

CamilleX Documentation

None

Thai Son Hoang

None

Table of contents

| | |
|-------------------------|---|
| 1. CamilleX User Manual | 3 |
| 2. Getting Started | 4 |
| 2.1 Installation | 4 |
| 2.2 Configuration | 4 |
| 2.3 IMPORTANT | 4 |
| 2.4 Basic Tutorial | 5 |

1. CamilleX User Manual

CamilleX new constructs (called `XMachines` and `XContexts`) for `Event-B` modelling. The new constructs are text files which are automatically translated into the corresponding `Rodin`'s `Event-B` constructs (i.e., `Machines` and `Contexts`) accordingly. Facility for translating to and from `Rodin`'s components to `CamilleX` components can be invoked manually. `CamilleX` is inspired by `Camille` text editor for `Rodin` and is based on `XText` technology, hence the name `CamilleX`.

- *Getting Started:*
- *Installation:* Information for installing the *CamilleX* feature.
- *Basic tutorial:* This tutorial provides a step-by-step walk-through working with `CamilleX` constructs.

2. Getting Started

2.1 Installation

[CamilleX](#) is available from the main Rodin update site (under [CamilleX](#) category). There are two versions of the feature, the standard version for users and the SDK version for software developers which include source code.

2.2 Configuration

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](#).

2.3 IMPORTANT

Currently, *CamilleX* not only supports *standard* Event-B machines and contexts, but also supports *Machine Inclusion* (for composition), and *Record* extension to the Event-B modelling language.

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 that use the Rodin files as source such as *UML-B* state-machines and class-diagrams, as the additional modelling elements will be over-written.

