HCL

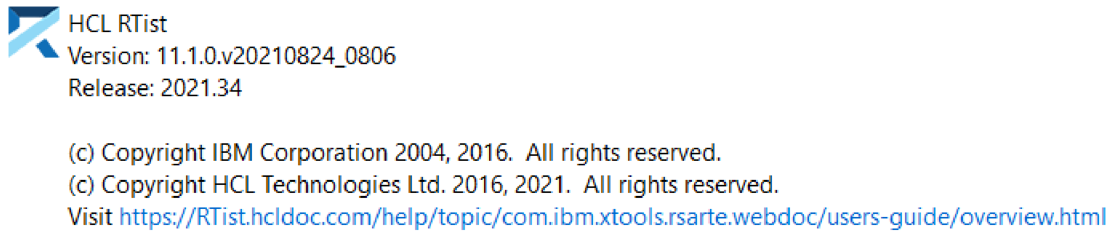
What’s New in HCL RTist 11.1

updated for release 2021.34

HCL SOFTWARE

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 RTist only


HCL RTist
Version: 11.1.0.v20210824_0806
Release: 2021.34

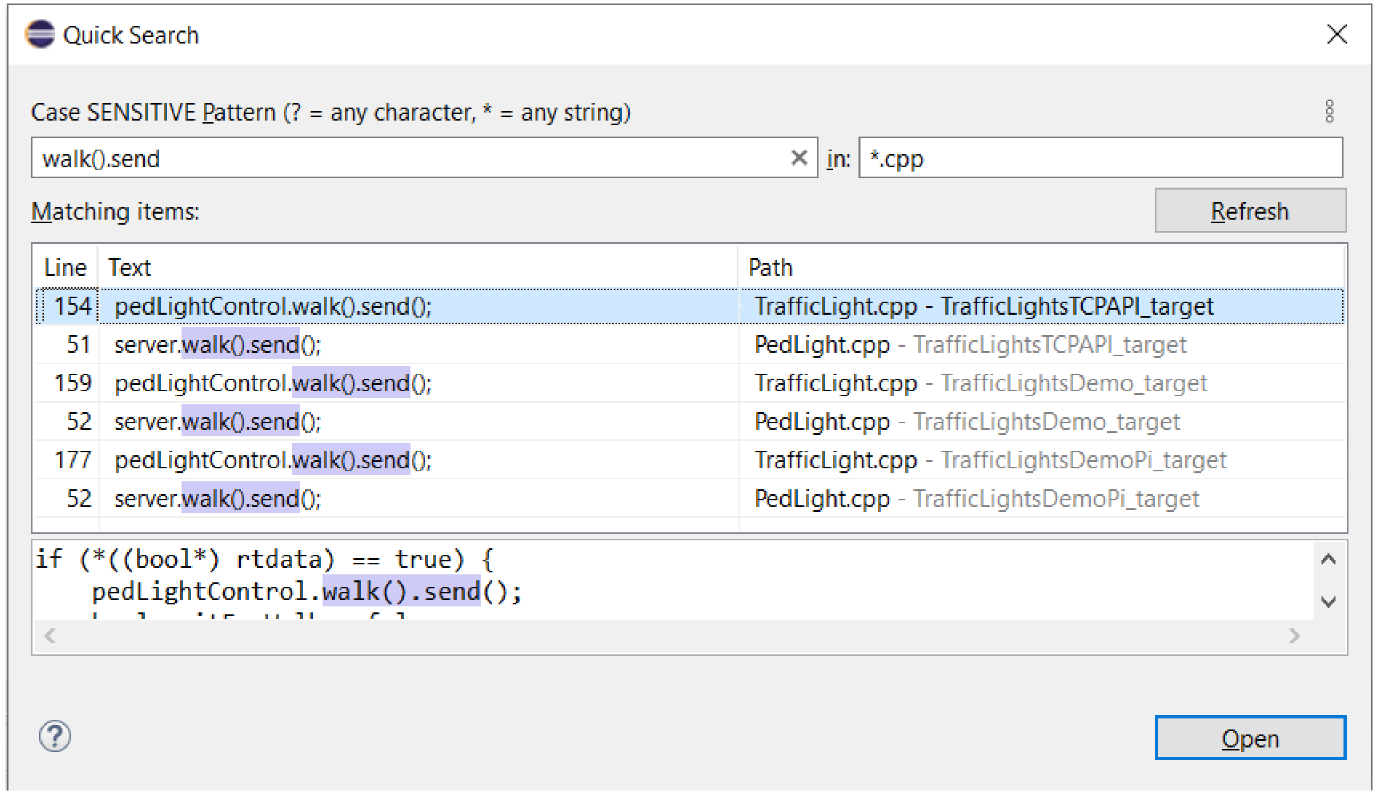
(c) Copyright IBM Corporation 2004, 2016. All rights reserved.
(c) Copyright HCL Technologies Ltd. 2016, 2021. All rights reserved.
Visit https://RTist.hcldoc.com/help/topic/com.ibm.xtools.rsarte.webdoc/users-guide/overview.html

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



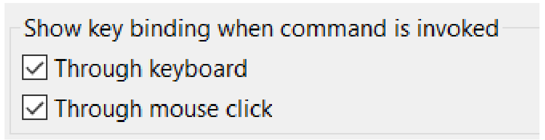
Line	Text	Path
154	pedLightControl.walk().send();	TrafficLight.cpp - TrafficLightsTCPAPI_target
51	server.walk().send();	PedLight.cpp - TrafficLightsTCPAPI_target
159	pedLightControl.walk().send();	TrafficLight.cpp - TrafficLightsDemo_target
52	server.walk().send();	PedLight.cpp - TrafficLightsDemo_target
177	pedLightControl.walk().send();	TrafficLight.cpp - TrafficLightsDemoPi_target
52	server.walk().send();	PedLight.cpp - TrafficLightsDemoPi_target

