CamilleX User Manual

Thai Son Hoang and Dana Dghaym
ECS, University of Southampton
{T.S.Hoang, dd4g12} at ecs dot soton dot ac dot uk

Version 1.0.0
(for feature version 1.0.0)
Monday 12th November, 2018

1 CamilleX Front-end Overview

The CamilleX Front-end provides new CamilleX 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 CamilleX components can be invoked manually. The overall process can be seen in Figure 1.

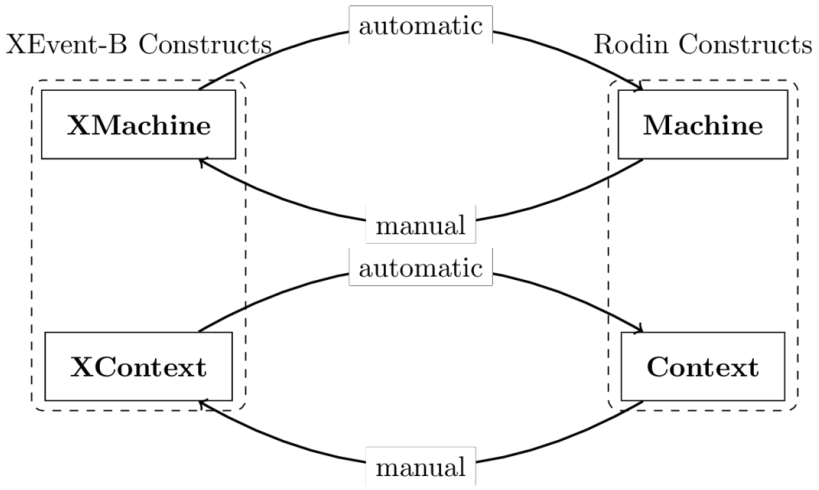

Figure 1: Overview of CamilleX and Rodin Event-B Constructs

2 Getting Started

2.1 Installation

2.1.1 Setup

- Before installing the CamilleX 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).

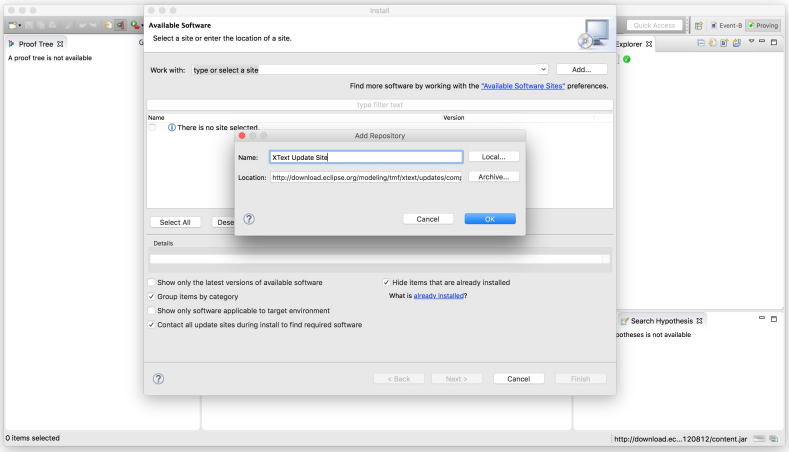

Figure 2: Adding XText Update Site

- The CamilleX feature is available from the main Rodin update site (under "Modelling Extensions" category). There are two versions of the feature, *CamilleX* providing facilities for working with CamilleX, and the *CamilleX (SDK)* is the feature including source code for software developers (see Figure 3).

2.1.2 IMPORTANT

- Currently, CamilleX **not only** supports "standard" Event-B machines and contexts, but also supports "*Machine Inclusion*" and "*Event Synchronisation*".

- 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 CamilleX if you use modelling plug-ins such as *iUML-B* state-machines and class-diagrams, as the additional modelling elements will be over-written.

- Windows users **must** change the workspace text file encoding to **UTF-8**. This can be updated under the Rodin Preferences: General/Workspace then in the "*Text file encoding*" section, select Other: UTF-8.

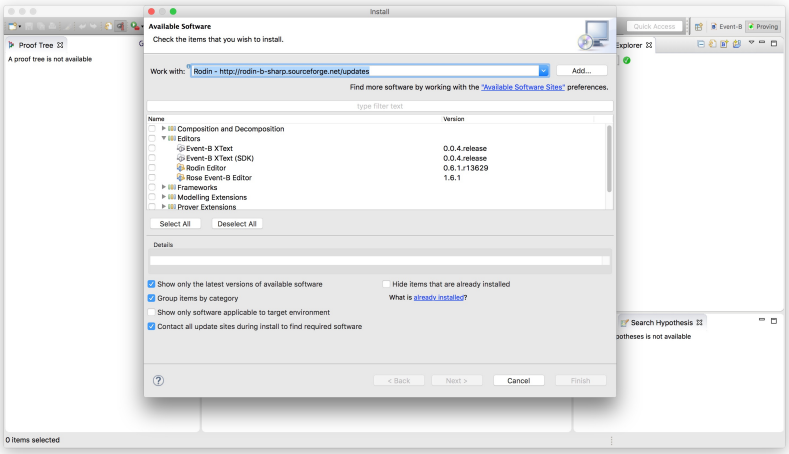

Figure 3: Adding XText Update Site

2.1.3 Known Issues

- Machine Inclusion:
  - Including the **same** machine to both the abstract and its refining machine can result in the repetition of invariants.

2.2 Basic Tutorial

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

2.2.1 Task 1. 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).

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

2.2.2 Task 2. 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).

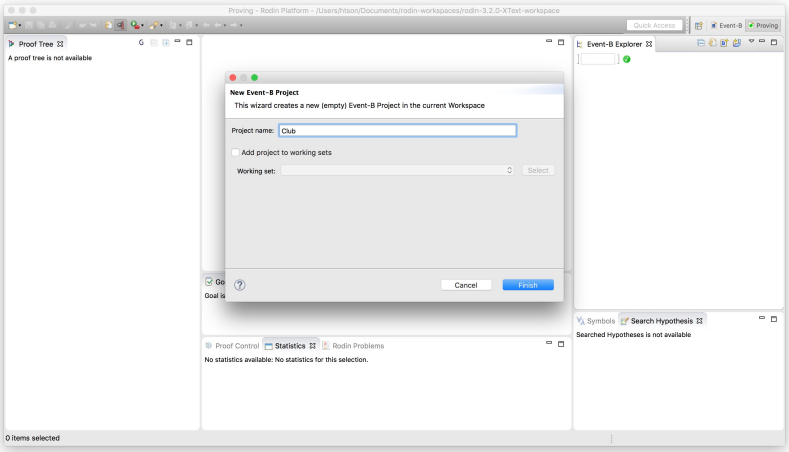

Figure 4: Create Event-B Project called “Club”

