WEiTI

FXU 5.0

Framework for eXecutable UML 5.0

Karol Redosz under the supervision of Anna Derezińska
2014-02-27

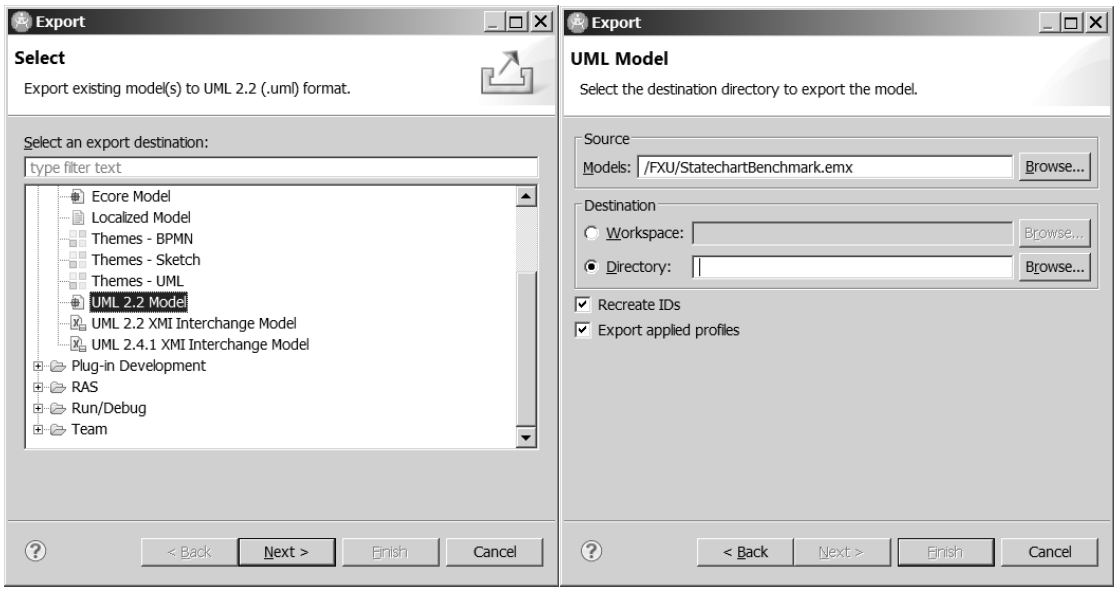
This document describes a process of creating an executable application using the FXU framework and shows changes introduced in the new version of the tool.

FXU is a framework used for creating applications according to MDE (Model-Driven Engineering) methodology. It performs transformation from UML class and state diagrams into a C# implementation. This document is an updated version of the FXU User Guide created by Marian Szczykulski. In this section, there is presented a simple example, which shows how to use the FXU Generator in order to generate a C# code from a UML model. For more information about FXU please visit the project homepage: *http://galera.ii.pw.edu.pl/~adr/FXU/*.

1. Creating a UML model

In the first step, a UML model has to be created. In the example, *IBM Rational Software Architect 9.0* has been used, but it is possible to use any CASE tool which supports exporting the UML model into the UML format of Eclipse (*.uml*).

The created model has to be exported to the UML format of Eclipse. It can be done by clicking "*File→Export*", selecting "*UML 2.2 Model*" on the "*Other*" section and following wizard's steps. It is important to select an "*Export applied profiles*" option in the second frame. Thanks to this, the *FXU Generator* will be able to read all applied profiles in the model and generate appropriate C# code.



Export

Select

Export existing model(s) to UML 2.2 (.uml) format.

Select an export destination:

type filter text

- Ecore Model
- Localized Model
- Themes - BPMN
- Themes - Sketch
- Themes - UML
- UML 2.2 Model
- UML 2.2 XMI Interchange Model
- UML 2.4.1 XMI Interchange Model
- Plug-in Development
- RAS
- Run/Debug
- Team

< Back | Next > | Finish | Cancel

Export

UML Model

Select the destination directory to export the model.