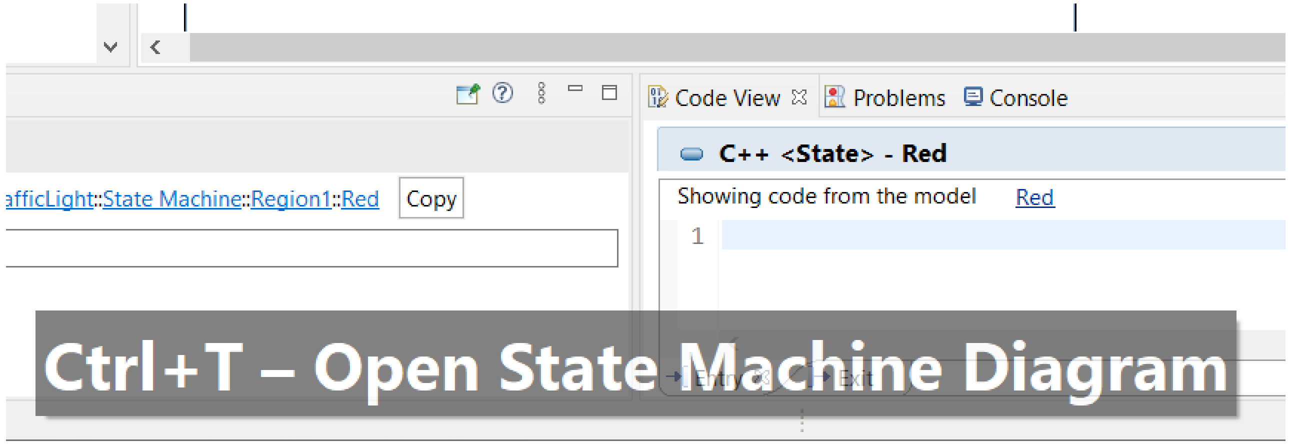
```
if (*((bool*) rtdata) == true) {
    pedLightControl.walk().send();
```

HCL SOFTWARE

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

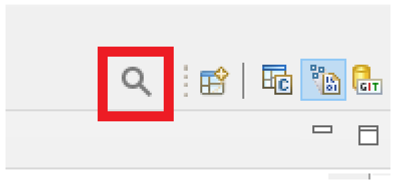

Show key binding when command is invoked
☑ Through keyboard
☑ Through mouse click

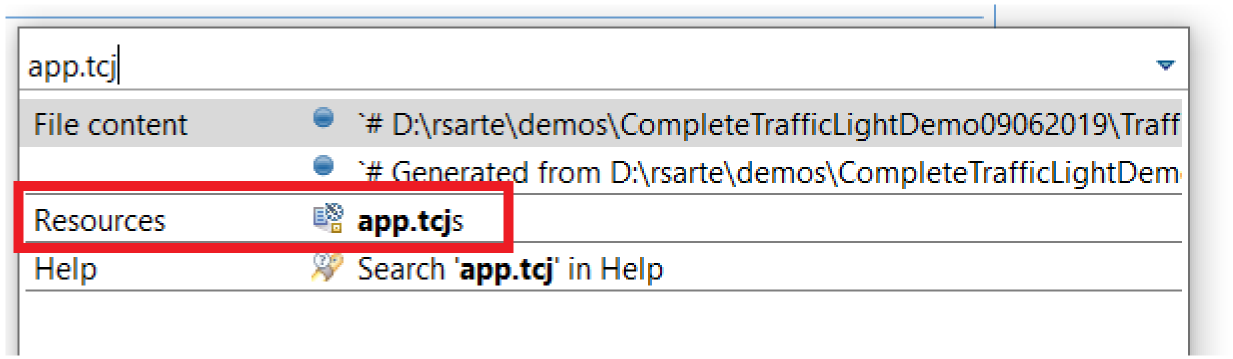

Code View
Problems
Console
C++ <State> - Red
Showing code from the model Red
afficLight::State Machine::Region1::Red
Copy
Entry
Exit
Ctrl+T – Open State Machine Diagram

HCL SOFTWARE

Eclipse 4.16 (2020.06)

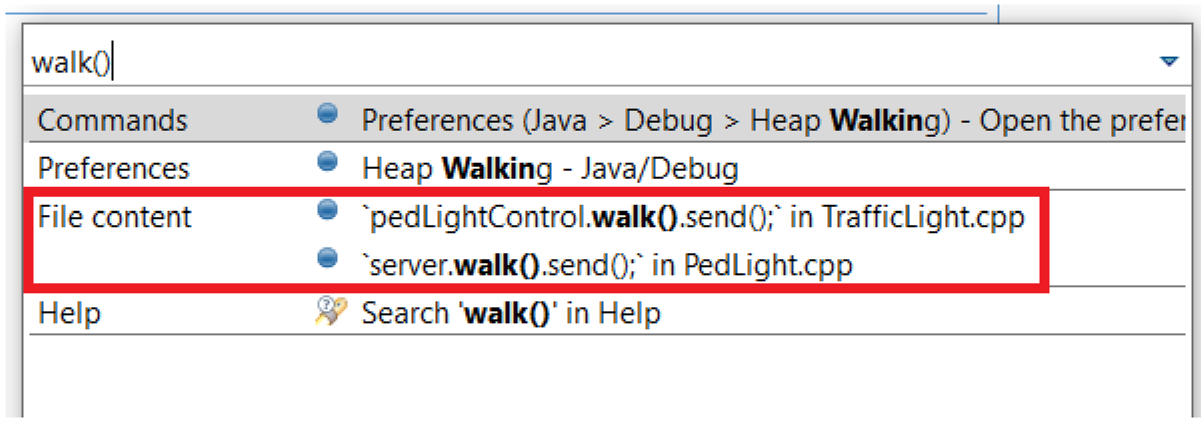
- Quick Access field replaced with toolbar button
  - Takes less space in the toolbar, and instead uses a normal dialog for typing an showing the results
  - Same key binding as before (Ctrl + 3) but the command is now called "Find Actions"
  - The results now also include matching files in the workspace,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





app.tcj

File content	`# D:\rsarte\demos\CompleteTrafficLightDemo09062019\Traff
	`# Generated from D:\rsarte\demos\CompleteTrafficLightDem
Resources	**app.tcj**s
Help	Search '**app.tcj**' in Help



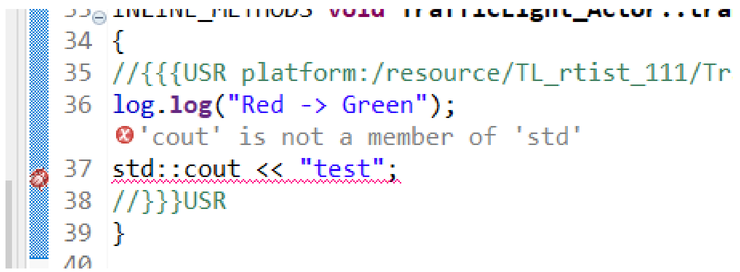
walk()

Commands	Preferences (Java > Debug > Heap **Walking**) - Open the prefer
Preferences	Heap **Walking** - Java/Debug
File content	`pedLightControl.**walk()**.send();` in TrafficLight.cpp
	`server.**walk()**.send();` in PedLight.cpp
Help	Search '**walk()**' in Help