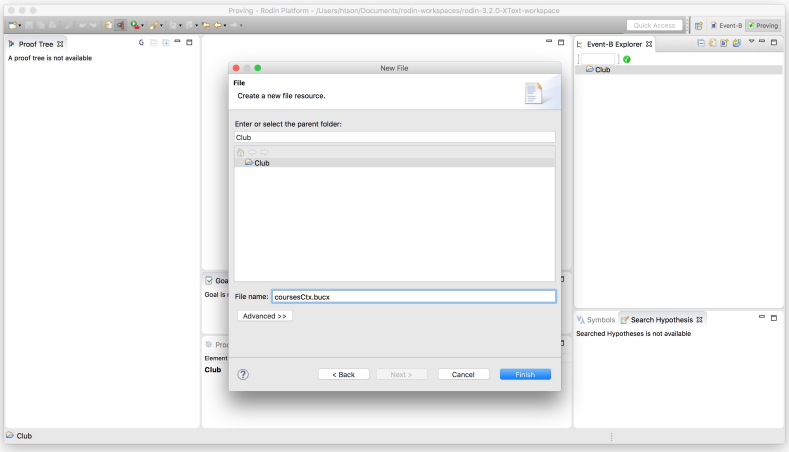

Figure 5: Create an XContext called “coursesCtx.bucx”

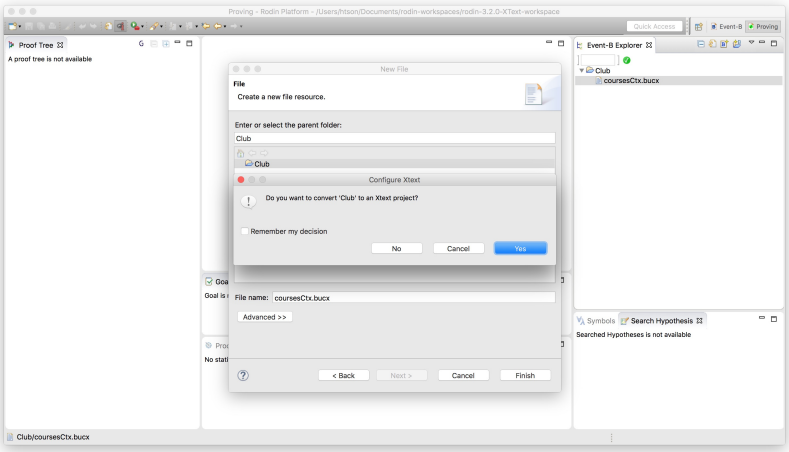

Figure 6: Convert "Club" to XText project

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:` $m \in \mathbb{N}_1$

theorem `@thm0_1:` $0 < m$

end

Important: In order to typeset Event-B mathematical symbol, e.g., $\mathbb{N}_1$, one can use content assist. For example, typing `NAT` and invoking content assist (e.g., on Mac OS `Ctrl+Space`), a dropdown list will appear with options for typesetting $\mathbb{N}$ and $\mathbb{N}_1$ (See Figure 7.

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

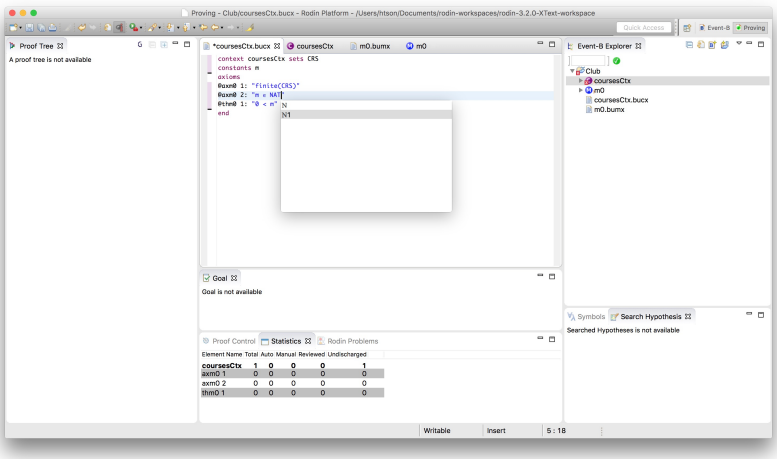

Figure 7: Type-setting $\mathbb{N}_1$ using Content Assist

Rodin Context "coursesCtx" should be visible in the Event-B Explorer (see Figure 8).

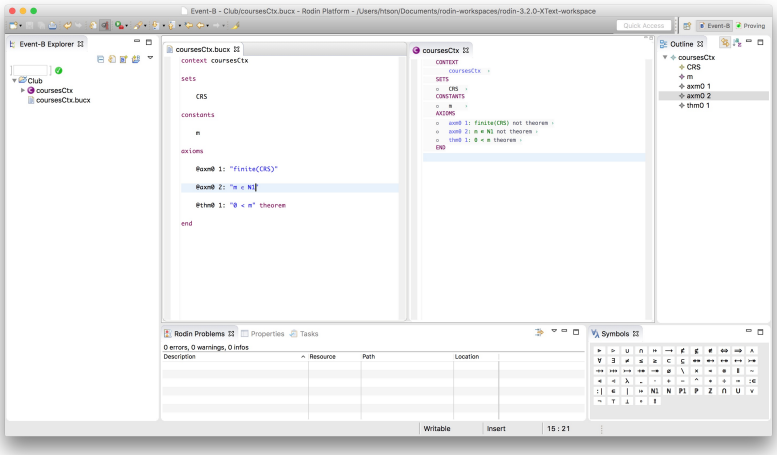

Figure 8: The final XContext coursesCtx.bucx

2.2.3 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 (see Figure 9. The newly created file should be opened automatically in an XMachine editor.

