EventB XText Front-end User Manual

Thai Son Hoang
ECS, University of Southampton
T.S.Hoang at ecs dot soton dot ac dot uk

Version 0.0.1
(for feature version 0.0.5)
Friday 20th January, 2017

1 Event-B XText Front-end Overview

The Event-B XText Front-end provides new XEvent-B constructs (XMachines and XContexts) which are text files which are automatically translated into the corresponding Rodin Event-B constructs (i.e., Machine and Context) accordingly. Facility for translating from Rodin Event-B components to XEvent-B components can be invoked manually. The overall process can be seen in Figure 1.

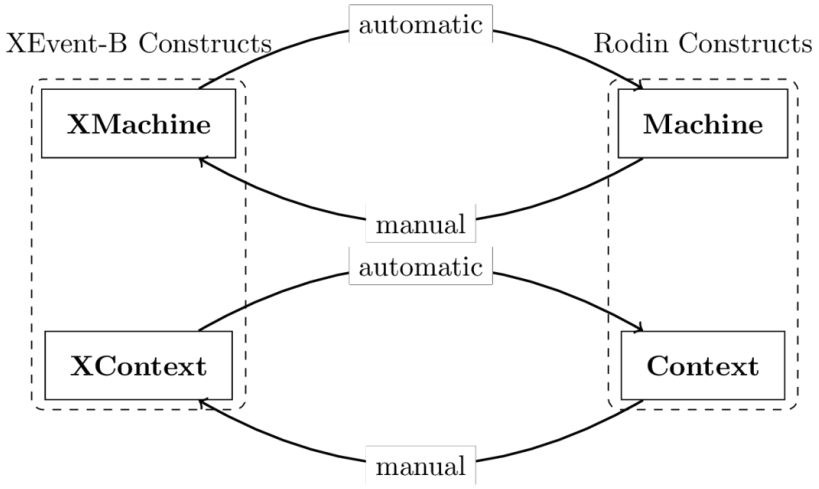

Figure 1: Overview of XEvent-B and Rodin Event-B Constructs

2 Getting Started

2.1 Installation

2.1.1 Setup

- Before installing the Event-B XText Front-end feature, you need to add the XText update site (`http://download.eclipse.org/modeling/tmf/xtext/updates/composite/releases/`) as an additional software site (see Figure 2).

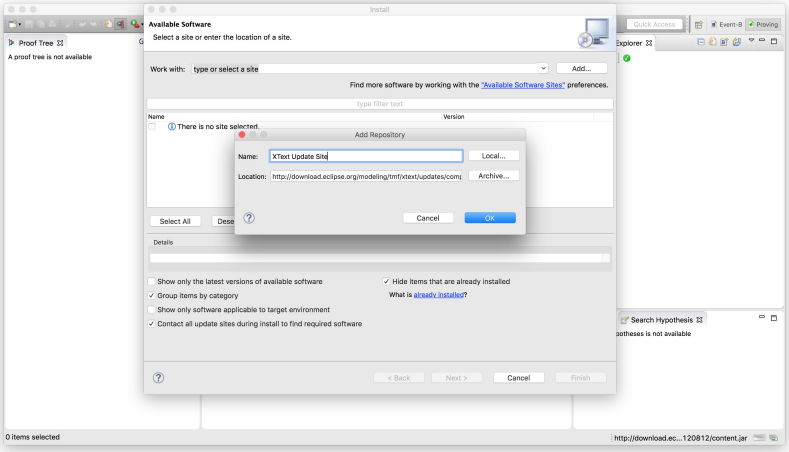

Figure 2: Adding XText Update Site

- The Event-B XText front-end feature is available from the main Rodin update site (under "Editors" category). There are two versions of the feature, *Event-B XText* providing facilities for working with Event-B XText Front-end, and the *Event-B XText (SDK)* is the feature including source code for software developers (see Figure 3).

2.1.2 Release Notes

0.0.5

- Event-B XText Documentations (0.0.1): Documentation plug-in (Initial version).