HCL SOFTWARE

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**

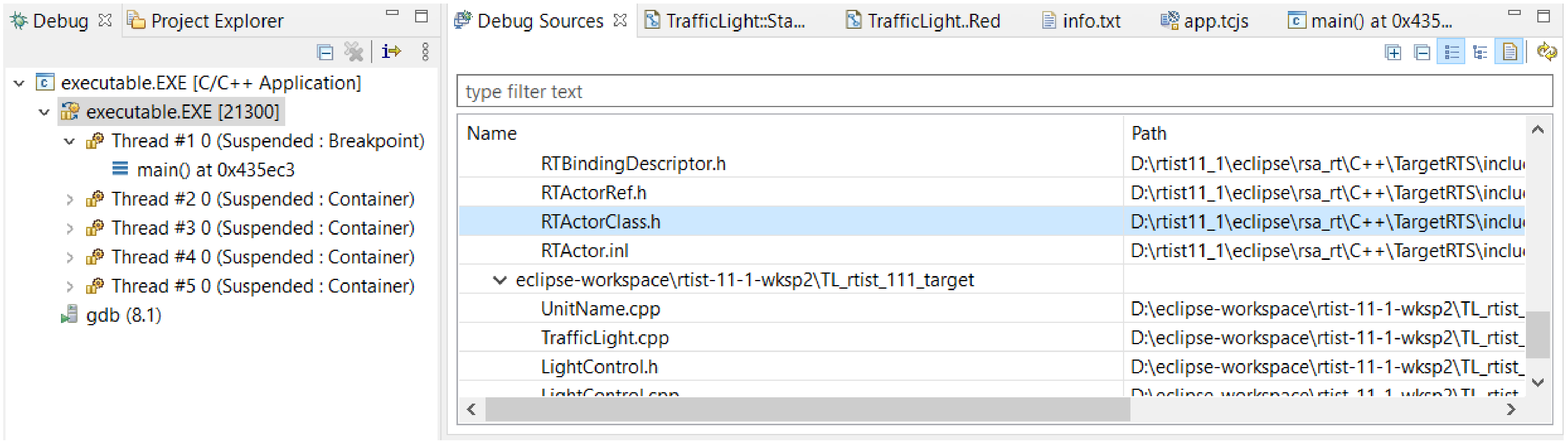


```
INLINE_METHODS void TrafficLight_Actor::tra
{
//{{{USR platform:/resource/TL_rtist_111/Tr
log.log("Red -> Green");
'cout' is not a member of 'std'
std::cout << "test";
//}}}USR
}
```

- There were several improvements in SWT and GTK
  - The minimal supported GTK version is now 3.20

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



```
executable.EXE [C/C++ Application]
  executable.EXE [21300]
    Thread #1 0 (Suspended : Breakpoint)
      main() at 0x435ec3
    Thread #2 0 (Suspended : Container)
    Thread #3 0 (Suspended : Container)
    Thread #4 0 (Suspended : Container)
    Thread #5 0 (Suspended : Container)
  gdb (8.1)
```

Name	Path
RTBindingDescriptor.h	D:\rtist11_1\eclipse\rsa_rt\C++\TargetRTS\inclu
RTActorRef.h	D:\rtist11_1\eclipse\rsa_rt\C++\TargetRTS\inclu
RTActorClass.h	D:\rtist11_1\eclipse\rsa_rt\C++\TargetRTS\inclu
RTActor.inl	D:\rtist11_1\eclipse\rsa_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

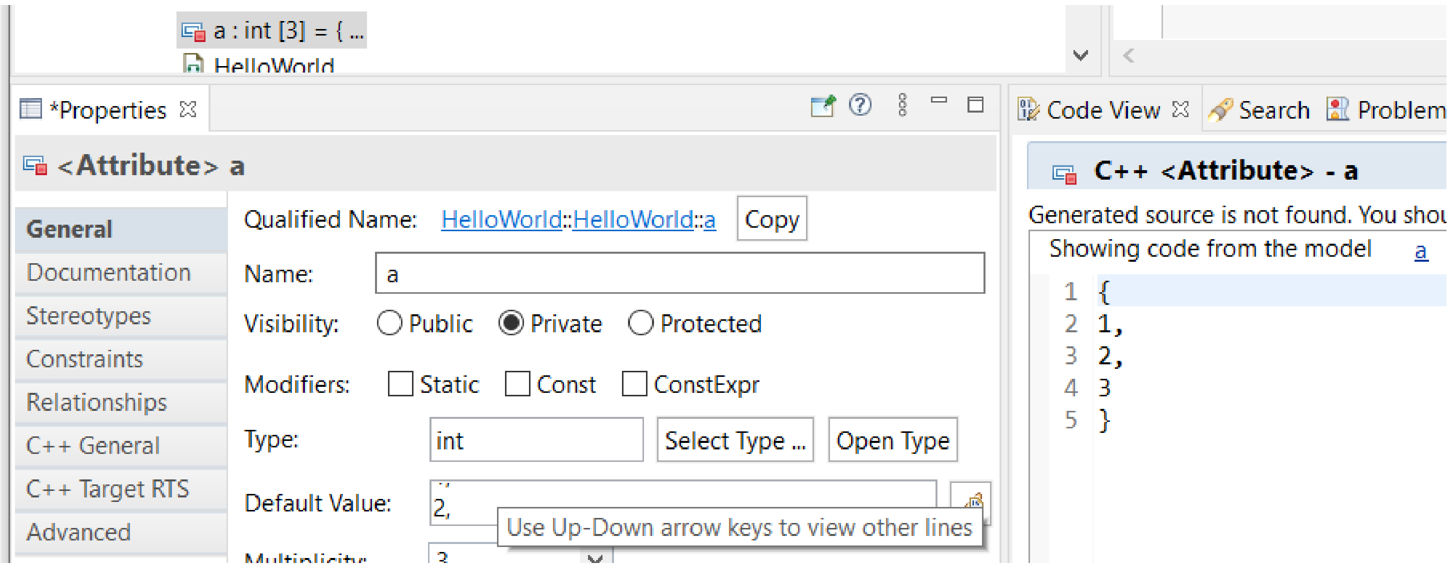
- The EGit integration in RTist has upgraded EGit from 5.4 to 5.8
  - This is the recommended and latest version for Eclipse 2020.06
- This upgrade provides several new features, performance improvements and bug fixes
  - For detailed information about the changes see
    https://wiki.eclipse.org/EGit/New_and_Noteworthy/5.5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



a : int [3] = { ...
HelloWorld

*Properties

<Attribute> a

General | Documentation | Stereotypes | Constraints | Relationships | C++ General | C++ Target RTS | Advanced

