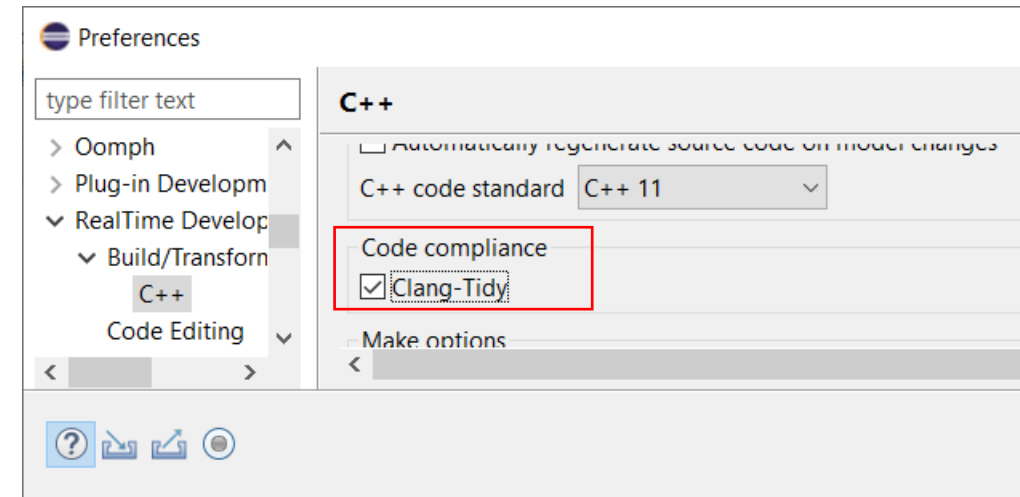
- ▶ A new preference was introduced to let the model compiler generate code according to certain code compliance rules
- ▶ Support for two specific Clang-Tidy rules are implemented:
 - **cppcoreguidelines-pro-type-static-cast-downcast**
Suppress warnings for use of `static_cast` to downcast event data in transition functions

```
transition2_t1( static_cast< const bool * > ( msg->data ), static_cast< P::Base * > ( msg->sap()  
/* NOLINT(cppcoreguidelines-pro-type-static-cast-downcast) */ ) );
```

- **misc-unused-parameters**

Suppress warnings for named function parameters that are not used in the function body

```
static void rtg_B_init(const RTObject_class * type /* NOLINT(misc-unused-parameters) */, B * target );
```



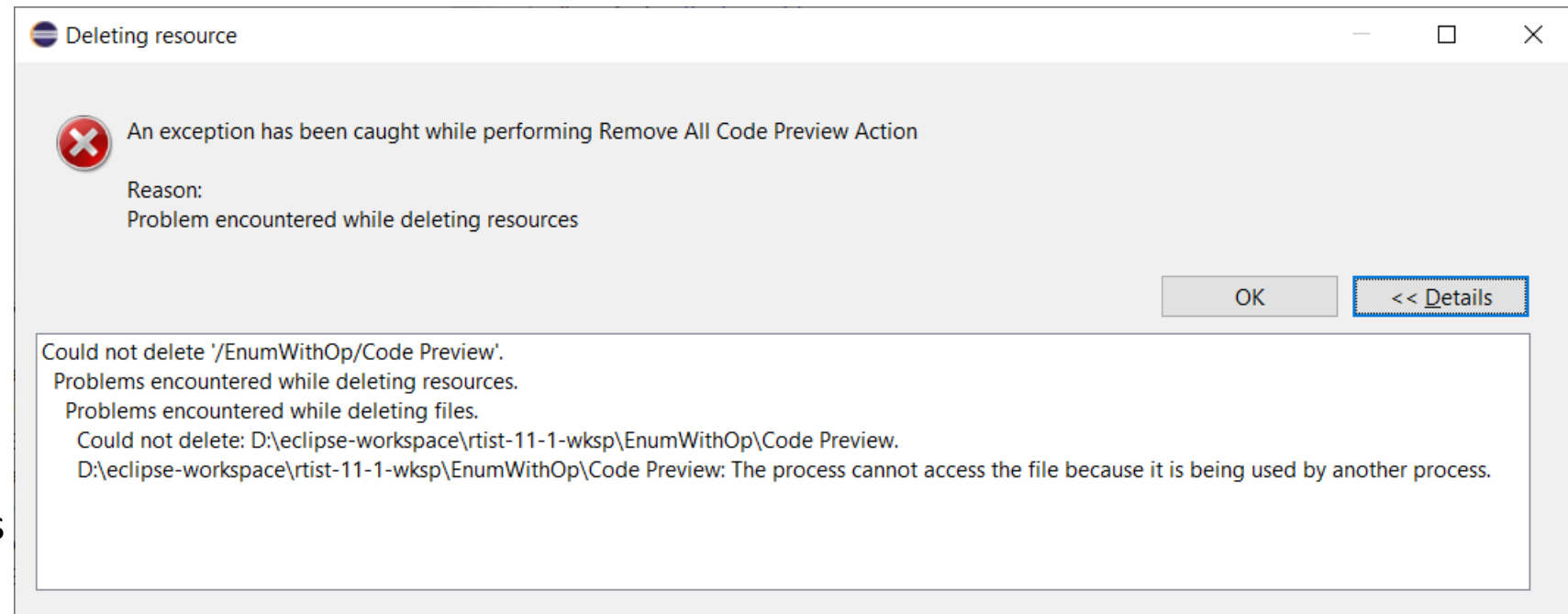
Error Message when Failing to Delete Files or Folders