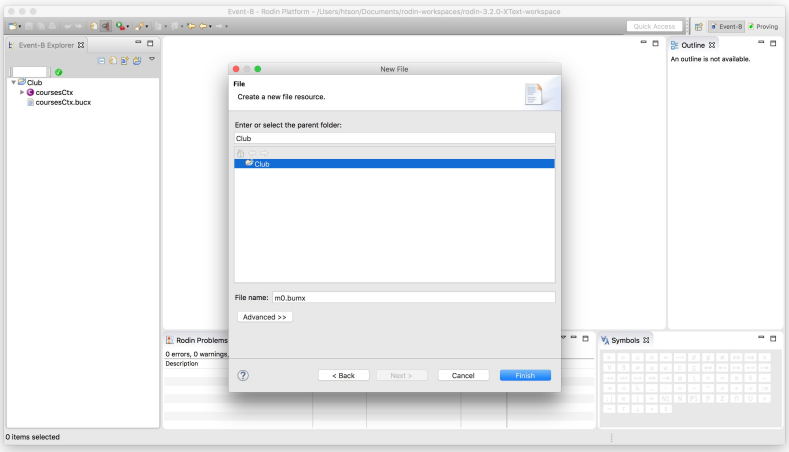

Figure 9: Type-setting $\mathbb{N}_1$ using Content Assist

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

machine m0

variables crs

invariants

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

theorem @thm0_2: $\text{finite}(crs)$

@inv0_2: $\text{card}(crs) \leq m$

@DLF: $(\text{card}(crs) \neq m) \lor (\exists\ cs \cdot cs \subseteq crs \land cs \neq \varnothing)$