Qualified Name: HelloWorld::HelloWorld::a [Copy]
Name: a
Visibility: ( ) Public [✓ Private] ( ) Protected
Modifiers: [ ] Static [ ] Const [ ] ConstExpr
Type: int [Select Type ...] [Open Type]
Default Value: 2,
Use Up-Down arrow keys to view other lines
Multiplicity: 3

Code View | Search | Problem

C++ <Attribute> - a

Generated source is not found. You shou
Showing code from the model a

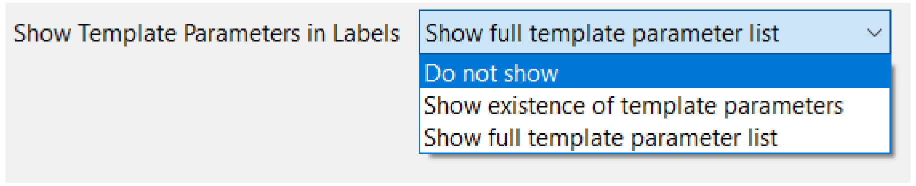
```
{
1,
2,
3
}
```

  - For quickly viewing a multi-line default value the Properties view can be handy

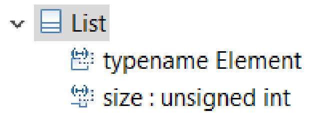
HCL SOFTWARE

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

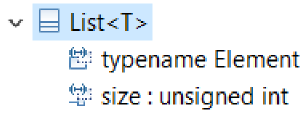

Show Template Parameters in Labels: Show full template parameter list

- Do not show
- Show existence of template parameters
- Show full template parameter list



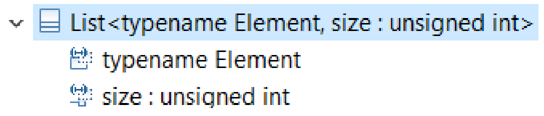
```
List
    typename Element
    size : unsigned int
```

Do not show template parameters



```
List<T>
    typename Element
    size : unsigned int
```

Show existance of template parameters

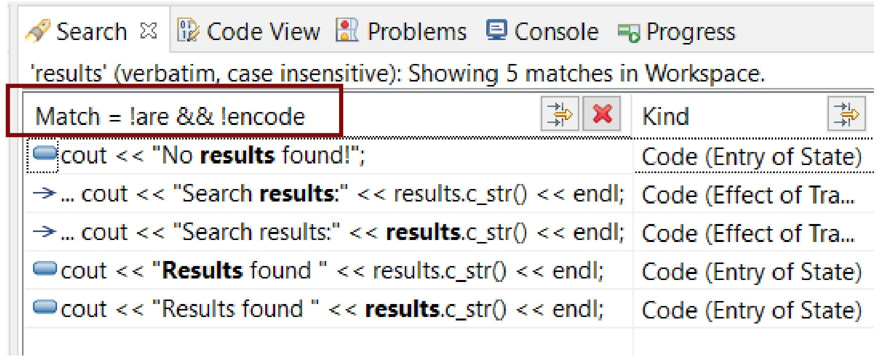


```
List<typename Element, size : unsigned int>
    typename Element
    size : unsigned int
```

Show full template parameter list

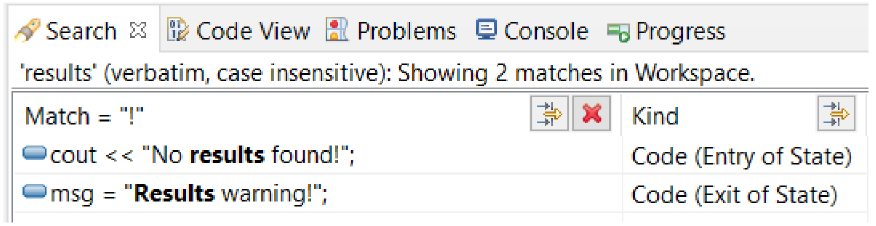
Search Filtering

- 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 <u>are not</u> present
  - Use a filter on the form `A && B && ... X` to only show matches where certain words <u>are</u> 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 &&



'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)

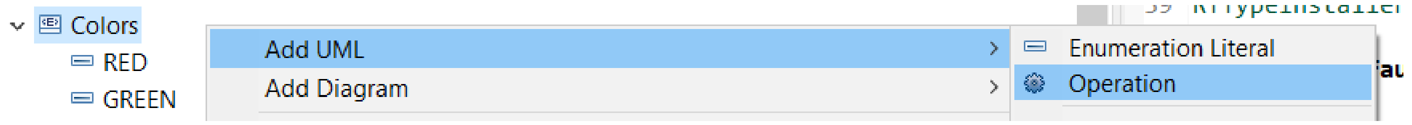


'results' (verbatim, case insensitive): Showing 2 matches in Workspace.

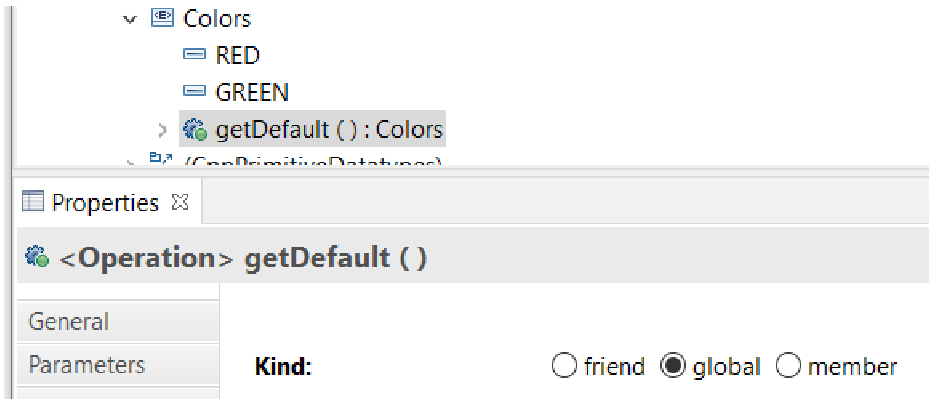
Match = "!"	Kind
cout << "No **results** found!";	Code (Entry of State)
msg = "**Results** warning!";	Code (Exit of State)

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



<Operation> getDefault ( )

Kind: friend [✓ global] member

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

HCL SOFTWARE

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:

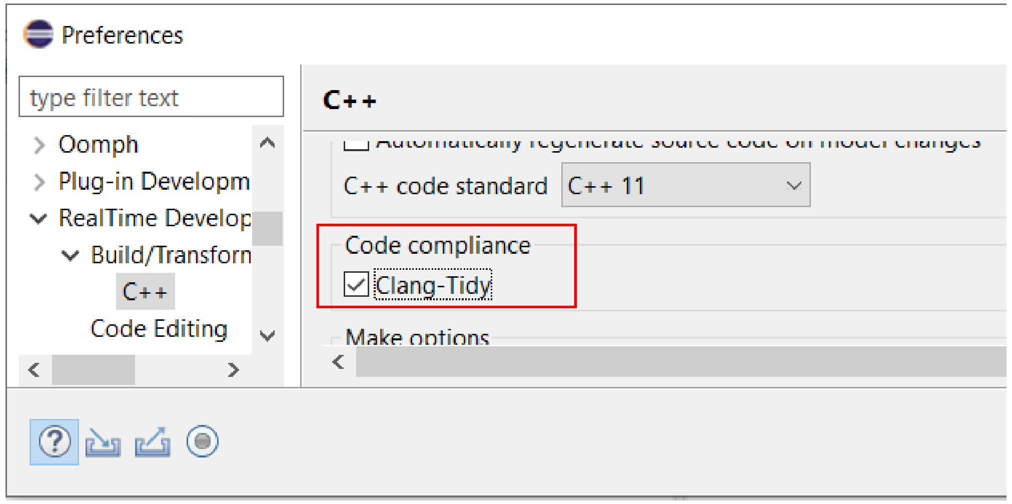
```
template <> inline const RTObject_class* RTObject_class::fromType<RTString>() {
    return &RTType_RTString;
}
```

- 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.

HCL SOFTWARE

Code Compliance

- A new group of preferences were introduced to let the model compiler generate code according to certain code compliance rules
  - As a first step support for one specific Clang-Tidy rule is implemented
  - It suppresses warnings for use of static_cast to downcast event data in transition functions



C++ code standard: C++ 11

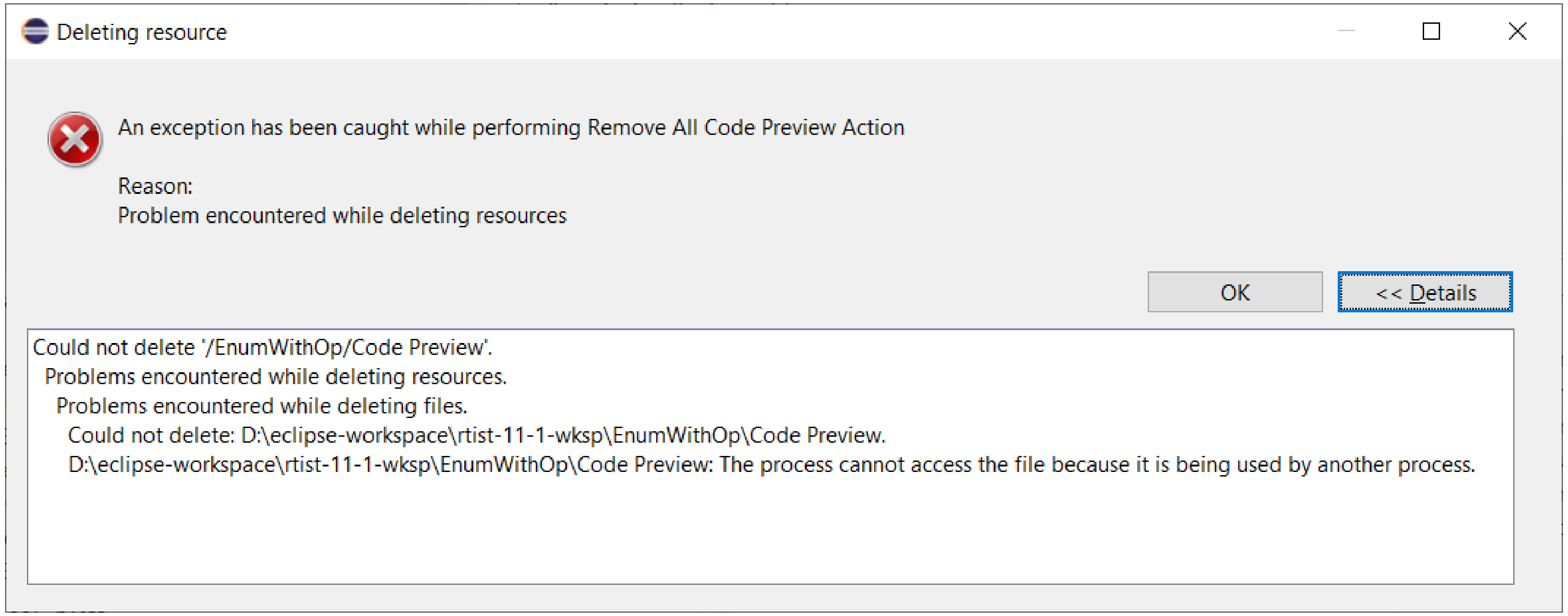
Code compliance

- [✓] Clang-Tidy

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

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.



An exception has been caught while performing Remove All Code Preview Action

Reason:
Problem encountered while deleting resources

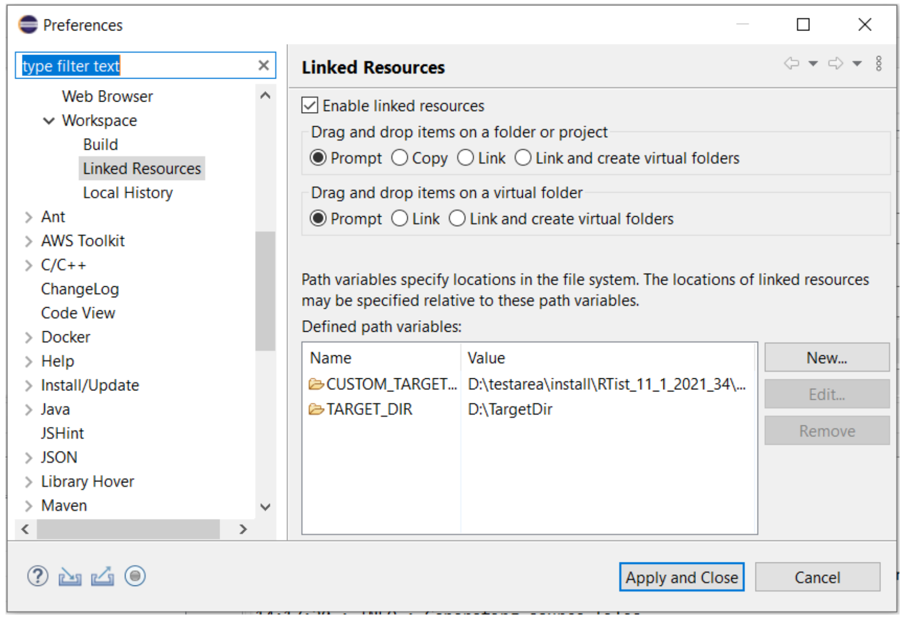
```
Could not delete '/EnumWithOp/Code Preview'.
 Problems encountered while deleting resources.
  Problems encountered while deleting files.
   Could not delete: D:\eclipse-workspace\rtist-11-1-wksp\EnumWithOp\Code Preview.
   D:\eclipse-workspace\rtist-11-1-wksp\EnumWithOp\Code Preview: The process cannot access the file because it is being used by another process.
```