- ▶ Certain commands in RTist involve deletion of files and/or folders

- Cleaning a TC
- Removing code preview
- ...etc

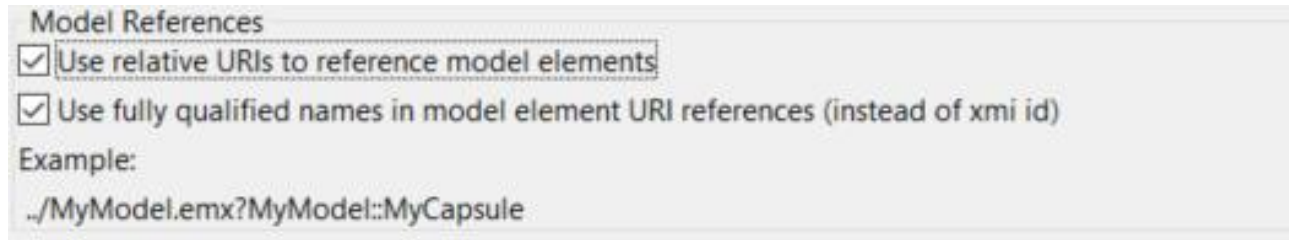
- ▶ Now, if the required files or folders cannot be deleted, a clear error message is shown

- Previously there would be a silent failure in such situations which could be hard to understand the reason for
- The new message is identical to what Eclipse would show if you directly try to remove the files/folders from the Project Explorer. Click the **Details** button to see exactly which file or folder that couldn't be deleted, and why.