Source

Models: /FXU/StatechartBenchmark.emx

Destination

[ ] Workspace:

[● Directory:]

[✓] Recreate IDs

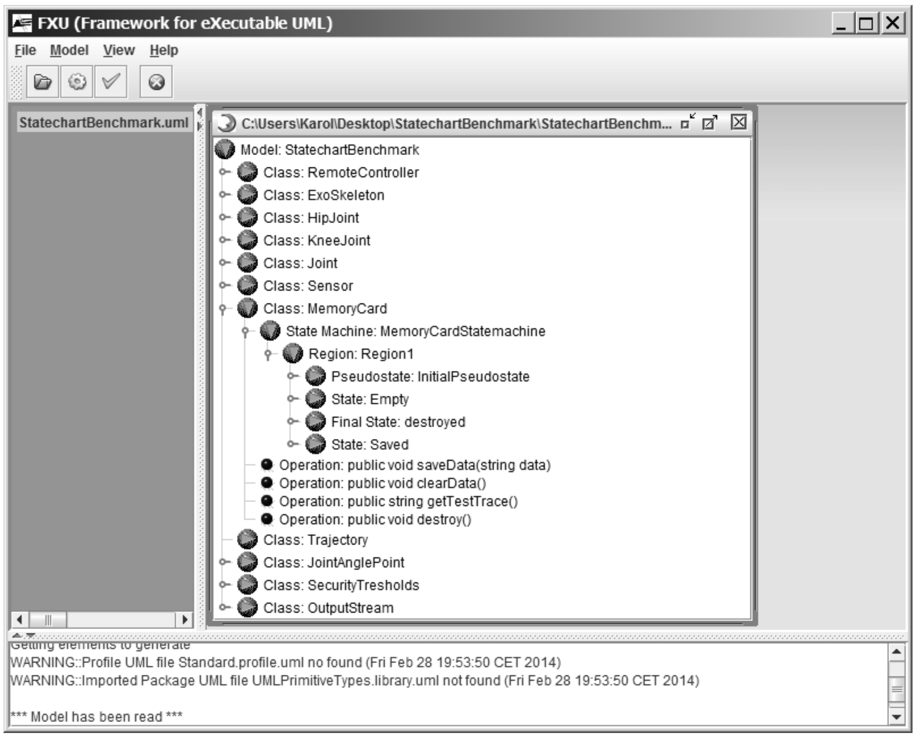
[✓] Export applied profiles

< Back | Next > | Finish | Cancel

Fig. 1. RSA 9.0 Export Wizard.

2. Launching the FXU Generator and loading the UML model

The generator can be launched by double clicking the "*FXU Generator 5.0.jar*" file (the application requires JRE1.7). In order to load the model click "*File→Open*" and select a file with the exported model in your file system.


FXU (Framework for eXecutable UML)
File
Model
View
Help
StatechartBenchmark.uml
C:\Users\Karol\Desktop\StatechartBenchmark\StatechartBenchm...

```
Model: StatechartBenchmark
    Class: RemoteController
    Class: ExoSkeleton
    Class: HipJoint
    Class: KneeJoint
    Class: Joint
    Class: Sensor
    Class: MemoryCard
        State Machine: MemoryCardStatemachine
            Region: Region1
                Pseudostate: InitialPseudostate
                State: Empty
                Final State: destroyed
                State: Saved
        Operation: public void saveData(string data)
        Operation: public void clearData()
        Operation: public string getTestTrace()
        Operation: public void destroy()
    Class: Trajectory
    Class: JointAnglePoint
    Class: SecurityTresholds
    Class: OutputStream
```

```
Getting elements to generate
WARNING::Profile UML file Standard.profile.uml no found (Fri Feb 28 19:53:50 CET 2014)
WARNING::Imported Package UML file UMLPrimitiveTypes.library.uml not found (Fri Feb 28 19:53:50 CET 2014)

*** Model has been read ***
```

Fig. 2. A UML model loaded to the FXU Generator.