events

event INITIALISATION

begin

@act0_1: $crs := \varnothing$

end

event OpenCourses

where

```
        @grd0_1: card(crs) ≠ m
        theorem @thm0_3: crs ≠ CRS
    then
        @act0_1: crs :| crs ⊂ crs' ∧ card(crs') ≤ m
    end
    anticipated event CloseCourses
    any cs where
        @grd0_1: cs ⊆ crs
        @grd0_2: cs ≠ ∅
    then
        @act0_1: crs := crs \ 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, see Figure 10).

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 (see Figure 11.

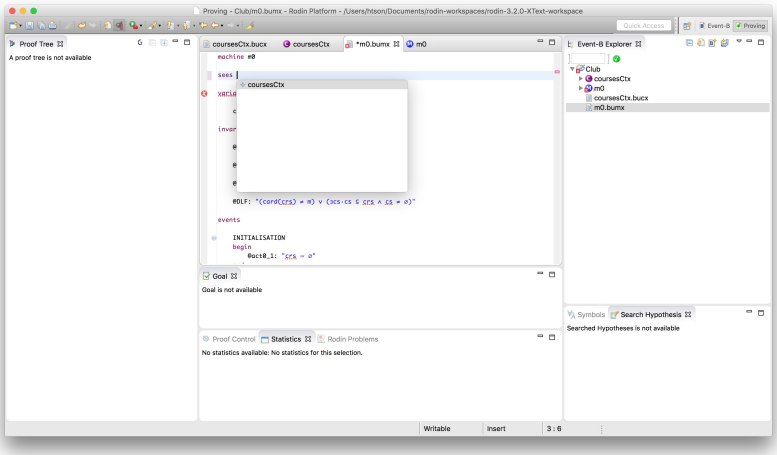

Figure 10: Content Assist for adding Sees clause

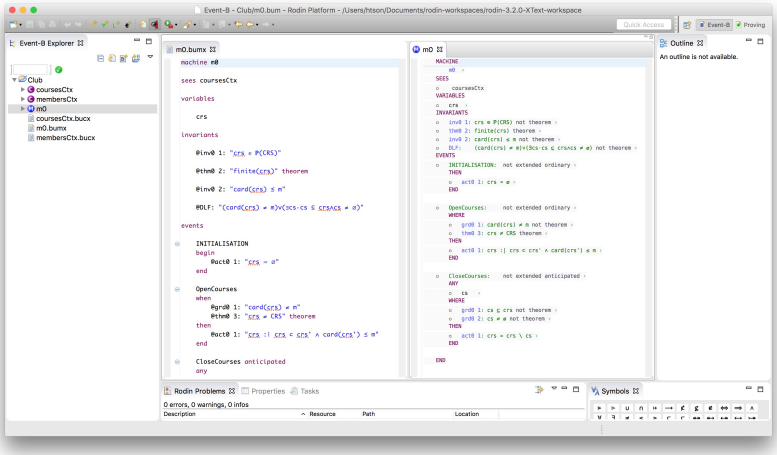

Figure 11: XMachine m0.bucx

2.2.4 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 (see Figure 12.

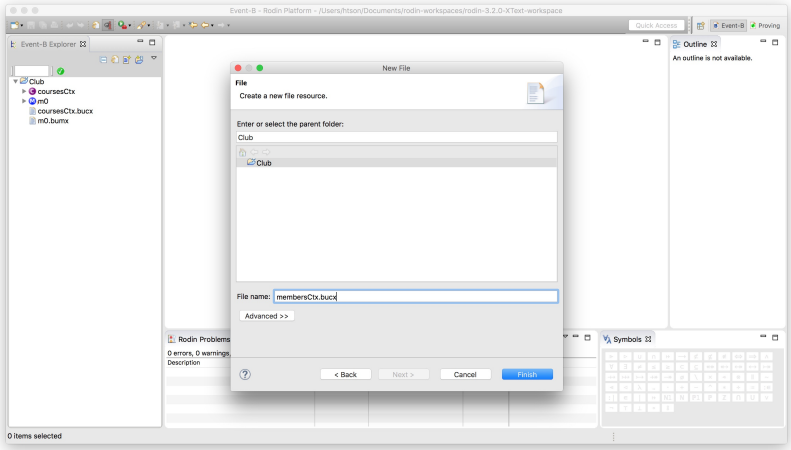

Figure 12: Create membersCtx.bucx

Step 2. Set the content of membersCtx.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 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 (see Figure 13.

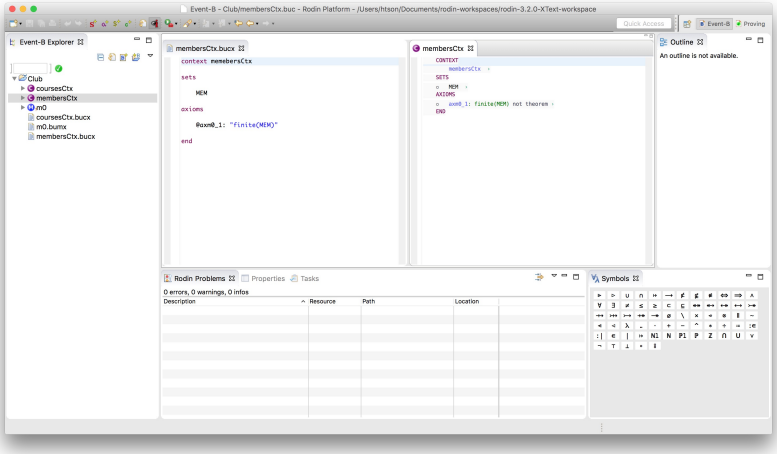

Figure 13: XContext membersCtx.bucx

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* (see Figure 14.

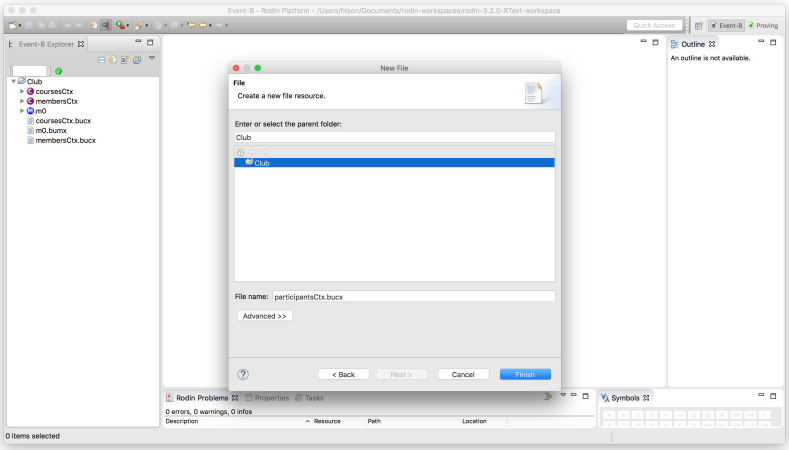

Figure 14: Create participantsCtx.bucx

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: $PRTCPT \in \mathbb{P}(MEM)$

theorem @thm1_1: finite($PRTCPT$)

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 (see Figure 15).

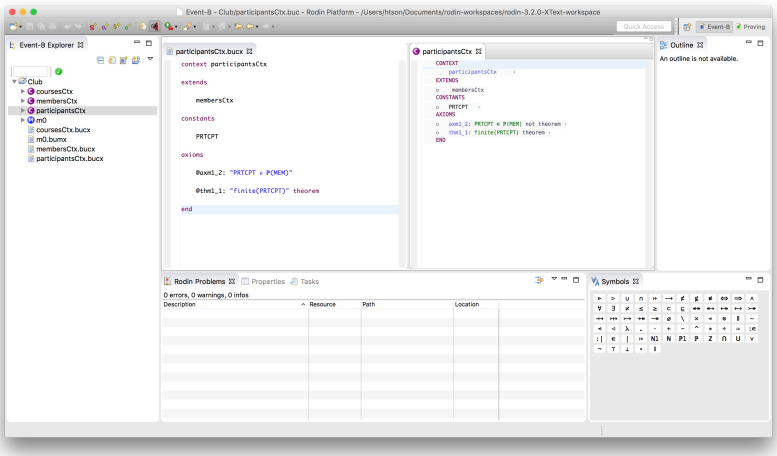

Figure 15: XContext participantsCtx.bucx

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* (see Figure 16.

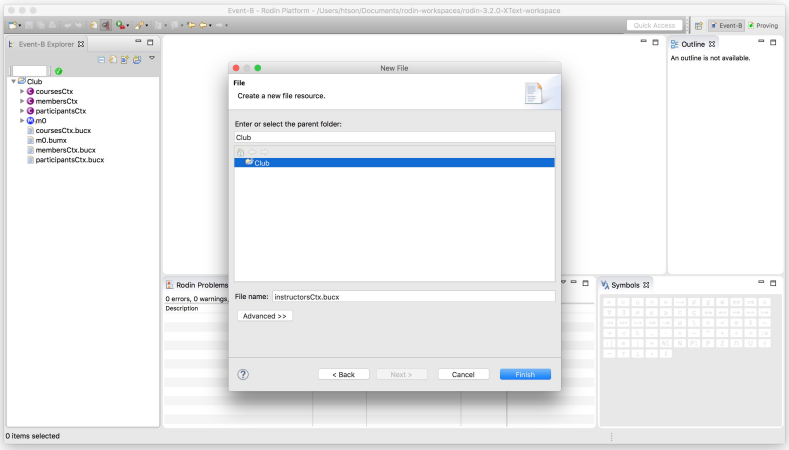

Figure 16: Create instructorsCtx.bucx

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:` $INSTR \in \mathbb{P}(MEM)$

`@axm1_4:` $instrs \in CRS \rightarrow INSTR$

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 (see Figure.

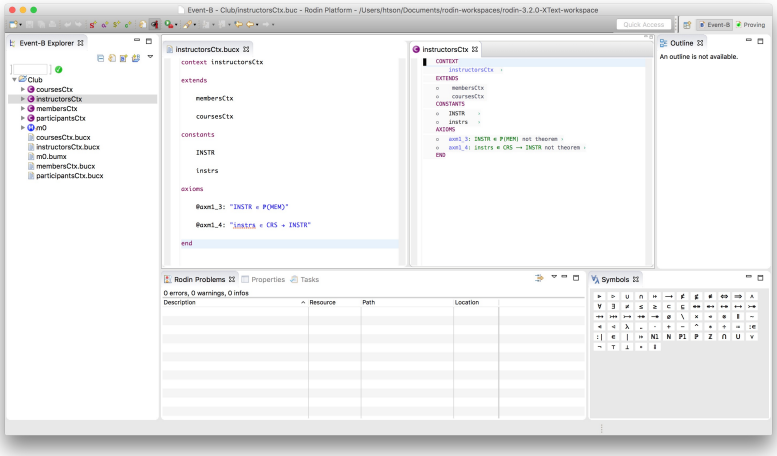

Figure 17: XContext instructorsCtx.bucx

2.2.5 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* (see Figure 18. The newly created file should be opened automatically in an XMachine editor.

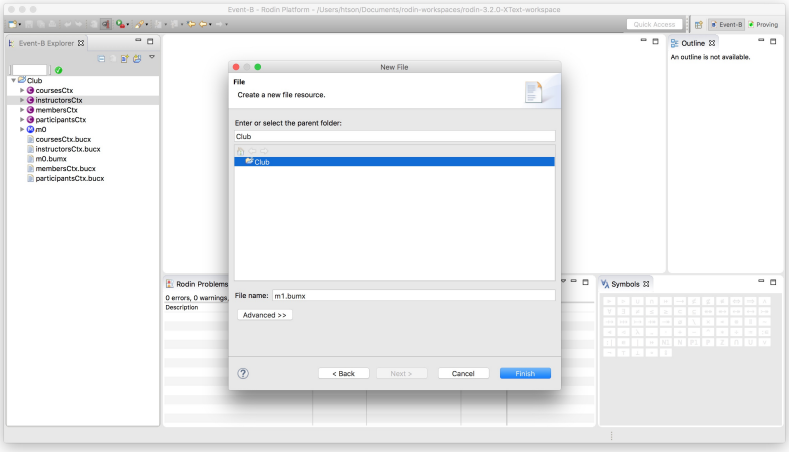

Figure 18: Create m1.bumx

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

variants @v0:$(crs \times PRTCPT) \setminus prtcpts$

events

event INITIALISATION extends INITIALISATION

then

@act1_2: $prtcpts := \varnothing$

end

event OpenCourses extends OpenCourses

where

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

end

anticipated event CloseCourses extends CloseCourses

then

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

end

convergent event Register

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 (see Figure 19).

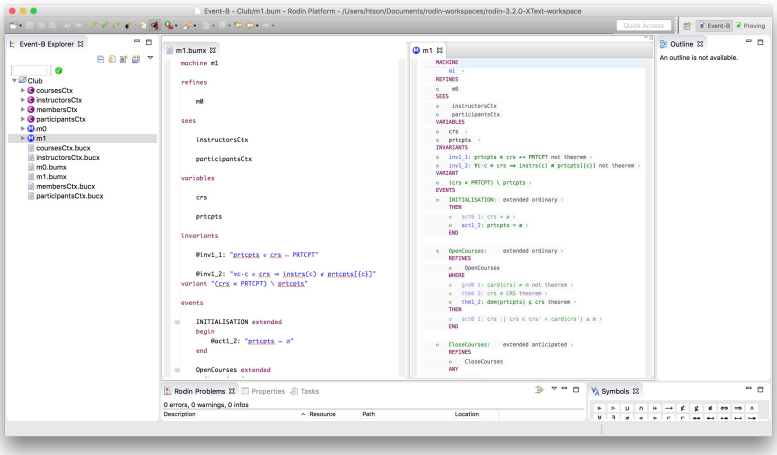

Figure 19: XMachine m1.bumx

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* (see Figure 20. The newly created file should be opened automatically in an XMachine editor.

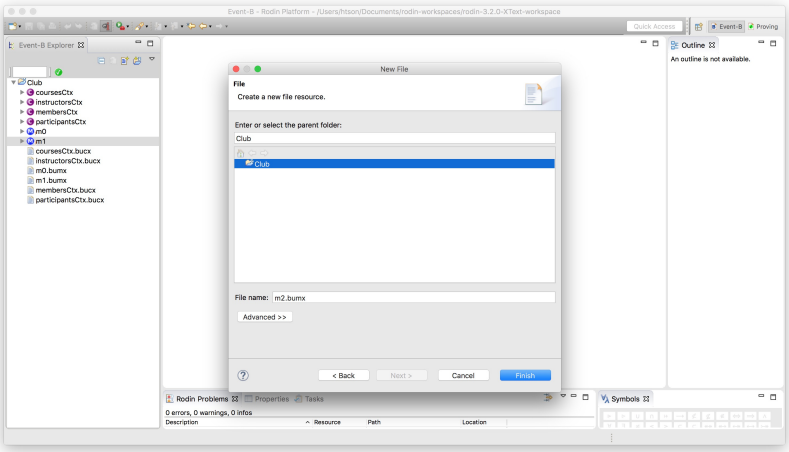

Figure 20: Create m2.bumx

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

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

variants @v0: $\mathrm{card}(atnds)$

events

event INITIALISATION

then

@act2_1: $atnds := \varnothing$

end

`event OpenCourse`

refines `OpenCourses`

any `c` **where**

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

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

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

then

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

with

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

end

convergent `event CloseCourse`

refines `CloseCourses`

any `c` **where**

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

then

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

with

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

end

convergent `event Register`

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: $p \notin atnds(c)$

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

then

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

end

end

Step 3. Auto-format the code Automatically format the content of "m2.bumx" by using short-cut (e.g., on Mac OS: Cmd+Shift+F).

Step 4. Save the file Save the file "m2.bumx".

Conclusion By now, the XMachine "m2.bucx" and the corresponding Rodin Machine "m2" should be visible in the Event-B Explorer (see Figure 21.