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)'
```



Linked Resources

☑ Enable linked resources

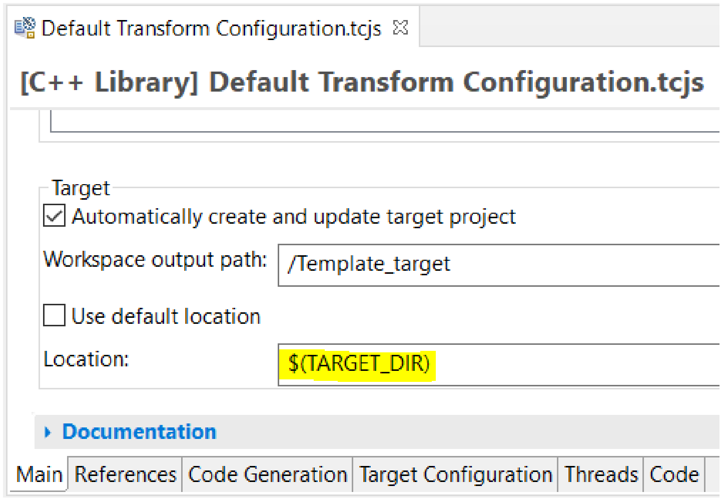
Drag and drop items on a folder or project
[✓ Prompt] Copy Link Link and create virtual folders

Drag and drop items on a virtual folder
[✓ Prompt] Link Link and create virtual folders

Path variables specify locations in the file system. The locations of linked resources may be specified relative to these path variables.

Defined path variables:

Name	Value
CUSTOM_TARGET...	D:\testarea\install\RTist_11_1_2021_34\...
TARGET_DIR	D:\TargetDir



[C++ Library] Default Transform Configuration.tcjs

Target
☑ Automatically create and update target project
Workspace output path: /Template_target
☐ Use default location
Location: $(TARGET_DIR)

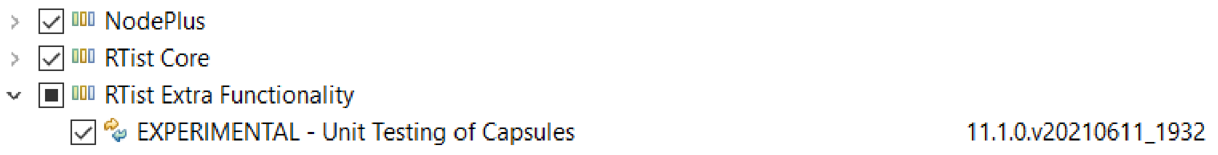
HCL SOFTWARE

Unit Testing of Capsules using the Mocha Framework (1/3)

RTist only

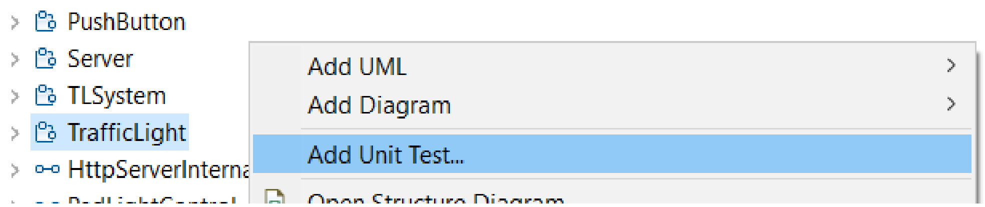
- Mocha is a popular JavaScript framework for testing asynchronous applications
- It's now possible to use Mocha also for unit testing capsules
  - Provided by a new component that can be selected when installing
  - Note that it depends on NodePlus and is currently an EXPERIMENTAL feature

MOCHA simple, flexible, fun