Now, the model is loaded and visualized in a tree-like form which represents the real hierarchy in the original UML model (Figure 2). It is important to check logs in the result frame at the bottom of the main window. There are a few warnings. They indicate that the FXU loader has not been able to find files with UML profiles that are applied in the model. In this case, stereotypes from these profiles were not used, so warnings can be ignored.

3. Validating the UML model (optional)

In order to perform model validation click "*Model→Validate Model*". The dialog window with a message "Model is valid" should appear. In the result frame, at the bottom of the main window, additional messages should appear.

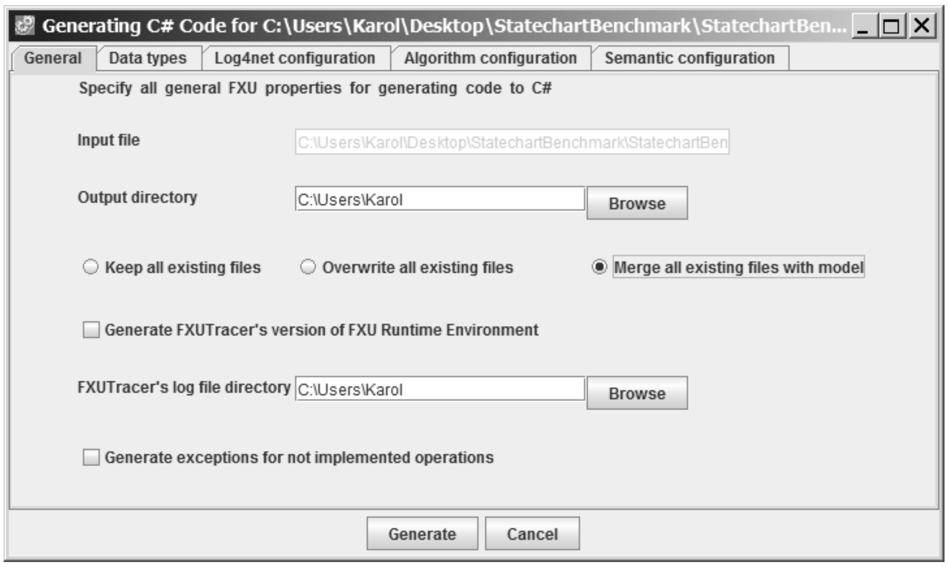
4. Generating C# code

In order to generate C# code from the loaded model click "*Model→Generate C# Code*". In the generation window, there is a possibility to configure some features of the generated code such as algorithms for the *FXU Runtime Environment*, a destination path, default data types or an application logger configuration. The generation window contains five tabs but only the first one (General Tab) was extended in the FXU 5.0.

General Tab (extended in the FXU 5.0):

- Output directory – indicates a place where generated C# code will be stored,

- Keep all existing files – if selected, all existing files of the same name in the output directory will be kept,
- Overwrite all existing files – if selected, all existing files of the same name in the output directory will be replaced by new ones,
- Merge all existing files with model – if selected, all existing files in the output directory will be merged with associated classes of the model (new option in the FXU 5.0),
- Generate FXU Tracer's version of FXU Runtime Environment – if selected, there will be application generated with the FXU Runtime Environment, which enables tracing of a state machines execution**,**
- Generate exception for not implemented operation – if selected, there will be exceptions generated for unimplemented methods. If not selected, dummy return values for unimplemented methods will be generated.


Generating C# Code for C:\Users\Karol\Desktop\StatechartBenchmark\StatechartBen...