2.4 Basic Tutorial

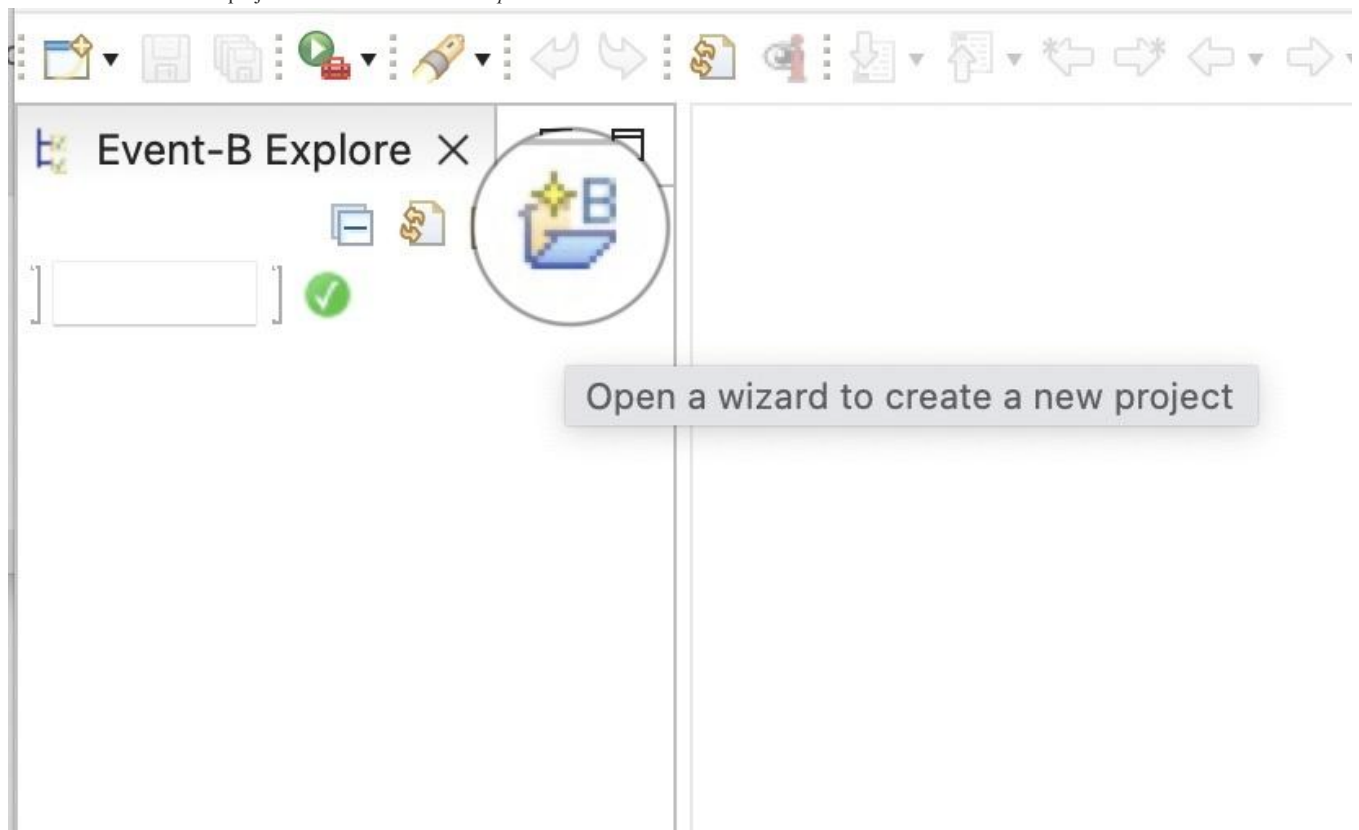
2.4.1 Task 1. Create an Event-B Project

Introduction

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

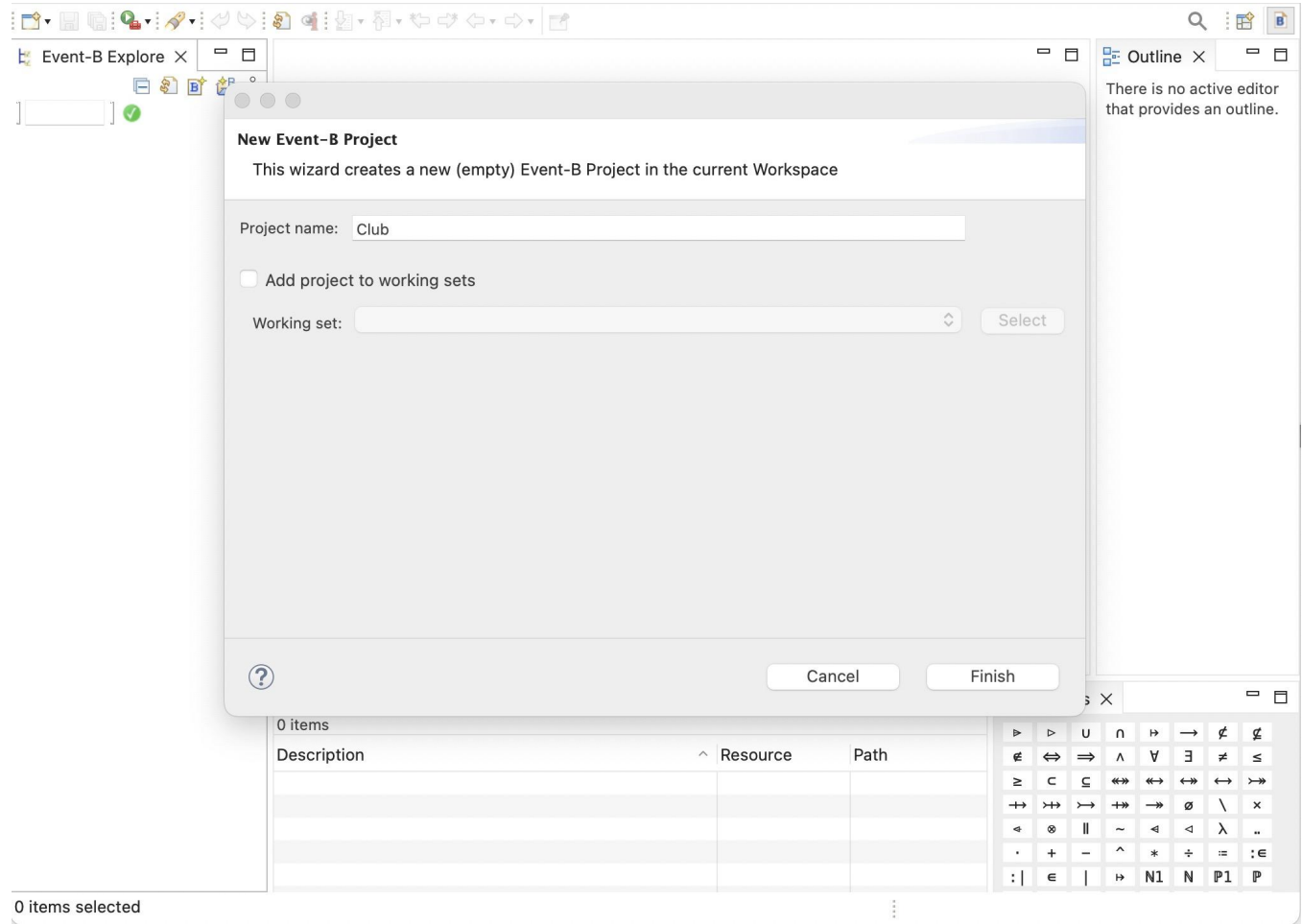
Step 1. Create a New Event-B Project Named `Club`

- Click on the new Event-B project button on the *Event-B Explorer*.



(The same wizard can be invoke through the menu `File -> New -> Event-B Project`.)

- From the pop-up dialog, enter `Club` as the `Project name`



- Click **Finish** to confirm the creation of the project.

New Event-B Project

This wizard creates a new (empty) Event-B Project in the current Workspace

Project name:

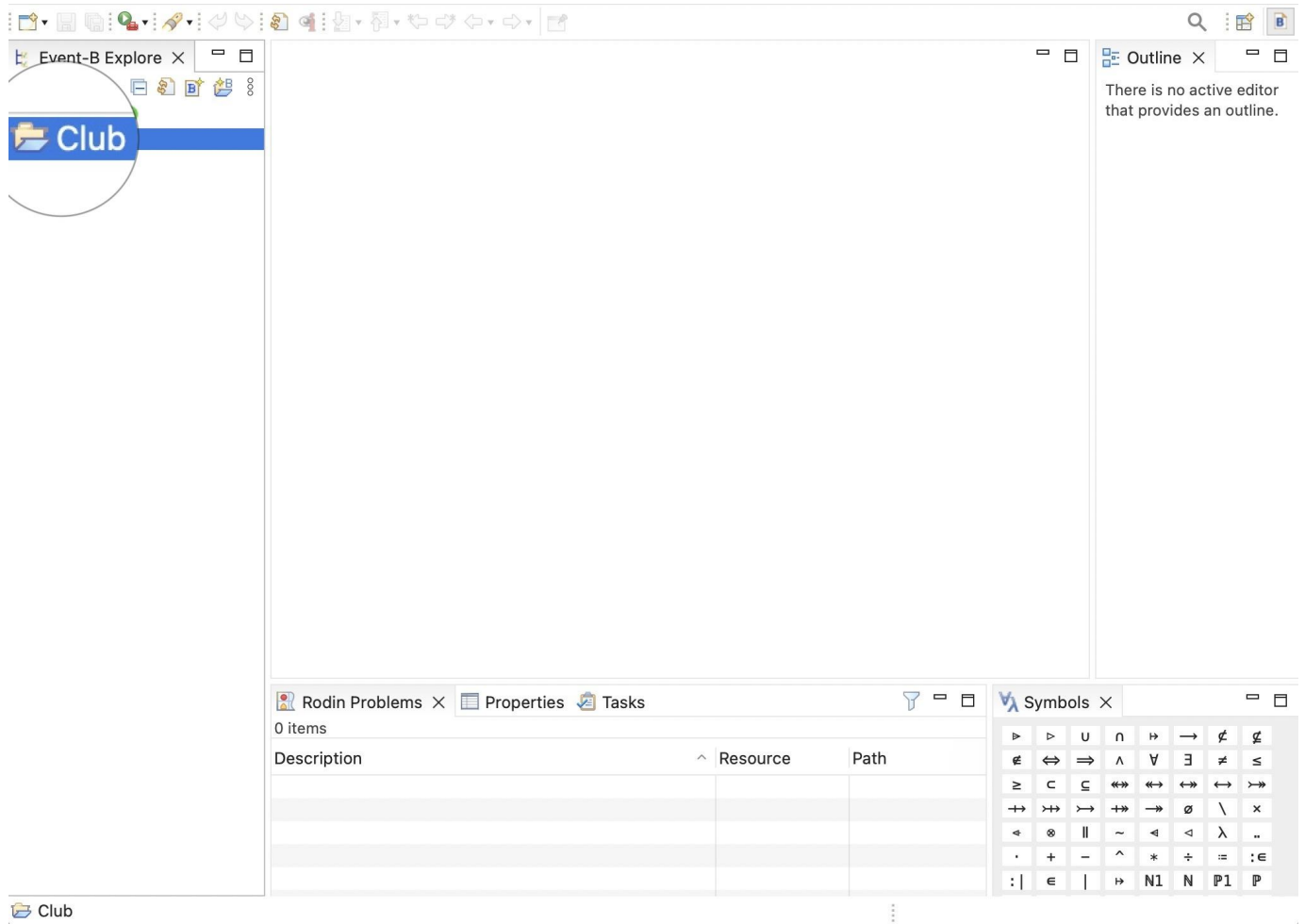
☐ Add project to working sets

Working set: Select

? Cancel Finish

Conclusion

By now, the project `Club` should be visible in the *Event-B Explorer*.