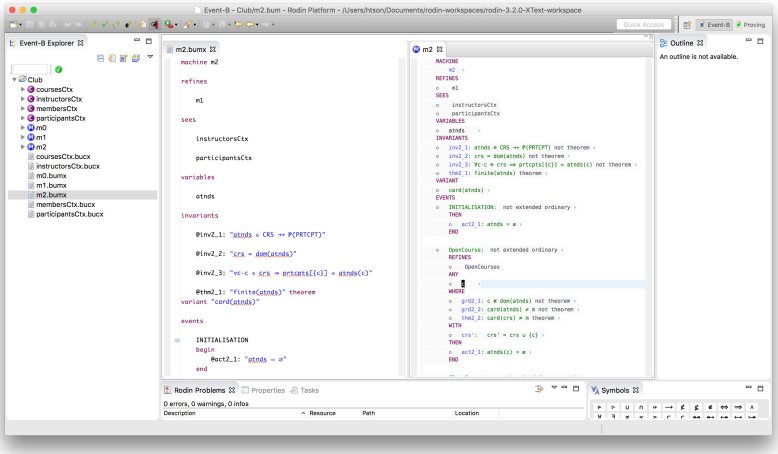

Figure 21: XMachine m2.bumx

2.3 Advanced Tutorial

This tutorial provides a step-by-step walk-through working with *machine inclusion* using XEvent-B. Following the same steps as in Section 2.2 to create machines and contexts, we can create a machine that can include other machines and can update the included machines variables via *event synchronisation*.

We illustrate the application of machine inclusion using XEvent-B by modelling a small example of "controlling cars on a bridge", which is based on Chapter 2 of "*Modeling in Event-B: System and Software Engineering*" book.

2.3.1 Task 1. Create the reusable model

Introduction The purpose of this task is to create the model that will be reused by other models using machine inclusion.

Step 1. Create a new Project (Sensor) with XMachine m0_SNSR.bumx Following the same steps as in Sections 2.2.1 and 2.2.3 for creating project and machines.

Step 2. Set the content of m0_SNSR.bumx Set the content of "m0_SNSR.bumx" as follows.

machine `m0_SNSR`

variables `SNSR`

invariants

theorem `@thm0_1:` $SNSR \in BOOL$

events

`event INITIALISATION`

then

`@act1:` $SNSR := FALSE$

end

`event SNSR_on`

where

`@grd1:` $SNSR = FALSE$

then

`@act1:` $SNSR := TRUE$

end

`event SNSR_off`

where

`@grd1:` $SNSR = TRUE$

then

`@act1:` $SNSR := FALSE$

end

end

Step 3. Auto-format and Save the file "m0_SNSR.bumx"

Conclusion By now, the XMachine "m0_SNSR.bumx" and the corresponding Rodin Machine "m0_SNSR" should be visible in the Event-B Explorer.

2.3.2 Task 2. Model the abstract level of cars on a bridge

Introduction The purpose of this task is to create the abstract model of the "cars on a bridge" example. At this level, we have not applied machine inclusion, but it is possible to apply machine inclusion right from the abstract level.

Step 1. Create the Context Car_c0_limit.bucx in a new project Car Following the same steps as in Section 2.2.2 for creating a simple context. **Set the content of "Car_c0_limit.bucx" as follows and save the file**.

context `Car_c0_limit`

constants `D`

axioms

`@axm1:` $D \in \mathbb{N}1$

end

Step 2. Create the Machine Car_m0_cars.bumx Set the content of "Car_m0_cars.bumx" as follows and save the file.

machine `Car_m0_cars`

sees `Car_c0_limit`

variables `A B C`

invariants

`@inv0_1:` $A \in \mathbb{N}$

`@inv0_2:` $B \in \mathbb{N}$

@inv0_3: $C \in \mathbb{N}$

@inv0_4: $A = 0 \lor C = 0$

@inv0_5: $A + B + C \leq D$

theorem @thm0_1: $B \leq D$

events

event INITIALISATION

then

@act1: $A := 0$

@act2: $B := 0$

@act3: $C := 0$

end

event ML_out

where

@grd1: $C = 0$

@grd2: $A + B \neq D$

then

@act1: $A := A + 1$

end

event ML_in

where

@grd1: $C \neq 0$

then

@act1: $C := C - 1$

end

event IL_in

where

@grd1: $A \neq 0$

then

@act1: $A := A - 1$

@act2: $B := B + 1$

end

event IL_out

where

@grd1: $B \neq 0$

@grd2: $A = 0$

then

@act1: $B := B - 1$

@act2: $C := C + 1$

end

end

Conclusion Saving the XContext and XMachine files will generate the corresponding Rodin files. In the "Car" you have the context "Car_c0_limit" and the machine "Car_m0_cars". Ideally the reusable models should be in a different project, that is why we added the reusable model in a different project "Sensor".

2.3.3 Task 3. Model an XMachine using machine inclusion

Introduction In this task we define the XMachine "Car_m1_SNSR.bumx" which is a refinement of the machine "Car_m0_cars" and includes two instances of "m0_SNSR". The keywords in red are **not** part of the standard Event-B syntax, they correspond to machine inclusion and event synchronisation.

Step 1. Create the file "Car_m1_SNSR.bumx" Set its contents as follows.

machine Car_m1_SNSR

includes Sensor.m0_SNSR as IL_out ML_out

refines Car_m0_cars

sees Car_c0_limit

variables A B C

invariants

@inv1_1: $IL\_out\_SNSR = TRUE \Rightarrow B \neq 0$

events

event INITIALISATION extends INITIALISATION

synchronises IL_out.INITIALISATION

synchronises ML_out.INITIALISATION

end

event ML_out extends ML_out

synchronises ML_out.SNSR_off

end

event ML_in extends ML_in

end

event IL_in extends IL_in

where

@inv0_2copy: "$B \in \mathbb{N}$" **theorem**

end

IL_out **refines** IL_out

synchronises IL_out.SNSR_off

where

@grd2: $A = 0$

then

@act1: $B := B - 1$