General | Data types | Log4net configuration | Algorithm configuration | Semantic configuration

Specify all general FXU properties for generating code to C#

Input file: C:\Users\Karol\Desktop\StatechartBenchmark\StatechartBen

Output directory: C:\Users\Karol

○ Keep all existing files ○ Overwrite all existing files ● Merge all existing files with model

☐ Generate FXUTracer's version of FXU Runtime Environment

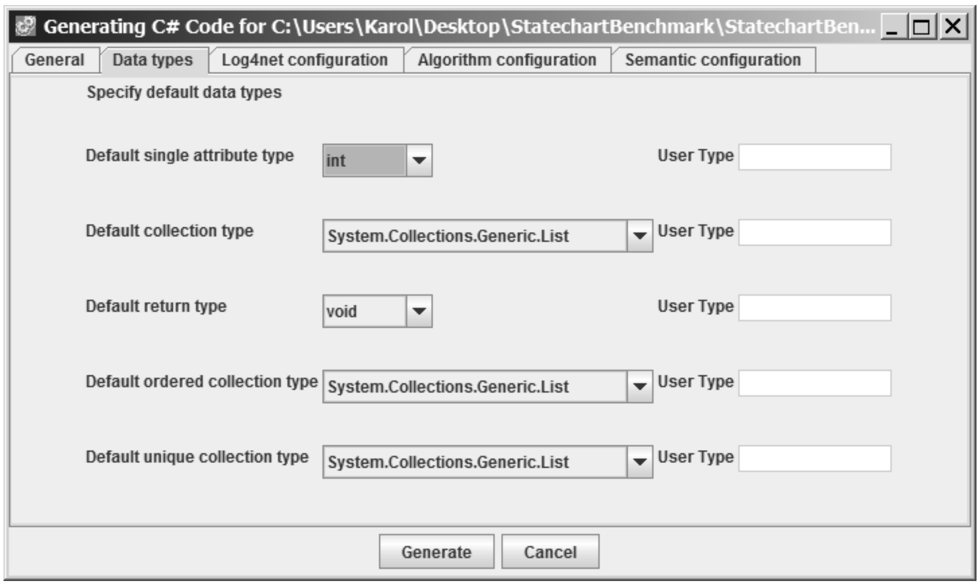
FXUTracer's log file directory: C:\Users\Karol

☐ Generate exceptions for not implemented operations

Fig. 3. FXU Generator – general properties.

<u>Data Types Tab:</u>

- Default single attribute type,
- Default collection type,
- Default return type,
- Default ordered collection type,
- Default unique collection type.


Generating C# Code for C:\Users\Karol\Desktop\StatechartBenchmark\StatechartBen...

General | Data types | Log4net configuration | Algorithm configuration | Semantic configuration

Specify default data types

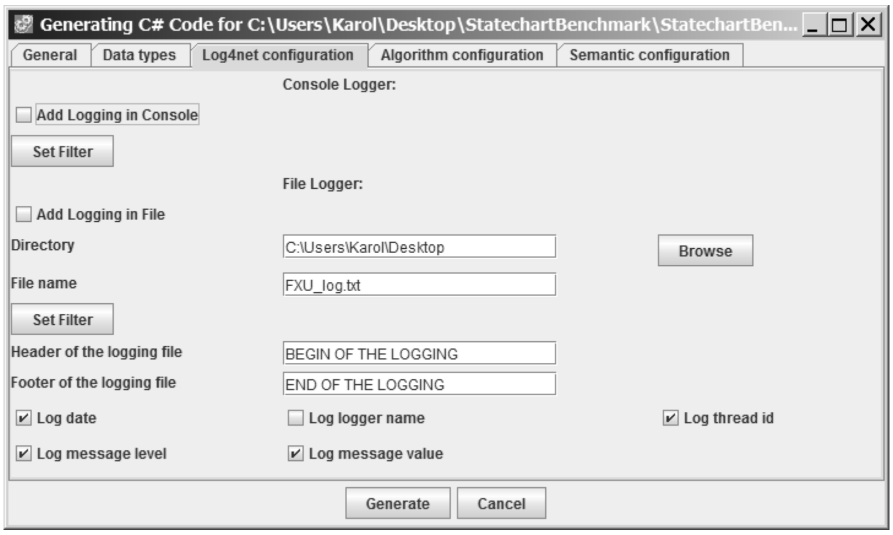
Default single attribute type	int	User Type	
Default collection type	System.Collections.Generic.List	User Type	
Default return type	void	User Type	
Default ordered collection type	System.Collections.Generic.List	User Type	
Default unique collection type	System.Collections.Generic.List	User Type	

Generate | Cancel

Fig. 4. FXU Generator – default data types.