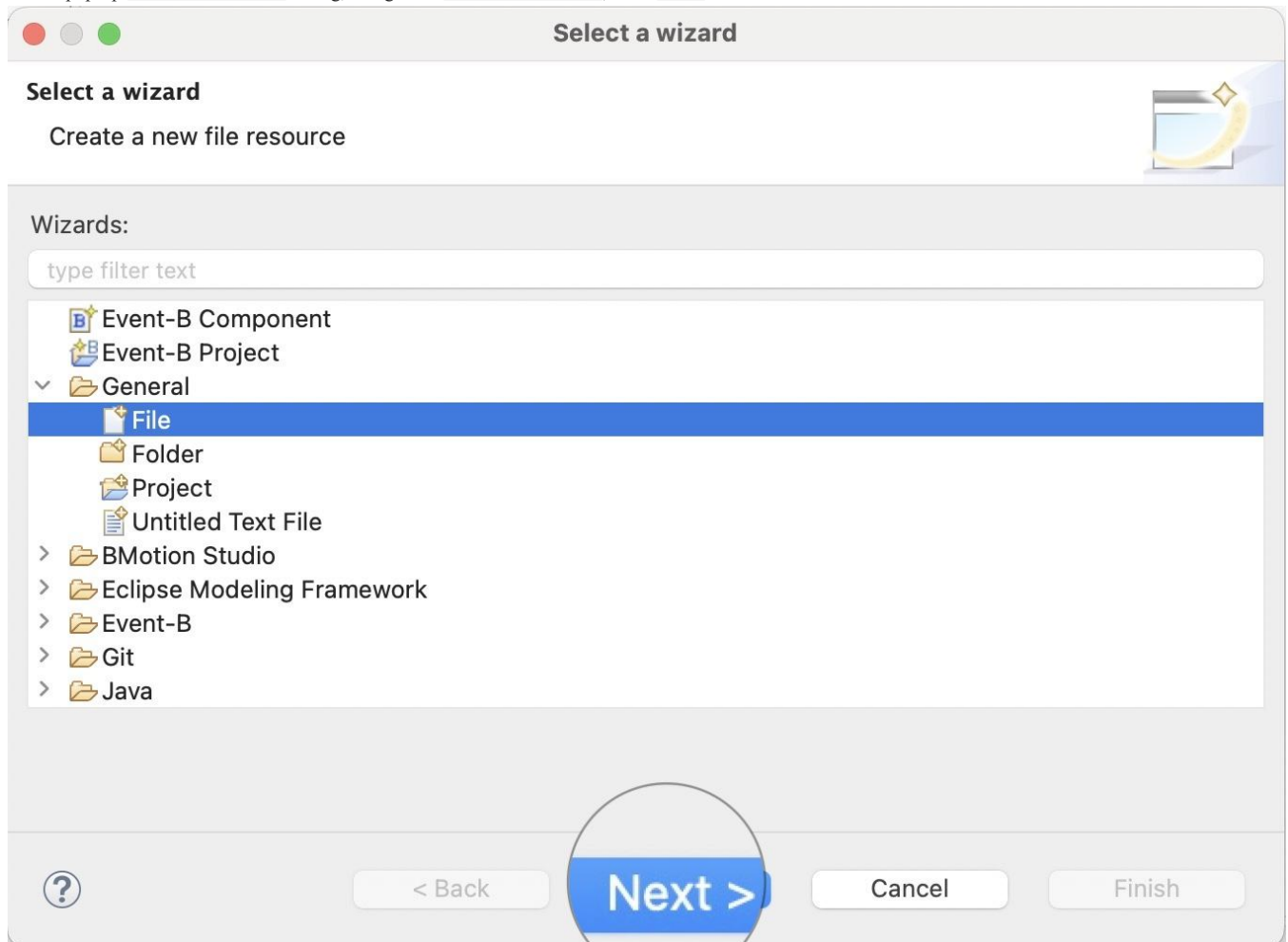
2.4.2 Task 2. Create an XContext

Introduction

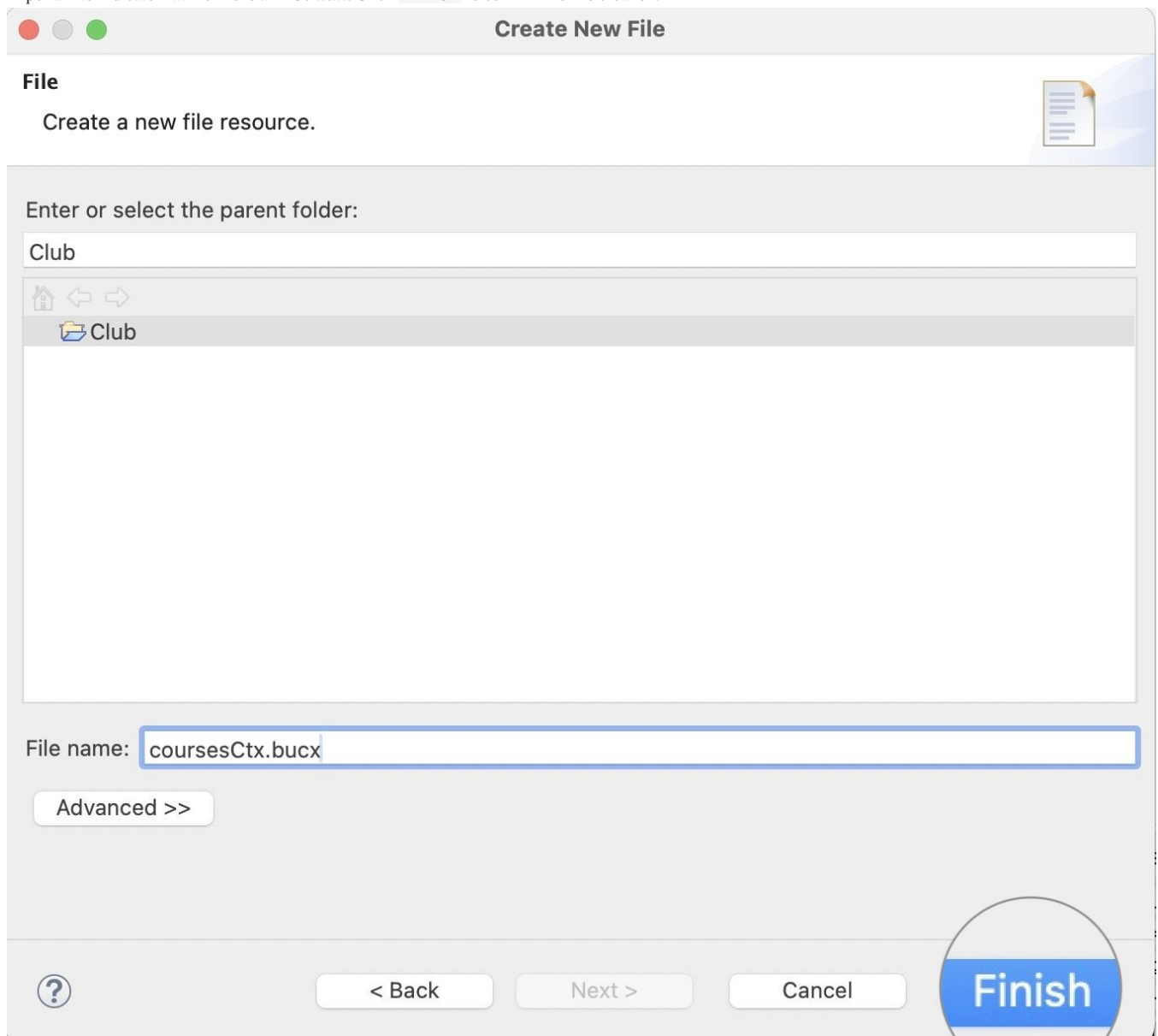
The purpose of this task is to create a simple XContext within the newly created project.

Step 1. Create a New XContext Named `coursesCtx.bucx`

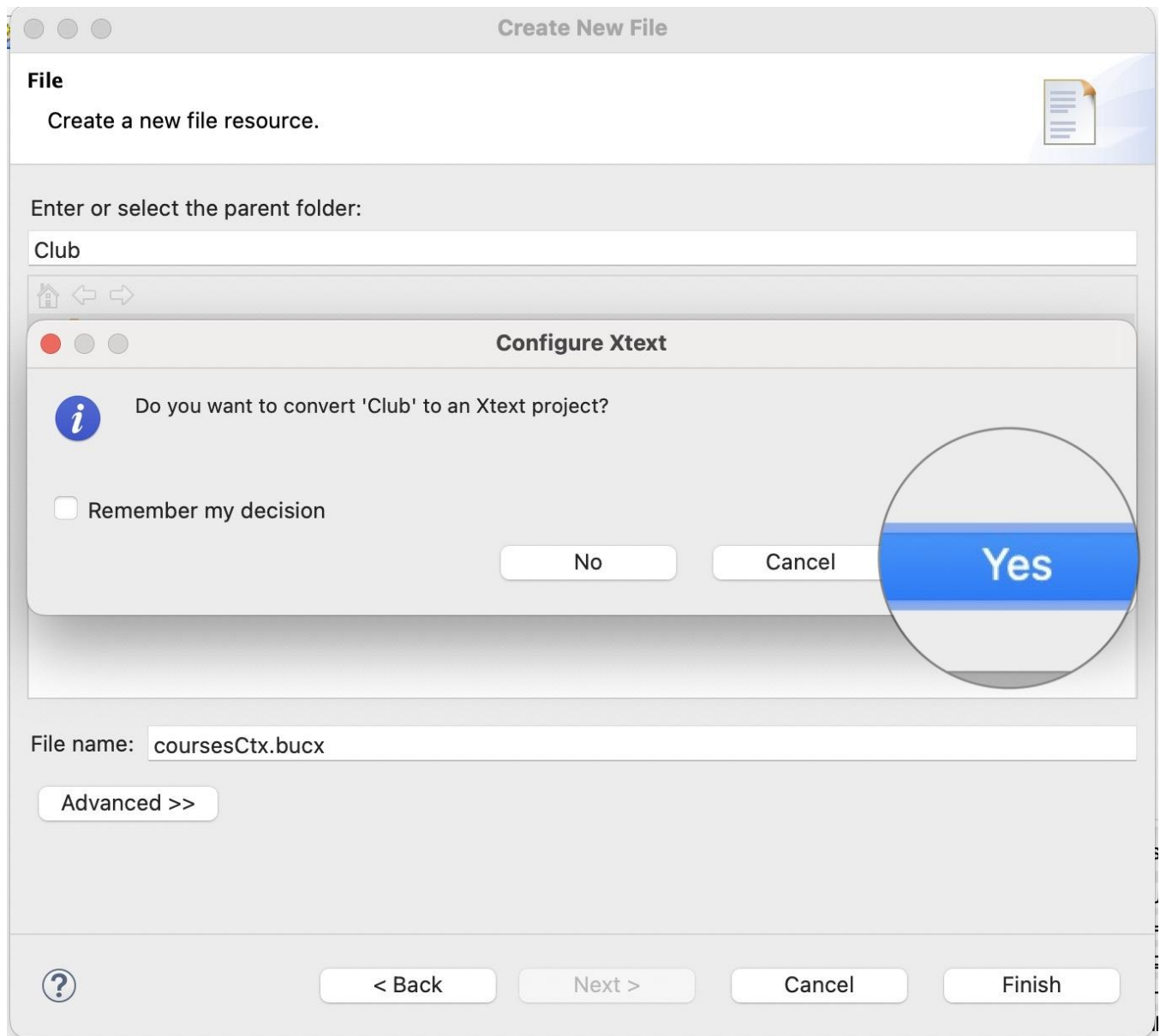
- Use the menu `File -> New -> Other` to open the `Select a wizard` dialog.
- On the pop-up `Select a wizard` dialog, navigate to `General -> File`, click `Next`.