0.0.4

- Updated plug-in dependency for the feature

0.0.3

- Event-B XText Context (0.0.3):
  - Issue #3: Single-line comment after the element, multi-line comment before the element

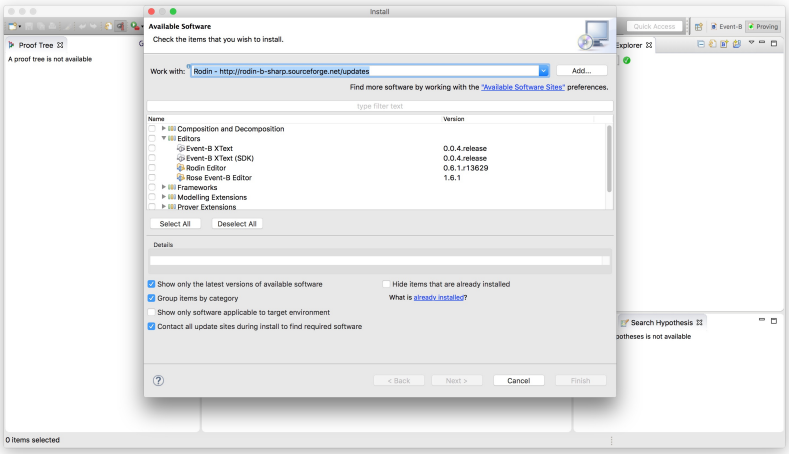

Figure 3: Adding XText Update Site

- Event-B XText Context IDE (0.0.2): Regenerated
- Event-B XText ContextUI IDE (0.0.2): Regenerated
- Event-B XText Machine (0.0.3):
  - Issue #3: Single-line comment after the element, multi-line comment before the element.
  - Issue #5: Event terminator using 'end' keyword instead of ';'
- Event-B XText Machine IDE (0.0.2) Regenerated
- Event-B XText Machine UI IDE (0.0.2) Regenerated

0.0.2

- Event-B XText Common (0.0.2):
  - Added transient value service for XContext and XMachine.
- Event-B XText Context (0.0.2):
  - Added formatter (used for auto-indentation).
- Event-B XText Machine (0.0.2):
  - Added formatter (used for auto-indentation).
- Event-B XText UI (0.0.1): Initial version
  - Added context menu for converting machines and contexts to XText.

0.0.1 Initial version contains the following plug-ins:

- Event-B XText Branding (0.0.1) Initial version: Branding information
- Event-B XText Common (0.0.1) Initial version: Common facilities
- Event-B XText Context (0.0.1) Initial version: Core support for Event-B contexts
- Event-B XText Context IDE (0.0.1) Initial version: IDE for Event-B contexts
- Event-B XText Context UI (0.0.1) Initial version: UI for Event-B contexts
- Event-B XText Machine (0.0.1) Initial version: Core support for Event-B machines
- Event-B XText Machine IDE (0.0.1) Initial version: IDE for Event-B machines
- Event-B XText Machine UI (0.0.1) Initial version: UI for Event-B machines

2.1.3 IMPORTANT

- Currently, Event-B XText front-end ONLY supports "standard" Event-B machines and contexts.
- Since the XContexts and XMachines are compiled to the Rodin files, the corresponding Rodin contexts and machines will be **OVER-WRITTEN**. Any changes in the Rodin files will not be lost.
- **DO NOT USE** the Event-B XText Front-end if you use modelling plug-ins such as *iUML-B* state-machines and class-diagrams, as the additional modelling elements will be over-written.

2.1.4 Known Issues

- Converting to XText: Currently, the "extended" attribute of events are not serialised.

2.1.5 Configuration

Event-B Explorer By default, XContext files (extension .bucx) and XMachine files (extension .bumx) are not display in the *Event-B Explorer*. To enable this, select *Customize view* for *Event-B Explorer* and uncheck the option *All files and folders*.

2.2 Basic Tutorial

This tutorial provides a step-by-step walk-through working with XEvent-B constructs. This tutorial also available as Cheatsheets with the Rodin Platform (`Help/Cheat Sheets/Event-B XText Cheatsheets/Event-B XText Basic Tutorial`).