```
        @act2: C := C + 1
    end
    event ML_out_ARR
    synchronises ML_out.SNSR_on
    end
    event IL_out_ARR
    synchronises IL_out.SNSR_on
    where
        @grd2: B ≠ 0
    end
end
```

Step 2. Auto-format the file "Car_m1_SNSR.bumx" and Save it.

Conclusion After saving the file a standard Event-B machine "Car_m1_SNSR" will be generated. The generated machine (Figure 22) is flattened to include the variables and invariants of the included machine "m0_SNSR" which are renamed according to the chosen prefixes. In addition to the guards and actions of the synchronised events. The project name must be specified when including a machine (e.g., Sensor.m0_SNSR), and the project (Sensor) of the included machines must be opened in the same workspace. You can also use content assist to see all available machines in the workspace.

When synchronising an event you can add the prefix of the required machine followed by the synchronised event name (e.g., IL_out.SNSR_on where "IL_out" is one of the included machine prefixes and "SNSR_on" is the synchronised event). It is also possible to include more than one machine and synchronise with more than one event. Notice the order of the generated elements in the flattened machine is the included elements from last to first then the source machine elements.

3 Concepts

3.1 XText Projects

Each project containing CamilleX constructs must be set to be XText project. An XText project has an associated XContext and XMachine builders that can compile CamilleX source files into Rodin files as they are changed. The builders can be turn off via the preferences, either workspace-wise or project-wise (see Figure 23.

Car_m1_SNSR