<u>Log4net Configuration Tab:</u>

- Add Logging in Console,
- Add Logging in file,
- Header of the logging file,
- Footer of the logging file,
- Logging information.


Generating C# Code for C:\Users\Karol\Desktop\StatechartBenchmark\StatechartBen...

General | Data types | Log4net configuration | Algorithm configuration | Semantic configuration

Console Logger:

☐ Add Logging in Console

Set Filter

File Logger:

☐ Add Logging in File

Directory	C:\Users\Karol\Desktop	Browse
File name	FXU_log.txt	
Header of the logging file	BEGIN OF THE LOGGING	
Footer of the logging file	END OF THE LOGGING	

Set Filter

☑ Log date ☐ Log logger name ☑ Log thread id

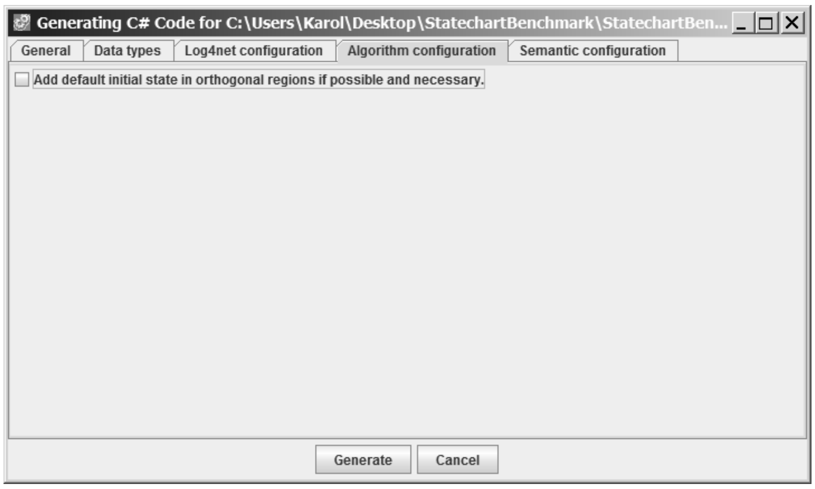
☑ Log message level ☑ Log message value

Generate | Cancel

Fig. 5. FXU Generator – log4net properties.

Algorithm Configuration Tab:

- Add default initial state in orthogonal regions if possible and necessary – if selected, the generator will fix the error situation when orthogonal region has no initial state.

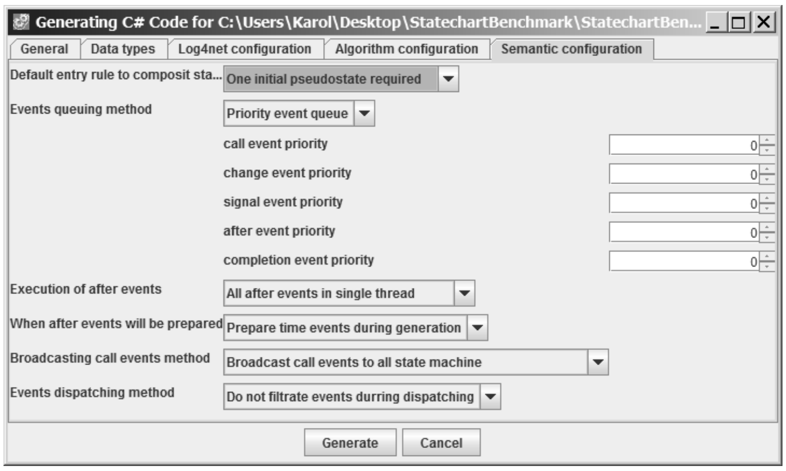

Generating C# Code for C:\Users\Karol\Desktop\StatechartBenchmark\StatechartBen...