More Flexible Model References in Transformation Configurations

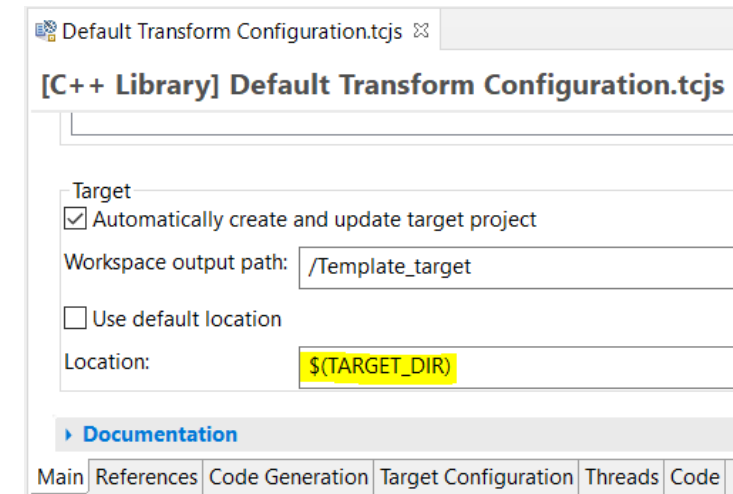
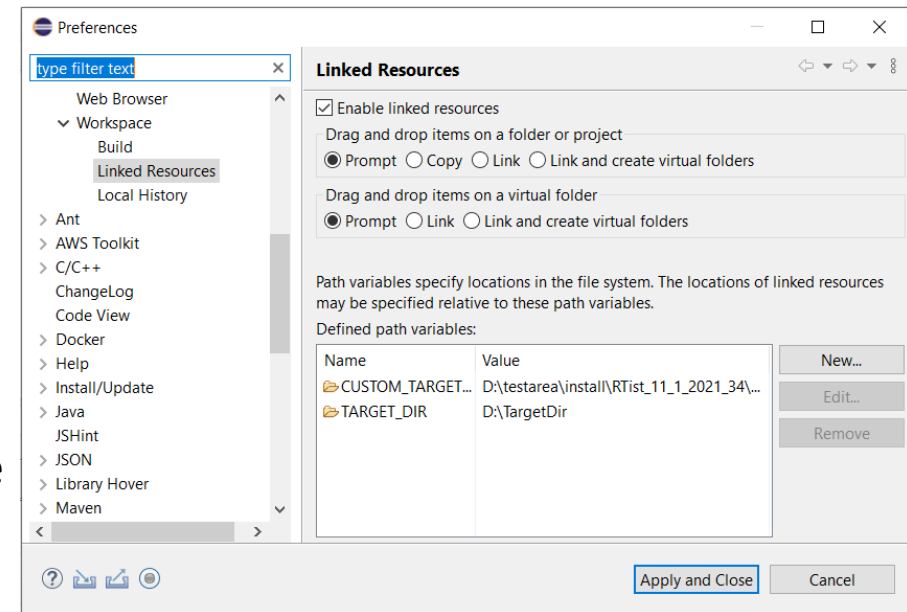
- ▶ A TC references model elements by means of URIs (e.g. list of source elements, top capsule etc)
- ▶ Such URIs can now be relative, and use qualified names instead of unique IDs to identify the element
 - Makes it easier to reuse a TC (e.g. by copy/paste) in different projects
- ▶ New preferences control how new URIs will be created:
RealTime Development – Transformation Configuration Editor – Model References



Support for Path Variables in Transformation Configurations

- ▶ Path variables can now be used in certain TC properties
 - Useful for those TC properties that specify a path
 - Define path variables in Preferences at **General – Workspace – Linked Resources**
 - This can be an alternative to using string substitutions (**Run/Debug – String Substitutions**) or environment variables in order to have a more generic TC (a path variable takes precedence over other kinds of variables, if the same variable name is used).
- ▶ The model compiler now prints a warning if a variable used in a TC property cannot be resolved

WARNING : Cannot resolve variable '\$(TARGET_DIR)' in 'Location' property:'\$(TARGET_DIR)'



- ▶ [Mocha](#) is a popular JavaScript framework for testing asynchronous applications
- ▶ It's now possible to use Mocha also for unit testing capsules
 - Provided by a new component that can be selected when installing
 - Note that it depends on NodePlus

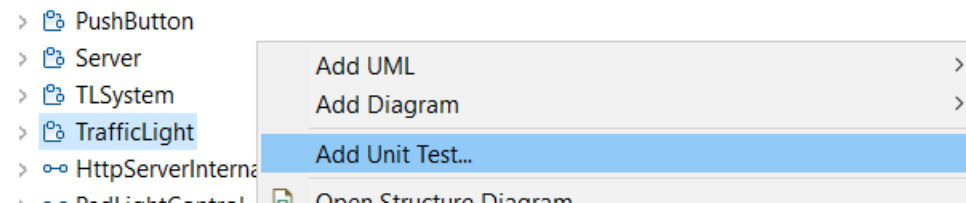


simple, flexible, fun

- > NodePlus
- > RTist Core
- > RTist Extra Functionality
- ▼ RTist Integrations
 - OneTest Embedded Integration