2.2.1 Task 1. Customise the Event-B Explorer

Introduction The purpose of this task is to customise the Event-B Explorer so that XEvent-B constructs are visible.

Step 1. Disable the filter on "All files and folders" Select "Customize View" of Event-B Explorer View. Make sure that "All files and folders" from the dialog is **Unchecked**.

Conclusion Since the filter on "All files and folders" is now disabled, there might be other files and folders than XEvent-B constructs will also be visible in the Event-B Explorer.

2.2.2 Task 2. Create an Event-B Project

Introduction The purpose of this task is to create an Event-B project for the XEvent-B constructs.

Step 1. Create a new Event-B Project Create a new Event-B Project named "Club" using the *New Event-B Project* wizard (see Figure 4).

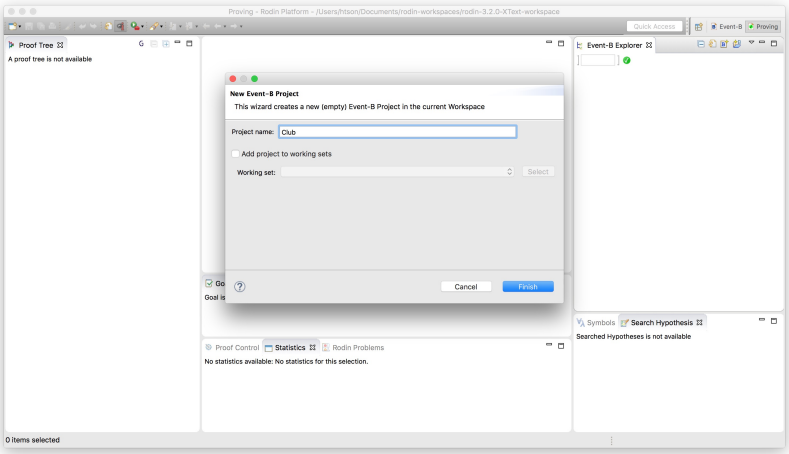

Figure 4: Create Event-B Project called "Club"

Conclusion By now, the project "Club" should be visible in the Event-B Explorer.

2.2.3 Task 3. Create a simple XContext coursesCtx.bucx

Introduction The purpose of this task is to create a simple XContext within the newly created project.

Step 1. Create a new XContext coursesCtx.bucx Create a new XContext named "coursesCtx.bucx" using the *New File wizard* (see Figure 5). **Important**: A pop-up dialog will be displayed asking to convert the "Club" project to XText project, please answer **Yes** (see Figure 6).

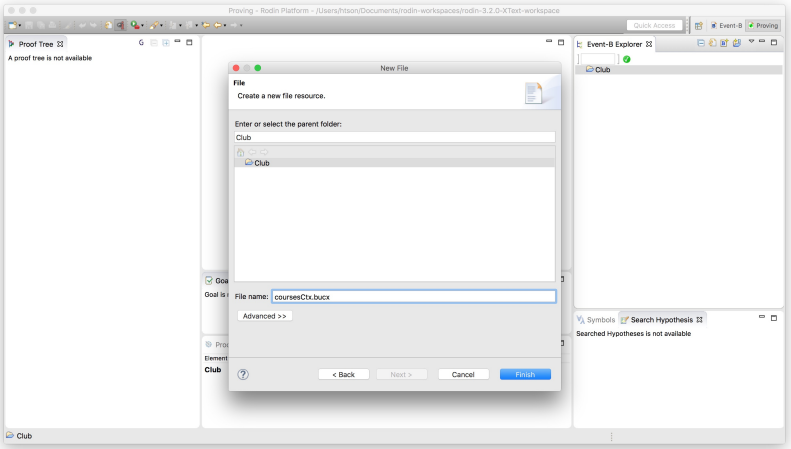

Figure 5: Create an XContext called “coursesCtx.bucx”