- On the `Create New File` dialog, choose `Club` project as the parent folder, and put `coursesCtx.bucx` as the `File name`. The file extension `.bucx` is important to indicate that the file is an *XContext*. Click `Finish` to confirm the file creation.



- Important:** A pop-up dialog will be displayed asking to convert the `Club` project to an *XText* project, please answer **Yes**. This enables the *XText* builder to work automatically for converting CamilleX constructs to Rodin constructs.



(If you miss this step, you can invoke it via right click on the `Club` project from the *Event-B Explorer* and `Configure -> Convert to XText Project`). The new created file `coursesCtx.bucx` will be opened automatically in an editor. It has some error markers and we will fix this in the next step.

Step 2. Set the Content of `courseCtx.bucx`

- Using the editor, set the content of `coursesCtx.bucx` as follows.

```
context coursesCtx
sets
  CRS    // The set of all courses
constants
  m      // The maximum number of courses
axioms
  @axm0_1: finite(CRS) // There can only be a finite number of courses
  @axm0_2: m ∈ N1     // The maximum number of courses is a non-zero natural number
theorem @thm0_1: 0 < m // The maximum number of courses is positive
end
```

```
coursesCtx.bucx X
context coursesCtx
sets
  CRS // The set of all courses
constants
  m // The maximum number of courses
axioms
  @axm0_1: finite(CRS) // There can only be a finite number of courses
  @axm0_2: m ∈ ℕ1 // The maximum number of courses is a non-zero natural number
theorem @thm0_1: 0 < m // The maximum number of courses is positive
axiom
  @axm0_3: m ≤ card(CRS)
end
```

TYPESETTING MATHEMATICAL SYMBOLS

In order to typeset Event-B mathematical symbols, e.g., \mathbb{N}_1 , there are three different approaches.

1. Using *Content Assist*. *Content Assist* can translate *ASCII* characters into Unicode symbols. For example, when typing `NAT` and invoking content assist (e.g., on `Ctrl` + `Space` on Mac OS), a dropdown list will appear with options for typesetting `N` and `N1`.

The screenshot shows a code editor window titled `*coursesCtx.bucx`. The code is as follows:

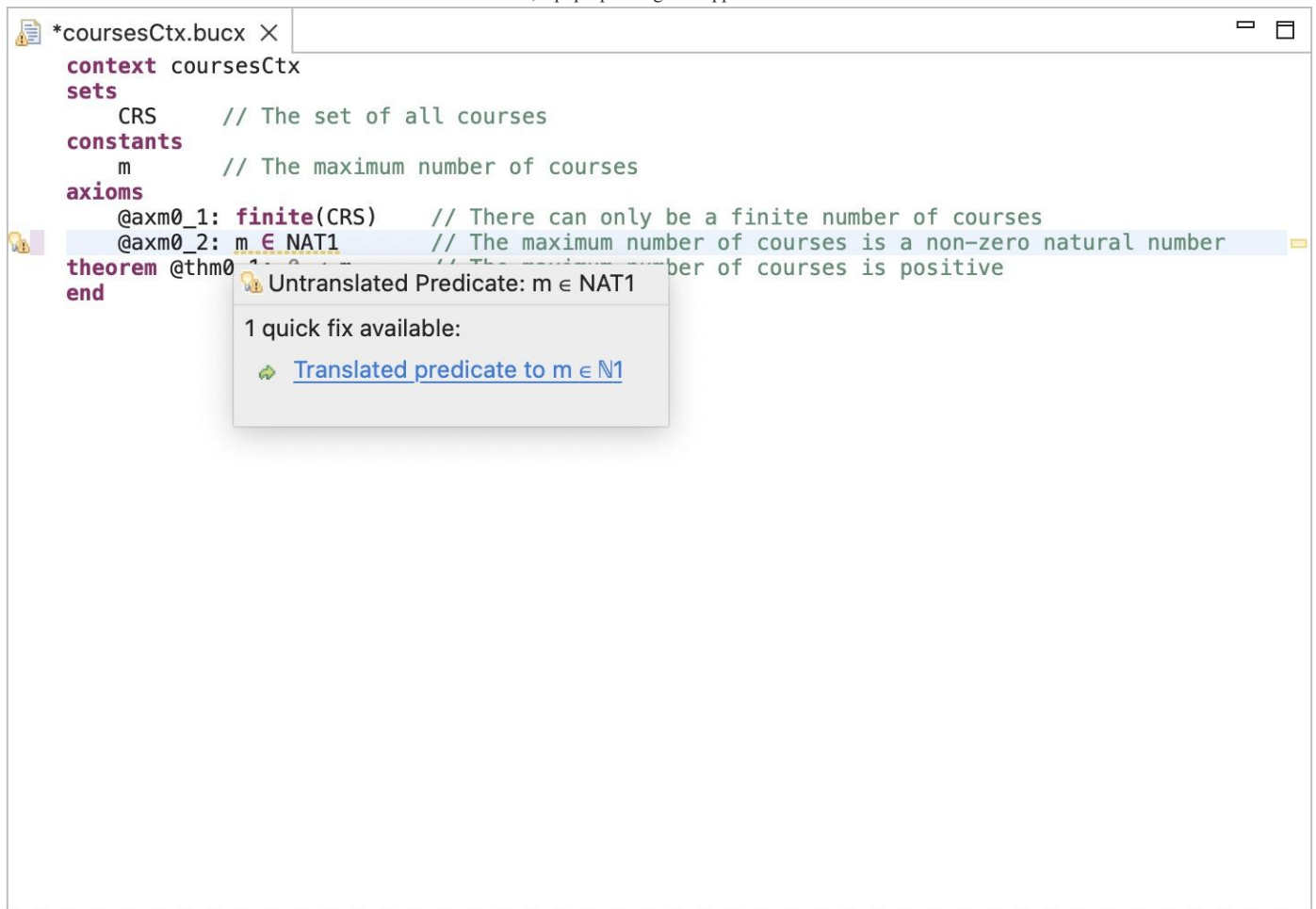
```

context coursesCtx
sets
  CRS      // The set of all courses
constants
  m        // The maximum number of courses
axioms
  @axm0_1: finite(CRS) // There can only be a finite number of courses
  @axm0_2: m ∈ NAT     // The maximum number of courses is a non-zero natural number
theorem @thm0_1: 0 < N
end