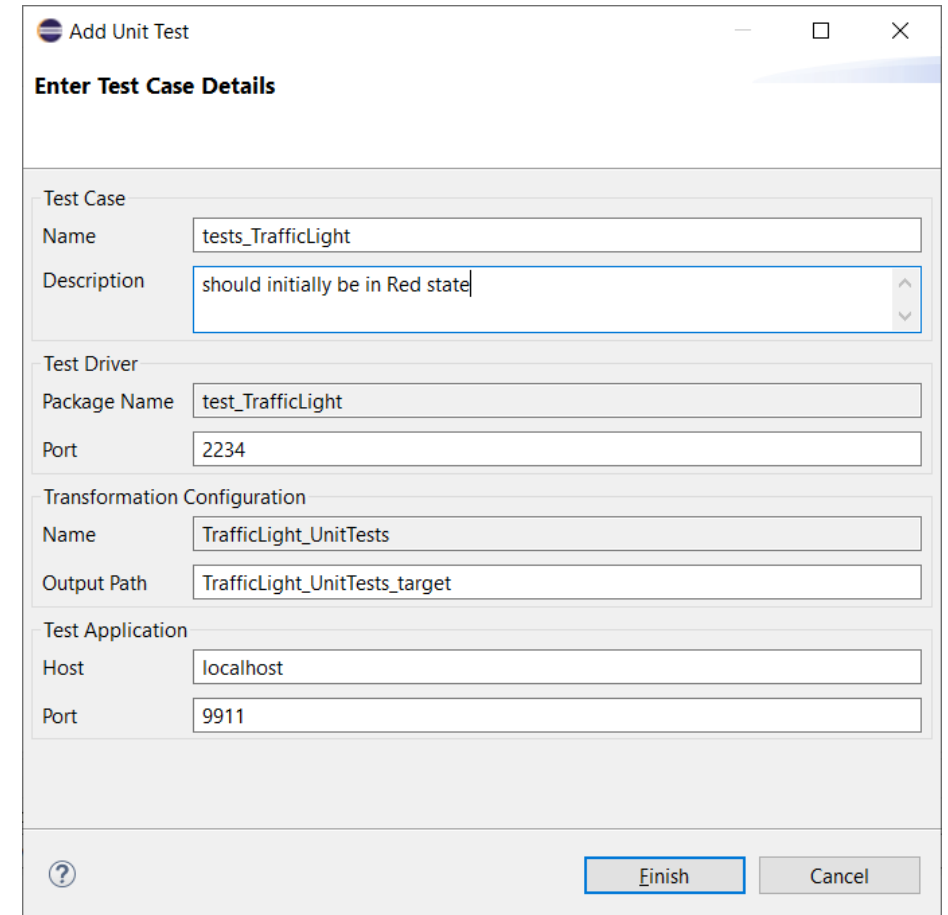
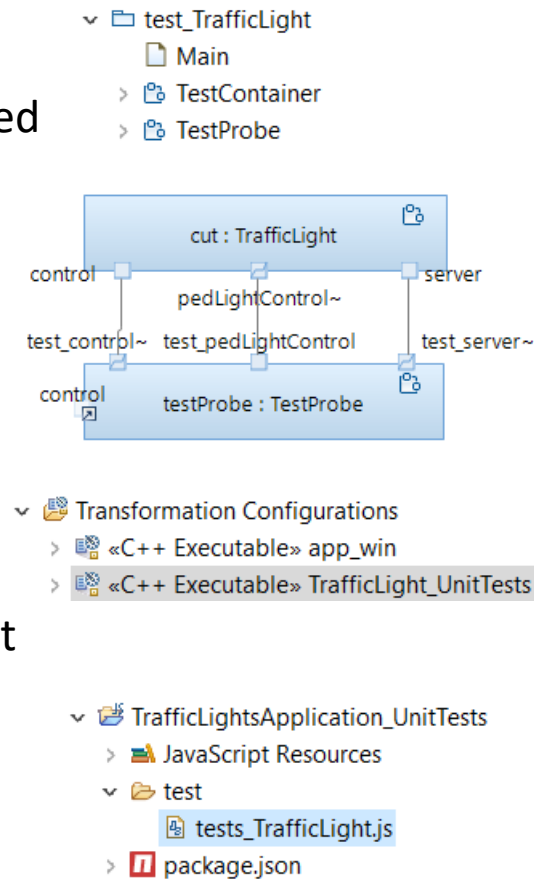
11.1.0.v20210924_0511