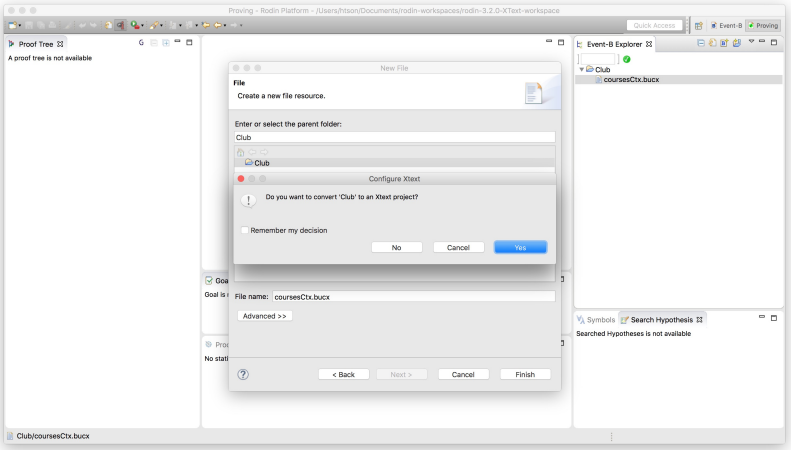

Figure 6: Convert “Club” to XText project

Step 2. Set the content of courseCtx.bucx Set the content of "coursesCtx.bucx" as follows.

context `coursesCtx`

sets `CRS`

constants `m`

axioms

`@axm0_1:` "finite(CRS)"

`@axm0_2:` "$m \in \mathbb{N}_1$"

`@thm0_1:` "$0 < m$" **theorem**

end

Step 3. Auto-format the code Automatically format the content of "coursesCtx.bucx" using short-cut (e.g., on Mac OS: `Cmd+Shift+F`).

Step 4. Save the file Save the file "coursesCtx.bucx".

Conclusion By now, the XContext "coursesCtx.bucx" and the corresponding Rodin Context "coursesCtx" should be visible in the Event-B Explorer.

Step 1. Create a new XMachine m0.bumx Create a new XMachine named "m0.bumx" using the *New File wizard*. The newly created file should be opened automatically in an XMachine editor.

Conclusion By now, the XMachine "m0.bumx" and the corresponding Rodin Machine "m0" (without any error) should be visible in the Event-B Explorer.

2.2.4 Task 4. Create a simple XMachine m0.bumx

Introduction The purpose of this task is to create a simple XMachine within the newly created project.

Step 1. Create a new XMachine m0.bumx Create a new XMachine named "m0.bumx" using the New File wizard. The newly created file should be opened automatically in an XMachine editor.

Step 2. Set the content of m0.bumx Set the content of "m0.bumx" as follows.

machine `m0`

variables `crs`

invariants

`@inv0_1:` "$crs \in \mathbb{P}(CRS)$"

`@thm0_2:` "finite(crs)" **theorem**

`@inv0_2:` "card(crs) $\leq m$"

`@DLF:` "(card(crs) $\neq m$) $\vee$ ($\exists\ cs \cdot cs \subseteq crs \wedge cs \neq \varnothing$)"

events

`INITIALISATION`

begin

`@act0_1:` "$crs := \varnothing$"

end

`OpenCourses`

when

`@grd0_1:` "card(crs) $\neq m$"

`@thm0_3:` "$crs \neq CRS$" **theorem**

then

`@act0_1:` "$crs :| crs \subset crs' \wedge$ card(crs') $\leq m$"

end

`CloseCourses` **anticipated**

any `cs` **where**

`@grd0_1:` "$cs \leq crs$"

`@grd0_2:` "$cs \neq \varnothing$"

then

`@act0_1:` "$crs := crs \setminus cs$"

end

end

Step 3. Auto-format the code Automatically format the content of "m0.bumx" by using short-cut (e.g., on Mac OS: Cmd+Shift+F).

Step 4. Save the file Save the file "m0.bumx".

Step 5. Add missing "sees" clause In the compiled Rodin Machine m0, there are several errors, due to the fact that **m0** refers to the sets and constants of the context courseCtx. **Add the missing "sees" clause** after the "machine" clause

```
sees courseCtx
```

(Note: One can use *Content Assist* after typing the "sees" keyword to select the context.