```
MACHINE
  Car_m1_SNSR
REFINES
  Car_m0_cars
SEES
  Car_c0_limit
VARIABLES
  ML_out_SNSR
  IL_out_SNSR
  A
  B
  C
INVARIANTS
  ML_out_thm0_1   :   ML_out_SNSR ∈ BOOL
  IL_out_thm0_1   :   IL_out_SNSR ∈ BOOL
  inv1_1   :   IL_out_SNSR = TRUE ⟹ B ≠ 0
EVENTS
  INITIALISATION   ≙
   extended
  STATUS
   ordinary
  BEGIN
   act1   :   A ≔ 0
   act2   :   B ≔ 0
   act3   :   C ≔ 0
   ML_out_act1   :   ML_out_SNSR ≔ FALSE
   IL_out_act1   :   IL_out_SNSR ≔ FALSE
  END

  ML_out   ≙
   extended
  STATUS
```

Figure 22: Flattened Machine “Car_m1_SNSR”

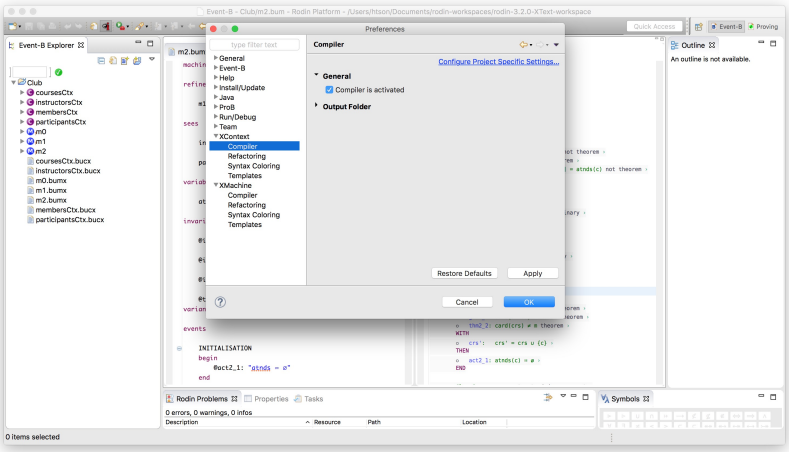

Figure 23: XContext Preference

The XText projects must be organised such that all CamilleX constructs has the project as the source container.

3.2 CamilleX Builders

The CamilleX Builders, i.e., XContext builder and XMachine builder, build CamilleX constructs, i.e., XContext and XMachine using their own compiler. If they are enabled, the CamilleX builders are run everytime an individual CamilleX file is saved. Problems detected by the CamilleX builders are classified as either warnings or errors. Compile-time errors are always reported as errors by the CamilleX buiders and in the presence of errors, no new Rodin files are created or updated, i.e., the CamilleX builders do not produce any new Rodin file content. In the case of machine inclusion and event synchronisation, a flattened machine is generated which includes data from the included machine and the synchronised events, which can be renamed if prefixing is applied.

3.3 Content Assist

Content assist are available for typesetting keywords and Event-B mathematical symbols. The short-cut for invoking content assist is `Ctrl+Space`. Figure 24 shows an example for content assist with keywords. For Event-B mathematical symbols, the key combination is defined by the Rodin Keyboard plug-in.

4 Tasks

4.1 Creating CamilleX Files

New CamilleX files can be create via the *New File wizard* with appropriate file extensions. The file extension for XContext is `bucx` and for XMachine is `bumx`. The syntax of XContext and XMachine can be seen in Section 5.2 and Section 5.3, respectively.

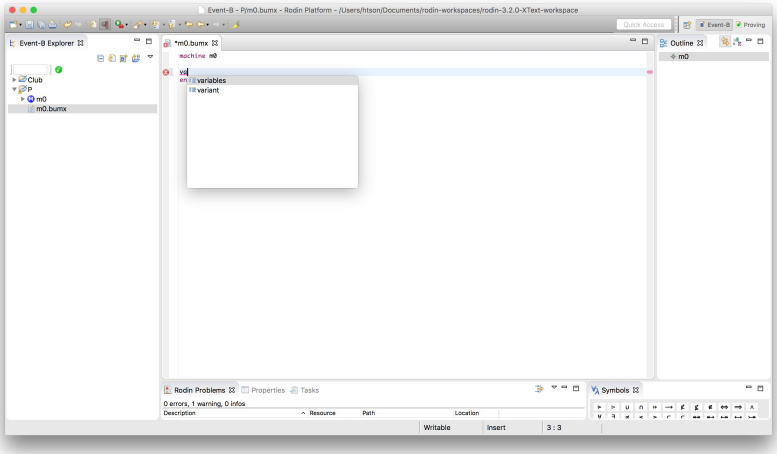

Figure 24: Keyword Content Assist

4.2 Typesetting Event-B Mathematical Symbols

Event-B mathematical symbols in predicates, expressions, and assignments are typeset using Content Assist. The definition of the character combinations is defined in the Rodin Keyboard plug-in.

5 Reference

5.1 Common Syntax

```
ML_COMMENT ::= /* STRING */

SL_COMMENT ::= // SL_STRING

ID ::= [^] (LETTER | _) {LETTER | DIGIT | _ }

XLABEL ::= @STRING:
```

5.2 XContext Syntax

```
XCONTEXT ::=

            context ID

            [extends ID { ID }]

            [sets XSET {XSET}]

            [constants XCONSTANT { XCONSTANT }]
```

```
            [axioms XAXIOM {XAXIOM}]

            end

    XSET ::= ID

    XCONSTANT ::= ID

    XAXIOM ::= [theorem ] XLABEL "XPREDICATE"
```

5.3 XMachine syntax

```
XMACHINE ::=

            machine ID

            [XINCLUDE { XINCLUDE }]

            [refines ID]

            [sees ID { ID }]

            [variables XVARIABLE {XVARIABLE}]

            [invariants XINVARIANT { XINVARIANT }]

            [variants XVARIANT {XVARIANT}]

            [events XEVENT { XEVENT }]

            end

    XINCLUDE ::=

            includes ID.ID

            [as ID {ID}]

    XVARIABLE ::= ID

    XINVARIANT ::= [theorem ] XLABEL "XPREDICATE"

    XVARIANT ::= XLABEL "XEXPRESSION"