- ▶ To create a Mocha unit test for a capsule, invoke the new context menu command **Add Unit Test**



► The **Add Unit Test** command creates everything necessary for writing a unit test for the capsule

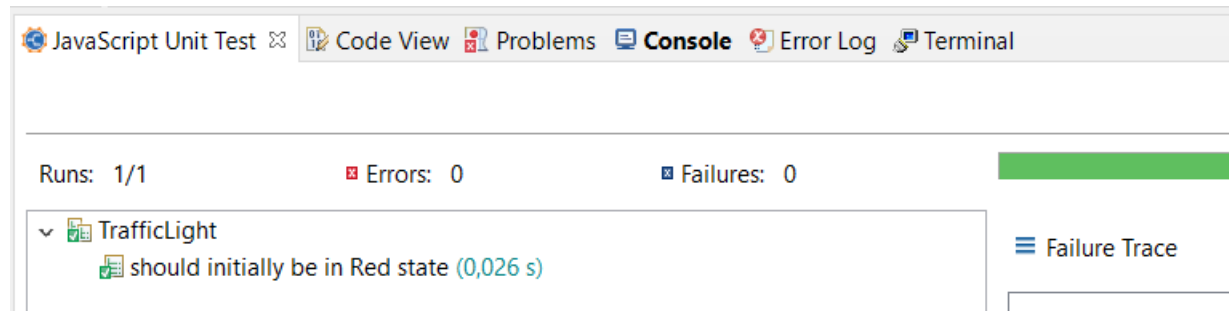
- A test driver model where all service ports of the capsule under test ("cut") are connected to similar but conjugated ports of a test probe capsule
- A TC for building the test driver model into an executable that uses the TcpServer library for exposing all test probe ports to the Mocha test script
- A Node.js project with a Mocha test script ready to implement the unit test



- ▶ The unit test can be executed right away
 - Build the test driver TC (only needed the first time, and whenever you change the capsule under test)
 - Install the Node.js dependencies for the JavaScript project (right-click on the project and do **Run As – npm install** (only needed the first time – it is assume you already have installed Mocha on the machine)
 - Run the testcase by right-click on the .js file and do **Run As – JavaScript Unit Test**