Step 6. Save the file again Save the file "m0.bumx" again.

Conclusion By now, the XMachine "m0.bumx" and the corresponding Rodin Machine "m0" (without any error) should be visible in the Event-B Explorer.

2.2.5 Task 5. Create extended XContexts

Introduction The purpose of this task is to create some more extended XContexts within the "Club" project.

Task 5.1. Create a simple XContext membersCtx.bucx Introduction The purpose of this sub-task is to create a simple XContext "membersCtx.bucx" within the "Club" project.

Step 1. Create a new XContext membersCtx.bucx Create a new XContext named "membersCtx.bucx" using the *New File* wizard.

Step 2. Set the content of courseCtx.bucx Set the content of "membersCtx.bucx" as follows.

```
context memebersCtx

sets MEM

axioms

    @axm0_1: "finite(MEM)"

end
```

Step 3. Auto-format the code Automatically format the content of "membersCtx.bucx" by using short-cut (e.g., on Mac OS: Cmd+Shift+F).

Step 4. Save the file ¡b¿Save the file **"membersCtx.bucx"**.

Conclusion By now, the XContext "membersCtx.bucx" and the corresponding Rodin Context "membersCtx" should be visible in the Event-B Explorer.

Task 5.2. Create an extended XContext participantsCtx.bucx Introduction The purpose of this sub-task is to create an extended XContext "participantsCtx.bucx" within the "Club" project.

Step 1. Create a new XContext participantsCtx.bucx Create a new XContext named "participantsCtx.bucx" using the *New File wizard*.

Step 2. Set the content of participantsCtx.bucx Set the content of "participantsCtx.bucx" as follows.

```
context participantsCtx

extends membersCtx

constants PRTCPT

axioms

    @axm1_2: "PRTCPT ∈ ℙ(MEM)"

    @thm1_1: "finite(PRTCPT)" theorem

end
```

Step 3. Auto-format the code Automatically format the content of "participantsCtx.bucx" by using short-cut (e.g., on Mac OS: Cmd+Shift+F).

Step 4. Save the file Save the file "participantsCtx.bucx".

Conclusion By now, the XContext "participantsCtx.bucx" and the corresponding Rodin Context "participantsCtx" should be visible in the Event-B Explorer.

Task 5.3. Create an extended XContext instructorsCtx.bucx Introduction The purpose of this sub-task is to create an extended XContext "instructorsCtx.bucx" within the "Club" project.

Step 1. Create a new XContext instructorsCtx.bucx Create a new XContext named "instructorsCtx.bucx" using the *New File wizard*.

Step 2. Set the content of instructorsCtx.bucx Set the content of "instructorsCtx.bucx" as follows.

```
context instructorsCtx

extends membersCtx coursesCtx

constants INSTR instrs

axioms

    @axm1_3: "INSTR ∈ ℙ(MEM)"

    @axm1_4: "instrs ∈ CRS → INSTR"

end
```

Step 3. Auto-format the code Automatically format the content of "intructorsCtx.bucx" by using short-cut (e.g., on Mac OS: Cmd+Shift+F).

Step 4. Save the file Save the file "instructorsCtx.bucx".

Conclusion By now, the XContext "instructorsCtx.bucx" and the corresponding Rodin Context "instructorsCtx" should be visible in the Event-B Explorer.

2.2.6 Task 6. Create refined XMachines

Introduction The purpose of this task is to create some more refined XMachines within the "Club" project.

Task 6.1. Create a refined XMachine m1.bumx Introduction The purpose of this sub-task is to create a refined XMachine "m1.bumx" within the "Club" project.

Step 1. Create a new XMachine m1.bumx Create a new XMachine named "m1.bumx" using the *New File wizard*. The newly created file should be opened automatically in an XMachine editor.