☐ Add default initial state in orthogonal regions if possible and necessary.

Fig. 6. FXU Generator – algorithm properties.

Semantic Configuration Tab:

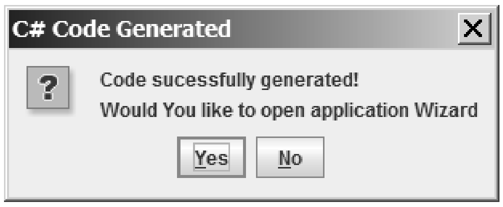
- Default entry rule to composit states,
- Events queuing method,
- Execution of after events,
- When after events will be prepared,
- Broadcatsing of call events method,
- Events dispatching method.


Generating C# Code for C:\Users\Karol\Desktop\StatechartBenchmark\StatechartBen...

Default entry rule to composit sta...	One initial pseudostate required	
Events queuing method	Priority event queue	
	call event priority	0
	change event priority	0
	signal event priority	0
	after event priority	0
	completion event priority	0
Execution of after events	All after events in single thread	
When after events will be prepared	Prepare time events during generation	
Broadcasting call events method	Broadcast call events to all state machine	
Events dispatching method	Do not filtrate events durring dispatching	

Fig. 7. FXU Generator – UML semantic variants.

In order to start a C# code generation process, click the "Generate" button. If generation process is completed an appropriate message dialog window will appear (Fig. 8).

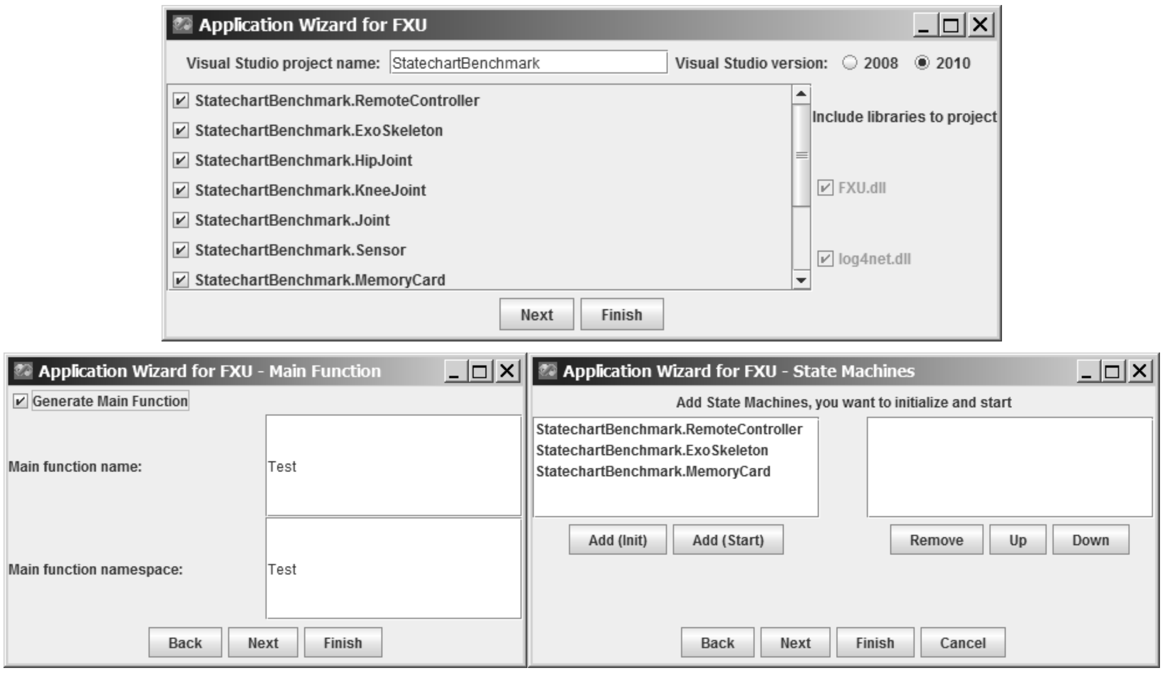

C# Code Generated

Code sucessfully generated!
Would You like to open application Wizard

Fig. 8. The message window appeared after successful code generation.