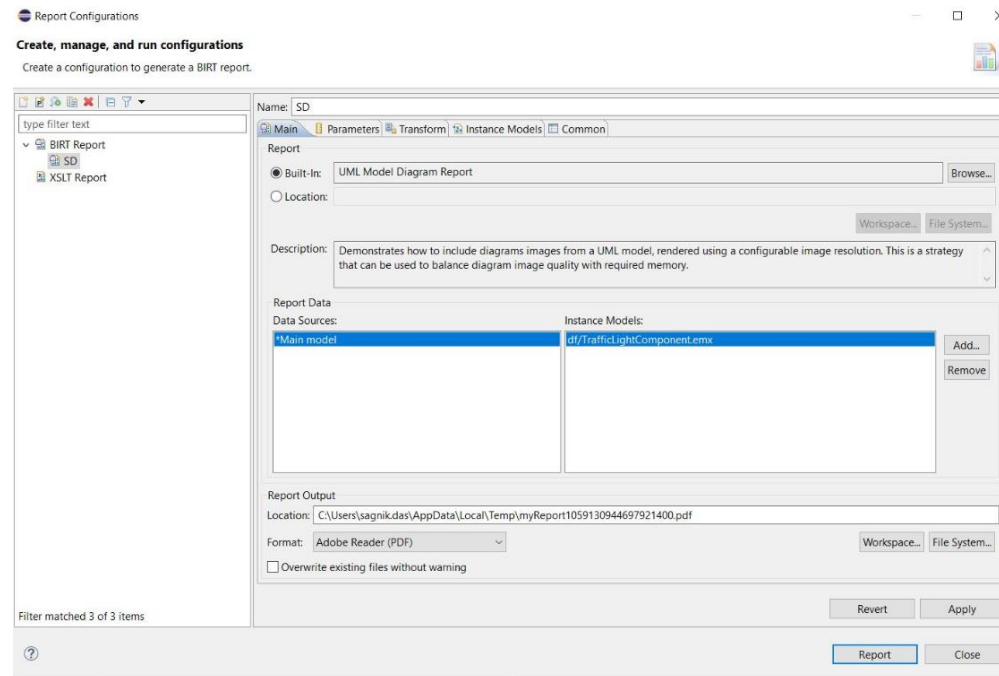
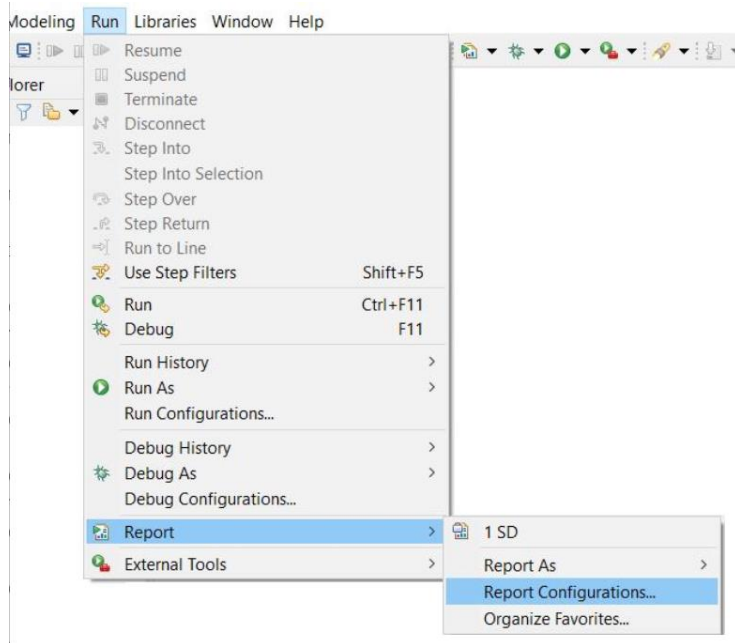
```
tests_TrafficLight.js | TestContainer | TrafficLight_UnitTests.tcjs
1 var assert = require('assert');
2 describe('TrafficLight', function() {
3     it('should initially be in Red state', function() {
4         this.timeout(15000);
5         const testProbe = require('rt-test-probe')('localhost', 9911);
6         return testProbe.startListenForEvents(2234)
7             .then((data) => {
8                 // TODO: Implement test here
9             })
10        .finally(() => {
11            testProbe.stopListenForEvents();
12        });
13    });
14 });
```

- ▶ The test execution result is shown in the **JavaScript Unit Test** view



Reporting with BIRT

- ▶ Create reports that include information from an RTist model
 - Same capabilities as in RTist 10.3, but now adapted for recent Eclipse versions (supports RTist 11.0 and RTist 11.1)
 - Delivered as a separate update site on [our InfoCenter](#). Installation instructions are included in the ZIP file.
 - This is currently an experimental feature



Diagrams for model: TrafficLightComponent

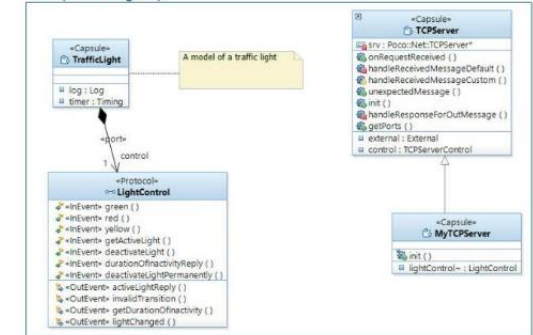
Introduction

No documentation available.

Model Diagrams

TrafficLightComponent

Main (Class Diagram)



HCL

*Relationship*TM
BEYOND THE CONTRACT

\$7 BILLION ENTERPRISE | 110,000 IDEAPRENEURS | 31 COUNTRIES

 WATCH THE FILM