Step 2. Set the content of m1.bumx Set the content of "m1.bumx" as follows.

```
machine m1

refines m0

sees instructorsCtx participantsCtx

variables crs prtcpts

invariants
```

@inv1_1: "$prtcpts \in crs \leftrightarrow PRTCPT$"

@inv1_2: "$\forall c \cdot c \in crs \Rightarrow instrs(c) \notin prtcpts[\{c\}]$"

variant "$(crs \times PRTCPT) \setminus prtcpts$"

events

INITIALISATION **extended**

begin

@act1_2: "$prtcpts := \varnothing$"

end

OpenCourses **extended**

refines OpenCourses

when

@thm1_2: "$\mathrm{dom}(prtcpts) \subseteq crs$" **theorem**

end

CloseCourses **extended** **anticipated**

refines CloseCourses

begin

@act1_2: "$prtcpts := cs \lhd prtcpts$"

end

Register **convergent**

any p c **where**

@grd1_1: "$p \in PRTCPT$"

@grd1_2: "$c \in crs$"

@grd1_3: "$p \neq instrs(c)$"

@grd1_4: "$c \mapsto p \neq prtcpts$"

then

@act1_1: "$prtcpts := prtcpts \cup \{c \mapsto p\}$"

end

end

Step 3. Auto-format the code Automatically format the content of "m1.bumx" by using short-cut (e.g., on Mac OS: Cmd+Shift+F).

Step 4. Save the file Save the file "m1.bumx".

Conclusion By now, the XMachine "m1.bucx" and the corresponding Rodin Machine "m1" should be visible in the Event-B Explorer.

Task 6.2. Create a refined XMachine m2.bumx Introduction The purpose of this sub-task is to create a refined XMachine "m2.bumx" within the "Club" project.

Step 1. Create a new XMachine m2.bumx Create a new XMachine named "m2.bumx" using the *New File wizard*. The newly created file should be opened automatically in an XMachine editor.

Step 2. Set the content of m2.bumx Set the content of "m2.bumx" as follows.

machine m2

refines m1

sees instructorsCtx participantsCtx

variables atnds

invariants

@inv2_1: "$atnds \in CRS \pfun \mathbb{P}(PRTCPT)$"

@inv2_2: "$crs = \mathrm{dom}(atnds)$"

@inv2_3: "$\forall c \cdot c \in crs \Rightarrow prtcpts[\{c\}] = atnds(c)$"

@thm2_1: "$\mathrm{finite}(atnds)$" **theorem**

variant "$\mathrm{card}(atnds)$"

events

INITIALISATION

begin

@act2_1: "$atnds := \varnothing$"

end

OpenCourse

refines OpenCourses

any c **where**

@grd2_1: "$c \notin \mathrm{dom}(atnds)$"

@grd2_2: "$\mathrm{card}(atnds) \neq m$"

@thm2_2: "$\mathrm{card}(crs) \neq m$" **theorem**

with

@crs': "$crs' = crs \cup \{c\}$"

then

@act2_1: "$atnds(c) := \varnothing$"

end

CloseCourse **convergent**

refines CloseCourses

any c **where**

@grd2_1: "$c \in \mathrm{dom}(atnds)$"

with

@cs: "$cs = \{c\}$"

then

@act1_2: "$atnds := \{c\} \lhd atnds$"

end

Register **convergent**

refines Register

any p c **where**

@grd2_1: "$p \in PRTCPT$"

@grd2_2: "$p \neq instrs(c)$"

@grd2_3: "$c \in \mathrm{dom}(atnds)$"

@grd2_4: "$p \notin atnds(c)$"

@thm2_3: "$atnds(c) = prtcpts[\{c\}]$" theorem

then

@act2_1: "$atnds(c) := atnds(c) \ \{p\}$"

end

end

Conclusion By now, the XMachine "m2.bucx" and the corresponding Rodin Machine "m2" should be visible in the Event-B Explorer.

3 Concepts

4 Tasks

5 Reference

6 Samples

7 Legal

7.1 RODIN Software User Agreement

June 1st, 2006

7.1.1 Usage Of Content

THE RODIN PROJECT MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT

IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW, THEN YOU MAY NOT USE THE CONTENT.