    XEVENT ::=
            [ordinary | convergent | anticipated ] event ID
```

```
        ([extended ID | refines ID { ID }])
            [any XPARAMETER { XPARAMETER }]
        [XSYNC { XSYNC }]
            [where XGUARD { XGUARD }]
            [then XACTION { XACTION }]
            [with XWITNESS { XWITNESS }]
        end
XSYNC ::=
        synchronises [ID.]ID
XPARAMETER ::= ID
XGUARD ::=  [theorem ] XLABEL "XPREDICATE"
XACTION ::= XLABEL "XASSIGNMENT"
XWITNESS ::= XLABEL "XPREDICATE"
```

5.4 Preferences

5.4.1 XContext Preferences

The following XContext preferences can be set on the the XContext preference page and its sub-pages.

Option	Description	Default
Compiler is activated	**Compiler is activated** or **deactivated**	Activated

Table 1: XContext Compiler Preferences

Compiler

Syntax Coloring

5.4.2 XMachine Preferences

The following XMachine preferences can be set on the XMachine preference page and its sub-pages.

Option	Description	Default
Comment	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Dark Green White None Platform dependent
Default	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Black White None Platform dependent
Invalid Symbol	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Black White None Platform dependent
Keyword	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Dark Purple White Bold Platform dependent
Number	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Dark Gray White None Platform dependent
Punctuation character	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Black White None Platform dependent
String	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Blue White None Platform dependent
Task Tag	**Color** **Background** **Style** (Italic, Bold, Underline, Strike through) **Font**	Light Blue White Bold Platform dependent

Table 2: XContext Syntax Coloring Preferences

Option	Description	Default
Compiler is activated	**Compiler is activated** or **deactivated**	Activated

Table 3: XMachine Compiler Preferences

Compiler

Syntax Coloring

5.5 XEvent-B Editors

5.5.1 XEvent-B Content Assist

In the XContext and XMachine editors press `Ctrl+Space` on code to complete. This opens a list of available code completions. Some tips for using code assist are listed in the following paragraph:

- You can use the mouse or the keyboard (Up Arrow, Down Arrow, Page Up, Page Down, Home, End, Enter) to navigate and select lines in the list.
- Clicking or pressing Enter on a selected line in the list inserts the selection into the editor.

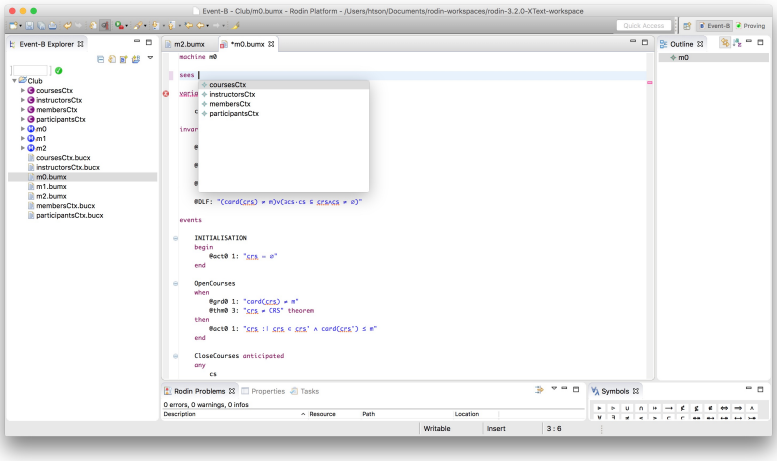

Figure 25: Content Assist for adding Sees clause

5.5.2 XEvent-B Formatter

In the XContent and XMachine editors press `Ctrl+Shift+F` on code to format it. If no selection is set then the entire source is formatted otherwise only the selection will be.

Option	Description	Default
Comment	**Color**	Dark Green
	Background	White
	Style (Italic, Bold, Underline, Strike through)	None
	Font	Platform dependent
Default	**Color**	Black
	Background	White
	Style (Italic, Bold, Underline, Strike through)	None
	Font	Platform dependent
Invalid Symbol	**Color**	Black
	Background	White
	Style (Italic, Bold, Underline, Strike through)	None
	Font	Platform dependent
Keyword	**Color**	Dark Purple
	Background	White
	Style (Italic, Bold, Underline, Strike through)	Bold
	Font	Platform dependent
Number	**Color**	Dark Gray
	Background	White
	Style (Italic, Bold, Underline, Strike through)	None
	Font	Platform dependent
Punctuation character	**Color**	Black
	Background	White
	Style (Italic, Bold, Underline, Strike through)	None
	Font	Platform dependent
String	**Color**	Blue
	Background	White
	Style (Italic, Bold, Underline, Strike through)	None
	Font	Platform dependent
Task Tag	**Color**	Light Blue
	Background	White
	Style (Italic, Bold, Underline, Strike through)	Bold
	Font	Platform dependent

Table 4: XMachine Syntax Coloring Preferences

6 Legal

6.1 RODIN Software User Agreement

June 1st, 2006

6.1.1 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.

6.1.2 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.

6.1.3 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.

A Release Notes

1.0.0 The feature is now called CamilleX (instead of XEvent-B)

- Branding (0.0.4): Updated logo to CamilleX.
- Common (0.0.5): Update copyright statements in source code.
- Cheatsheets (1.0.0): Updated the name to use CamilleX instead of XEvent-B.
- Documentation (1.0.0): Updated the name to use CamilleX instead of XEvent-B.
- UI (0.1.0): Updated the name to use CamilleX instead of XEvent-B.
- XContext (1.0.0): Updated the name to use CamilleX instead of XEvent-B.
  - Update dependency ranges
- XContext IDE (1.0.0): Updated the name to use CamilleX instead of XEvent-B.
  - Update dependency ranges
- XContext UI (1.0.0): Updated the name to use CamilleX instead of XEvent-B.
  - Update dependency ranges
- XMachine (1.0.0): Updated the name to use CamilleX instead of XEvent-B.
  - Update dependency ranges
- XMachine IDE (1.0.0): Updated the name to use CamilleX instead of XEvent-B.

  - Update dependency ranges

- XMachine UI (1.0.0): Updated the name to use CamilleX instead of XEvent-B.
  - Update dependency ranges

0.0.7

- XEvent-B Branding (0.0.3): Updated logo to XEvent-B.
- XEvent-B Common (0.0.4): Enhancement (Issue #11).
  - Machines from different projects can now be included.
  - Machines are now included using qualified name defined as: *projectName.machineName*
- XEvent-B Documentations (0.0.7): Update documentation for 0.0.7 release.
- XEvent-B XContext (0.0.5): Changed dependency on XText to [2.12.0, 3.0.0).
- XEvent-B XContext IDE (0.0.4): Changed dependency on XText to [2.12.0, 3.0.0).
- XEvent-B XContext UI (0.0.4): Changed dependency on XText to [2.12.0, 3.0.0).
- XEvent-B XMachine (0.0.5):
  - Changed dependency on XText to [2.12.0, 3.0.0).
  - Fixed Issue #8: Comments are not parsed.
  - Fixed Issue #10: Variants not translated: Fix is part of inclusion plug-in release 0.2.0.
  - Flattened machines now have the included machine elements generated before the source machine.
  - Order of generating elements of multiple inclusions and/or instances is from last to first.
  - This update is part of inclusion plug-in release 0.2.0.
- XEvent-B XMachine IDE (0.0.4): Changed dependency on XText to [2.12.0, 3.0.0).
- XEvent-B XMachine UI (0.0.4):
  - Changed dependency on XText to [2.12.0, 3.0.0).
  - Regenerated from XEvent-B XMachine 0.0.5

0.0.6

- Renamed plug-ins and features to XEvent-B (instead of Event-B XText).
- XEvent-B Branding (0.0.2): Renamed from Event-B XText Branding.
- XEvent-B Documentations (0.0.2): Renamed from Event-B XText Documentations.
- XEvent-B Cheatsheets (0.0.2): Renamed from Event-B XText Cheatsheets.
- XEvent-B Common (0.0.3): Renamed from Event-B XText Common.
- XEvent-B UI (0.0.2): Renamed from Event-B XText UI.
- XEvent-B XContext (0.0.4): Renamed from Event-B XText Context.
- XEvent-B XContext IDE (0.0.3): Renamed from Event-B XText Context IDE.
- XEvent-B XContext UI (0.0.3): Renamed from Event-B Context UI.
- XEvent-B XMachine (0.0.4): Renamed from Event-B XText Machine.
  - Support Machine Inclusion and Event Synchronisation.
- XEvent-B XMachine IDE (0.0.3): Renamed from Event-B XText Machine IDE.
- XEvent-B XMachine UI (0.0.3): Renamed from Event-B XText Machine UI.

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
- 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