5. Creating a Visual Studio Project (optional)

The last step is a Microsoft Visual Studio 2008/2010 project generation. It is an optional step and can be omitted. Figure 9 shows first three windows of the FXU Application Wizard. There is a possibility to specify a project name, to generate the Main function and specify its containing class namespace and name. In the third window of the wizard, selected state machines can be initialized and started.


Application Wizard for FXU

Visual Studio project name: StatechartBenchmark
Visual Studio version: ○ 2008 ● 2010

- ✓ StatechartBenchmark.RemoteController
- ✓ StatechartBenchmark.ExoSkeleton
- ✓ StatechartBenchmark.HipJoint
- ✓ StatechartBenchmark.KneeJoint
- ✓ StatechartBenchmark.Joint
- ✓ StatechartBenchmark.Sensor
- ✓ StatechartBenchmark.MemoryCard

Include libraries to project
- ✓ FXU.dll
- ✓ log4net.dll

Application Wizard for FXU - Main Function

✓ Generate Main Function
Main function name: Test
Main function namespace: Test

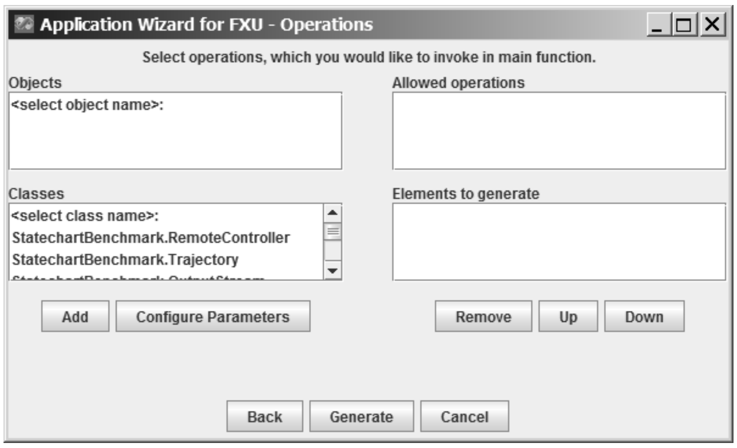
Application Wizard for FXU - State Machines

Add State Machines, you want to initialize and start

StatechartBenchmark.RemoteController
StatechartBenchmark.ExoSkeleton
StatechartBenchmark.MemoryCard

Fig. 9. First three windows of Application Wizard.

In the last window of the wizard, there is a possibility to specify which operations have to be invoked in the Main function. There is also a possibility to configure attributes for invoked operations. The window is shown in the figure 10.


Application Wizard for FXU - Operations

Select operations, which you would like to invoke in main function.

Objects

<select object name>:

Allowed operations

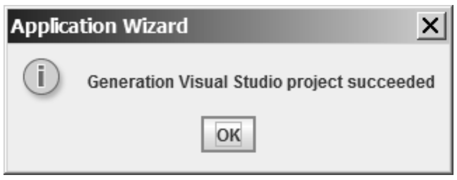
Classes

<select class name>:
StatechartBenchmark.RemoteController
StatechartBenchmark.Trajectory

Elements to generate

Fig. 10. Application Wizard – configuration of operations in the Main function.

In order to generate the *Microsoft Visual Studio* project click the "*Generate*" button. If the generation process is successful, an appropriate message window will appear (Fig. 11).


Application Wizard

Generation Visual Studio project succeeded

Fig. 11. Message window appeared after successful *Microsoft Visual Studio* project generation.

The project is created and ready to open and run in the *Microsoft Visual Studio*.