```
> ☑ NodePlus
> ☑ RTist Core
v ■ RTist Extra Functionality
    ☑ EXPERIMENTAL - Unit Testing of Capsules                11.1.0.v20210611_1932
```

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

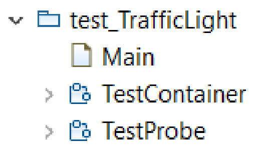


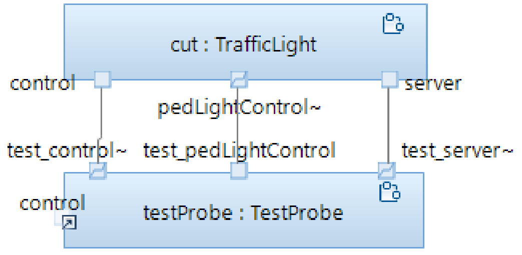
HCL SOFTWARE

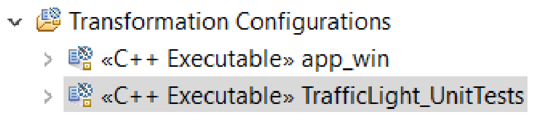
Unit Testing of Capsules using the Mocha Framework (2/3)

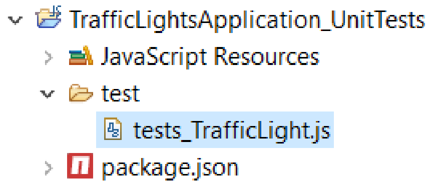
RTist only

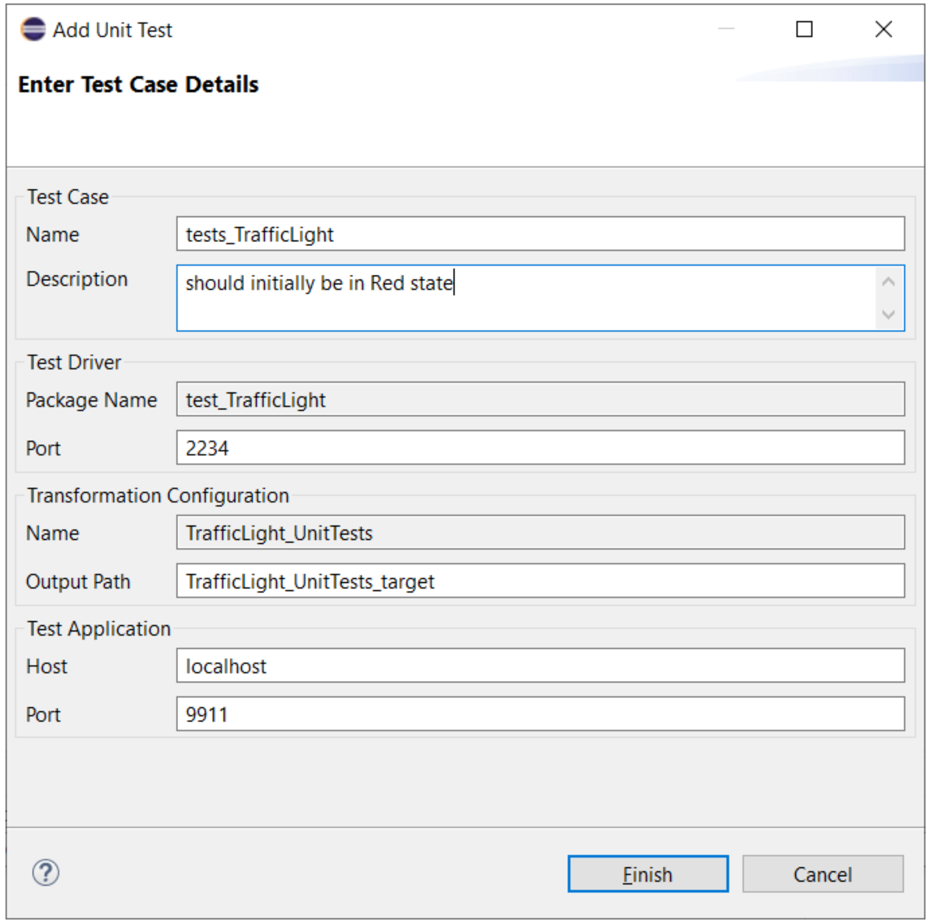
- The **Add Unit Test** command creates everything necessary for writing a unit test for the capsule
  - A test driver model where all service ports of the capsule under test (”cut”) are connected to similar but conjugated ports of a test probe capsule
  - A TC for building the test driver model into 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






Transformation Configurations
«C++ Executable» app_win
«C++ Executable» TrafficLight_UnitTests





Enter Test Case Details

Test Case

Name	tests_TrafficLight
Description	should initially be in Red state

Test Driver

Package Name	test_TrafficLight
Port	2234

Transformation Configuration

Name	TrafficLight_UnitTests
Output Path	TrafficLight_UnitTests_target

Test Application

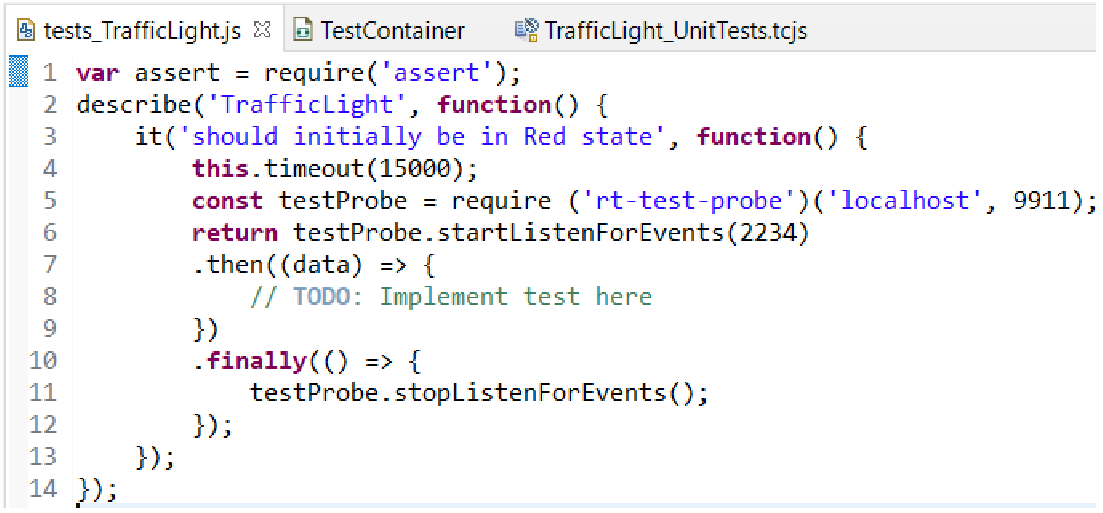
Host	localhost
Port	9911

HCL SOFTWARE

Unit Testing of Capsules using the Mocha Framework (3/3)

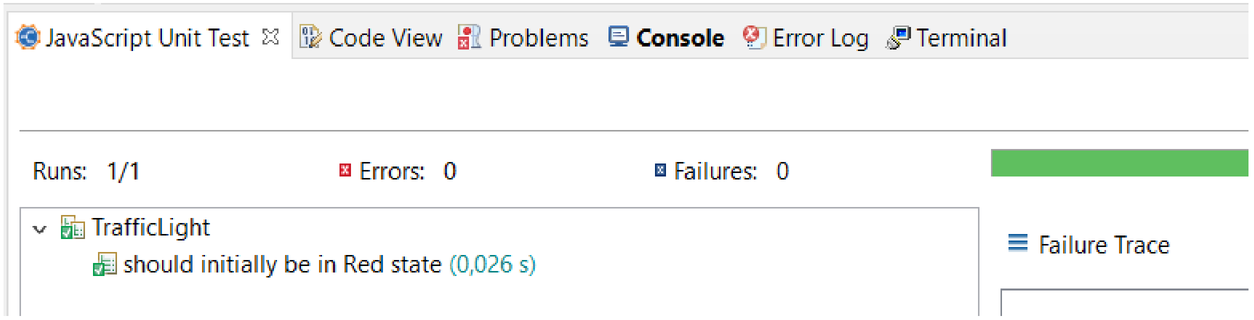
RTist only

- 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**