```

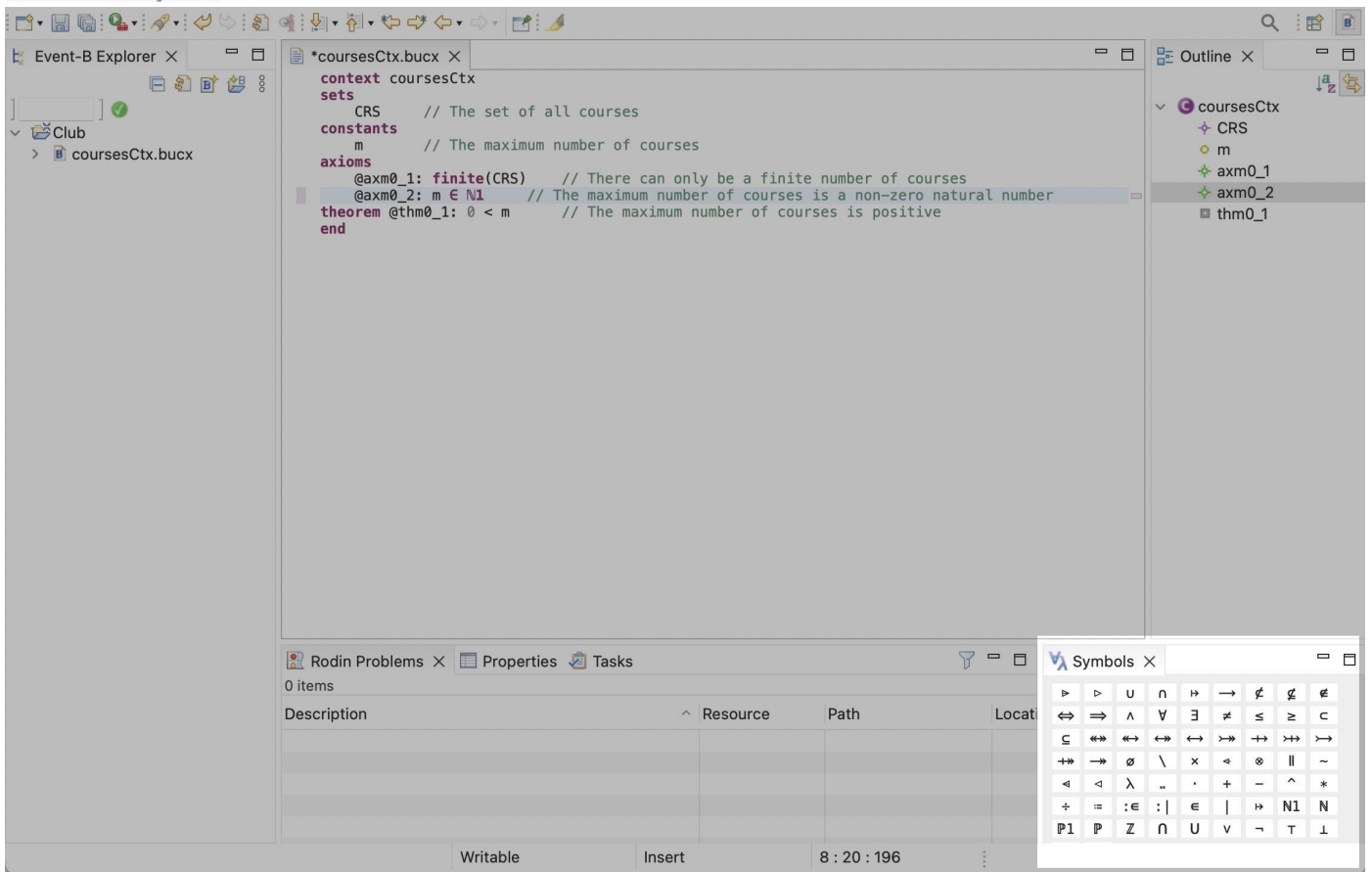
A dropdown menu is open below the line `@axm0_2: m ∈ NAT`, showing two options: `N` and `N1`. The `N` option is currently selected.

2. Using *Quick Fix*. The *CamilleX* editor offer quick fixes for ASCII untranslated formula. Untranslated formula are indicated by warnings with yellow squiggly lines under the formula. Hover the mouse over the untranslated formulae, a pop-up dialog will appear to offer to translate the formulae.