7.1.2 Applicable Licences

Unless otherwise indicated, all Content made available by the CODA project is provided to you under the terms and conditions of one of the following licences.Unless otherwise indicated, all Content made available by the Rodin Project is provided to you under the terms and conditions of the Eclipse Public License Version 1.0 ("EPL"). A copy of the EPL is provided with this Content and is also available at `http://www.eclipse.org/legal/epl-v10.html`. For purposes of the EPL, "Program" will mean the Content.

Content includes, but is not limited to, source code, object code, documentation and other files maintained in the Rodin SourceForge CVS repository ("Repository") in CVS modules ("Modules") and made available as downloadable archives ("Downloads").

- Content may be structured and packaged into modules to facilitate delivering, extending, and upgrading the Content. Typical modules may include plug-ins ("Plug-ins"), plug-in fragments ("Fragments"), and features ("Features").
- Each Plug-in or Fragment may be packaged as a sub-directory or JAR (Java^{TM}ARchive) in a directory named "plugins".
- A Feature is a bundle of one or more Plug-ins and/or Fragments and associated material. Each Feature may be packaged as a sub-directory in a directory named "features". Within a Feature, files named "feature.xml" may contain a list of the names and version numbers of the Plug-ins and/or Fragments associated with that Feature.
- Features may also include other Features ("Included Features"). Within a Feature, files named "feature.xml" may contain a list of the names and version numbers of Included Features.

The terms and conditions governing Plug-ins and Fragments should be contained in files named "about.html" ("Abouts"). The terms and conditions governing Features and Included Features should be contained in files named "license.html" ("Feature Licenses"). Abouts and Feature Licenses may be located in any directory of a Download or Module including, but not limited to the following locations:

- The top-level (root) directory
- Plug-in and Fragment directories
- Inside Plug-ins and Fragments packaged as JARs

- Sub-directories of the directory named "src" of certain Plug-ins
- Feature directories

Note: if a Feature made available by the Rodin Project is installed using the Eclipse Update Manager, you must agree to a license ("Feature Update License") during the installation process. If the Feature contains Included Features, the Feature Update License should either provide you with the terms and conditions governing the Included Features or inform you where you can locate them. Feature Update Licenses may be found in the "license" property of files named "feature.properties" found within a Feature. Such Abouts, Feature Licenses, and Feature Update Licenses contain the terms and conditions (or references to such terms and conditions) that govern your use of the associated Content in that directory.

THE ABOUTS, FEATURE LICENSES, AND FEATURE UPDATE LICENSES MAY REFER TO THE EPL OR OTHER LICENSE AGREEMENTS, NOTICES OR TERMS AND CONDITIONS. SOME OF THESE OTHER LICENSE AGREEMENTS MAY INCLUDE (BUT ARE NOT LIMITED TO):

- Common Public License Version 1.0 (available at `http://www.eclipse.org/legal/cpl-v10.html`)
- Apache Software License 1.1 (available at `http://www.apache.org/licenses/LICENSE`)
- Apache Software License 2.0 (available at `http://www.apache.org/licenses/LICENSE-2.0`)
- IBM Public License 1.0 (available at `http://oss.software.ibm.com/developerworks/opensource/license10.html`)
- Metro Link Public License 1.00 (available at `http://www.opengroup.org/openmotif/supporters/metrolink/license.html`)
- Mozilla Public License Version 1.1 (available at `http://www.mozilla.org/MPL/MPL-1.1.html`)

IT IS YOUR OBLIGATION TO READ AND ACCEPT ALL SUCH TERMS AND CONDITIONS PRIOR TO USE OF THE CONTENT. If no About, Feature License, or Feature Update License is provided, please contact the Rodin Project to determine what terms and conditions govern that particular Content.

7.1.3 Cryptography

Content may contain encryption software. The country in which you are currently may have restrictions on the import, possession, and use, and/or re-export to another country, of encryption software. BEFORE using any encryption software, please check the country's laws, regulations and policies concerning the import, possession, or use, and re-export of encryption software, to see if this is permitted.

- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.