```
var assert = require('assert');
describe('TrafficLight', function() {
    it('should initially be in Red state', function() {
        this.timeout(15000);
        const testProbe = require ('rt-test-probe')('localhost', 9911);
        return testProbe.startListenForEvents(2234)
        .then((data) => {
            // TODO: Implement test here
        })
        .finally(() => {
            testProbe.stopListenForEvents();
        });
    });
});
```

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

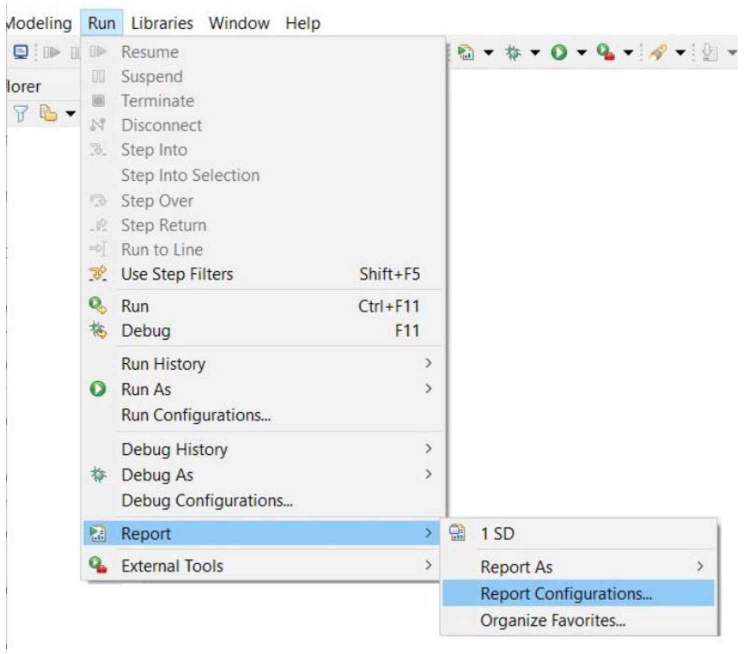


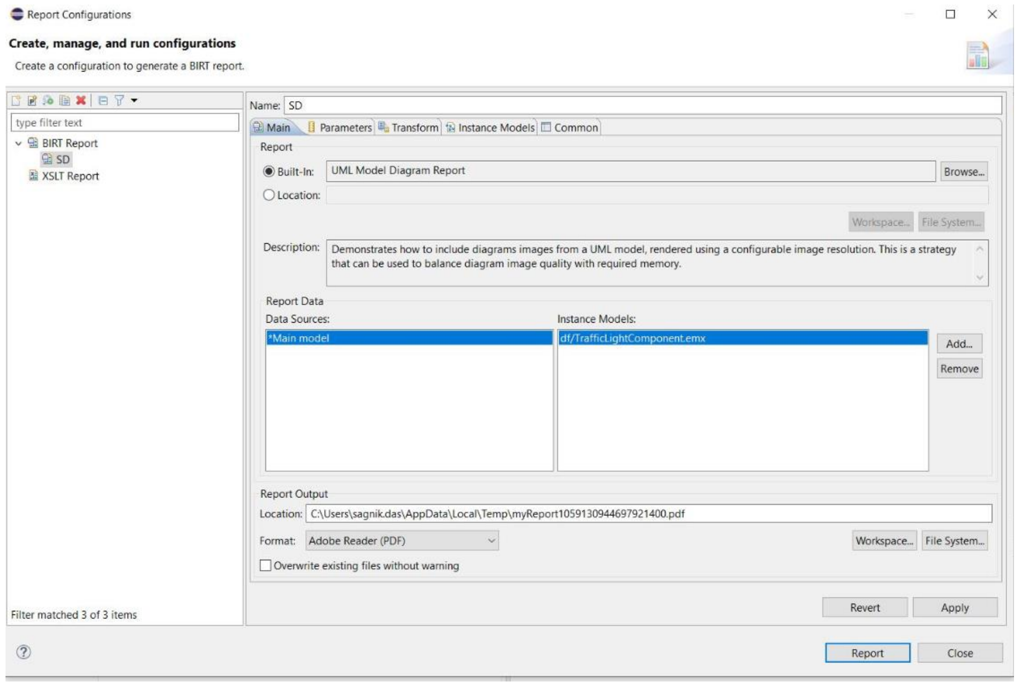
```
Runs: 1/1        Errors: 0        Failures: 0
TrafficLight
    should initially be in Red state (0,026 s)
```

HCL SOFTWARE

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




Report Configurations

Create, manage, and run configurations

Create a configuration to generate a BIRT report.

type filter text
- BIRT Report
  - SD
- XSLT Report

Filter matched 3 of 3 items

Name: SD

Main | Parameters | Transform | Instance Models | Common

Report
- (●) Built-In: UML Model Diagram Report
- ( ) Location:

Description: Demonstrates how to include diagrams images from a UML model, rendered using a configurable image resolution. This is a strategy that can be used to balance diagram image quality with required memory.

Report Data

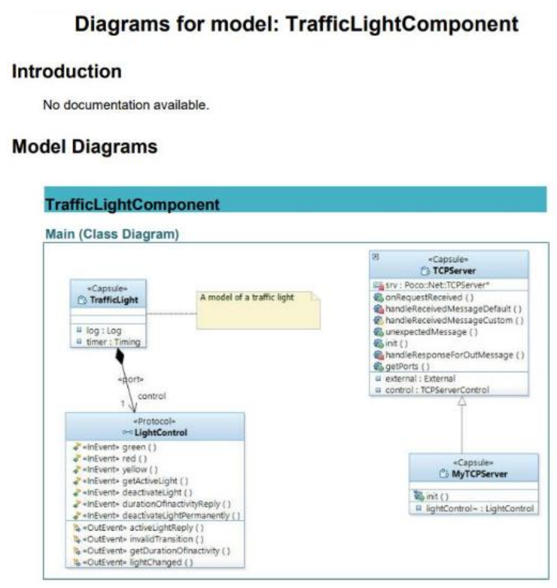
Data Sources:	Instance Models:
*Main model	df/TrafficLightComponent.emx

Report Output

Location: C:\Users\sagnik.das\AppData\Local\Temp\myReport1059130944697921400.pdf

Format: Adobe Reader (PDF)

[ ] Overwrite existing files without warning


Diagrams for model: TrafficLightComponent

Introduction

No documentation available.

Model Diagrams

HCL SOFTWARE

HCL

Relationship™

BEYOND THE CONTRACT

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

WATCH THE **FILM**