3. Using *Symbols* Table. Symbols can be inserted into the *CamilleX* editor. (If the *Symbols* table is not visible in your Rodin, you can open it from the menu `Window ->`

Show View -> Symbols.



Step 3. Save the `coursesCtx.bucx` file

Save the file `coursesCtx.bucx`, the *XText* builder will generate Rodin context `coursesCtx` automatically.

Conclusion

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

The screenshot shows the Event-B Explorer interface with the following components:

- Event-B Explorer (Left Panel):** Displays a tree view with 'Club' expanded, showing 'coursesCtx' and 'coursesCtx.bucx'.
- coursesCtx.bucx (Main Editor):** Contains the following code:


```
context coursesCtx
sets
  CRS // The set of all courses
constants
  m // The maximum number of courses
axioms
  @axm0_1: finite(CRS) // There can only be a finite number of courses
  @axm0_2: m ∈ N1 // The maximum number of courses is a non-zero natural number
theorem @thm0_1: 0 < m // The maximum number of courses is positive
end
```
- Outline (Right Panel):** Shows the structure of 'coursesCtx' with elements: CRS, m, axm0_1, axm0_2, and thm0_1.
- Rodin Problems (Bottom Panel):** Shows a table with 0 items. The table has columns: Description, Resource, Path, and Location.
- Symbols (Bottom Right Panel):** A grid of mathematical symbols for insertion.
- Status Bar (Bottom):** Shows 'Writable', 'Insert', and '9 : 1 : 257'.

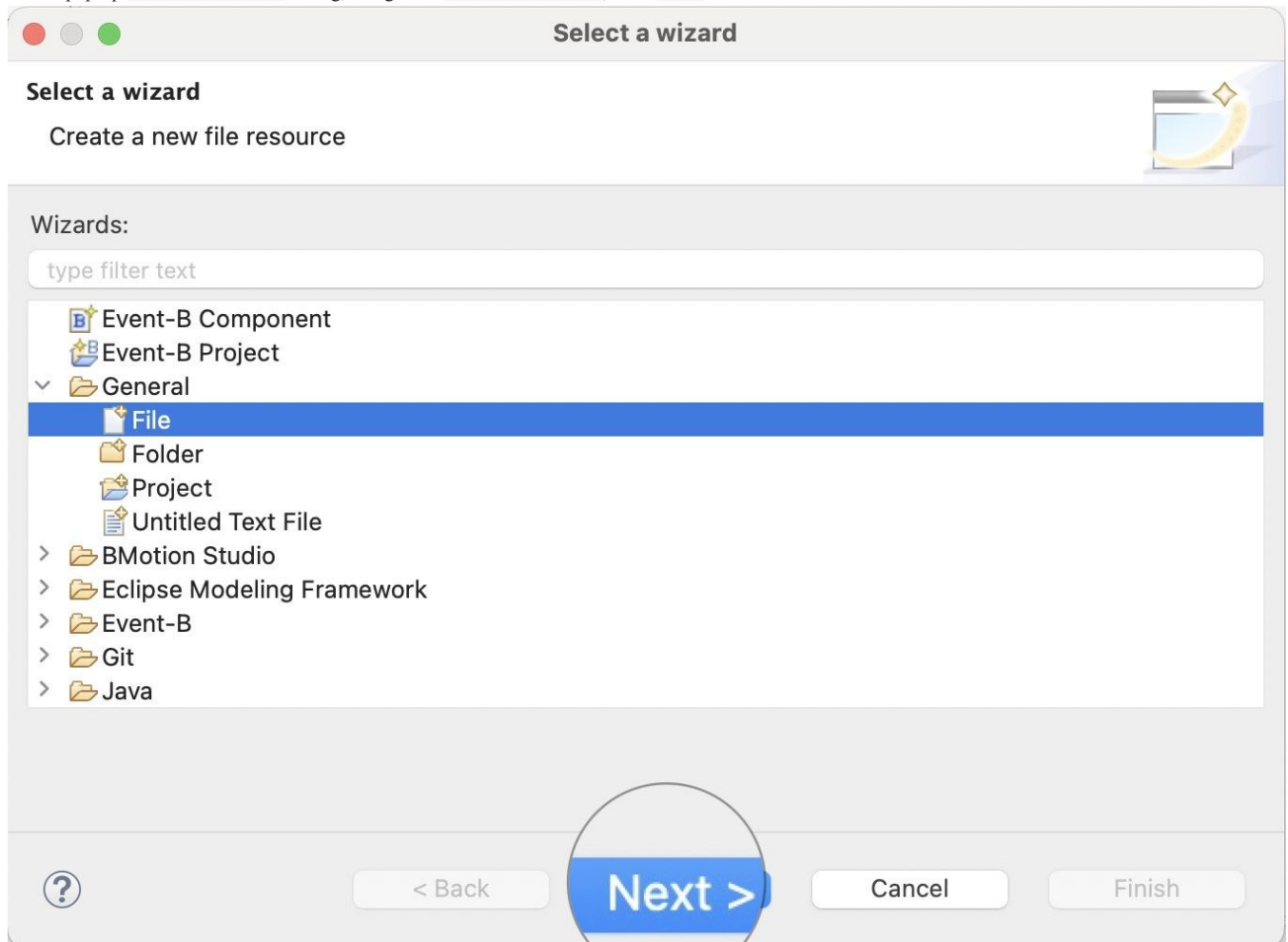
2.4.3 Task 3. Create an XMachine

Introduction

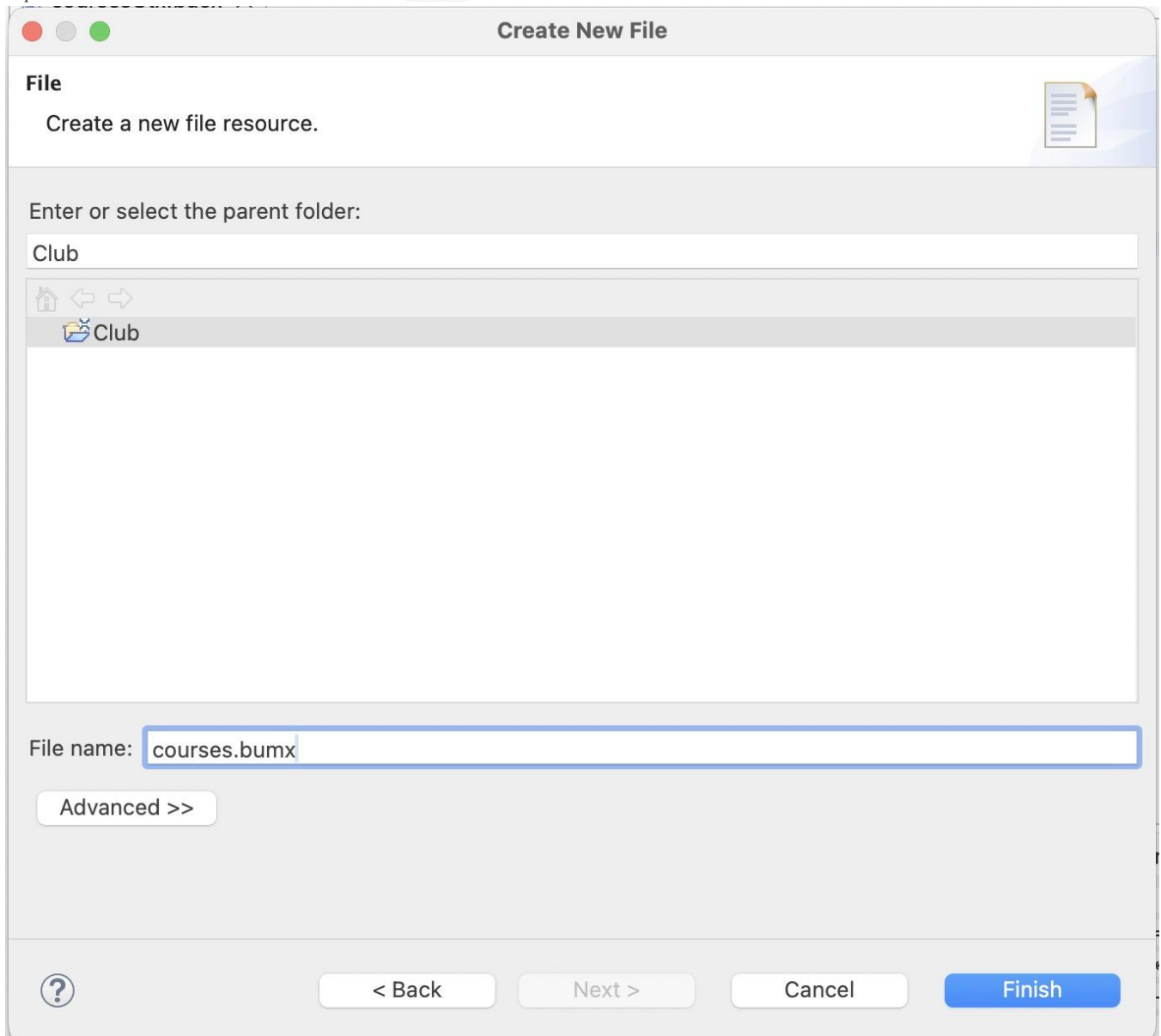
The purpose of this task is to create a simple XMachine within the newly created project.

Step 1. Create a New XMachine Named `courses.bumx`

- Use the menu `File -> New -> Other` to open the `Select a wizard` dialog.
- On the pop-up `Select a wizard` dialog, navigate to `General -> File`, click `Next`.



- On the **Create New File** dialog, choose **Club** project as the parent folder, and put **courses.bumx** as the **File name**. The file extension **.bumx** is important to indicate that the file is an *XMachine*. Click **Finish** to confirm the file creation.



Step 2. Set the Content of `course.bumx`

- Using the editor, set the content of `courses.bumx` as follows.

```

machine courses

sees coursesCtx

variables
  crs    // The set of existing courses

invariants
  @inv0_1: crs  $\subseteq$  CRS

theorem
  @thm0_2: finite(crs)

invariant
  @inv0_2: card(crs)  $\leq$  m

event INITIALISATION
begin
  @act1: crs =  $\emptyset$ 
end

```

```

/*
 * Event to open a set of courses using non-deterministic assignment.
 */
event OpenCourses
when
  @grd0_1 : card(crs) ≠ m
  theorem @thm0_3 : crs = CRS
then
  @act0_1 : crs := | crs ⊂ crs' ∧ card(crs') ≤ m
end

/*
 * Event to close a set of courses using event parameters
 */
anticipated event CloseCourses
any cs
where
  @grd1: cs ⊆ crs
  @grd2: cs ≠ ∅
then
  @act1: crs = crs \ cs
end

```

coursesCtx.bucx courses.bumx ×

```

machine courses

sees coursesCtx

variables
  crs    // The set of existing courses

invariants
  @inv0_1: crs ⊆ CRS

⊖ theorem
  @thm0_2: finite(crs)

⊖ invariant
  @inv0_2: card(crs) ≤ m

⊖ event INITIALISATION
begin
  @act1: crs = ∅
end

⊖ /*
  * Event to open a set of courses using non-deterministic assignment.
  */
⊖ event OpenCourses
when
  @grd0_1 : card(crs) ≠ m
  theorem @thm0_3 : crs = CRS
then
  @act0_1 : crs := | crs ⊂ crs' ∧ card(crs') ≤ m
end

⊖ /*
  * Event to close a set of courses using event parameters
  */
⊖ anticipated event CloseCourses
any cs
where
  @grd1: cs ⊆ crs
  @grd2: cs ≠ ∅
then
  @act1: crs = crs \ cs
end

```

Step 3. Save the `courses.bumx` file

Save the file `courses.bumx`, the *XText* builder will generate Rodin context `courses` automatically.

Conclusion

By now, the XMachine “courses.bucx” and the corresponding Rodin Machine “courses” should be visible in the Event-B Explorer. courses in the File Explorer