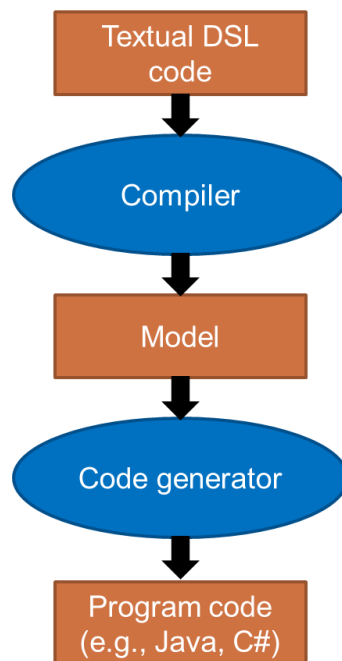# Practice 3 – Preparing the projects

Dr. Balázs Simon, 2024

1   Introduction

This guideline shows how to create the projects needed to solve the practice.

The processing of a textual DSL consist of the following steps:

The compiler builds a model from the textual DSL code. In order to be able to do this, the elements of the model and their relationships must be specified. This is the purpose of the meta-model. When defining the compiler, the mapping of the DSL source code elements to the meta-model elements must also be specified. If this mapping is accurate enough, the model can be built automatically by the compiler. By traversing the model, a code generator can produce some useful output code.

To follow this guideline, you will need the **Eclipse IDE for Java and DSL Developers** tool from the following link:

https://www.eclipse.org/downloads/packages/release/2023-12/r/eclipse-ide-java-and-dsl-developers

Use exactly this version and this edition of Eclipse, otherwise, you may run into problems due to version conflicts in the referenced libraries.

Remarks: the **MANIFEST.MF** files appearing in this guideline are taken from a previous semester, so the version numbers in them may be different than the version numbers used by the Eclipse version downloaded from the link above. Ignore the version numbers used in this guideline, and use the

version numbers produced by your own Eclipse. Your own **MANIFEST.MF** files should only be modified by adding the parts marked with yellow background.
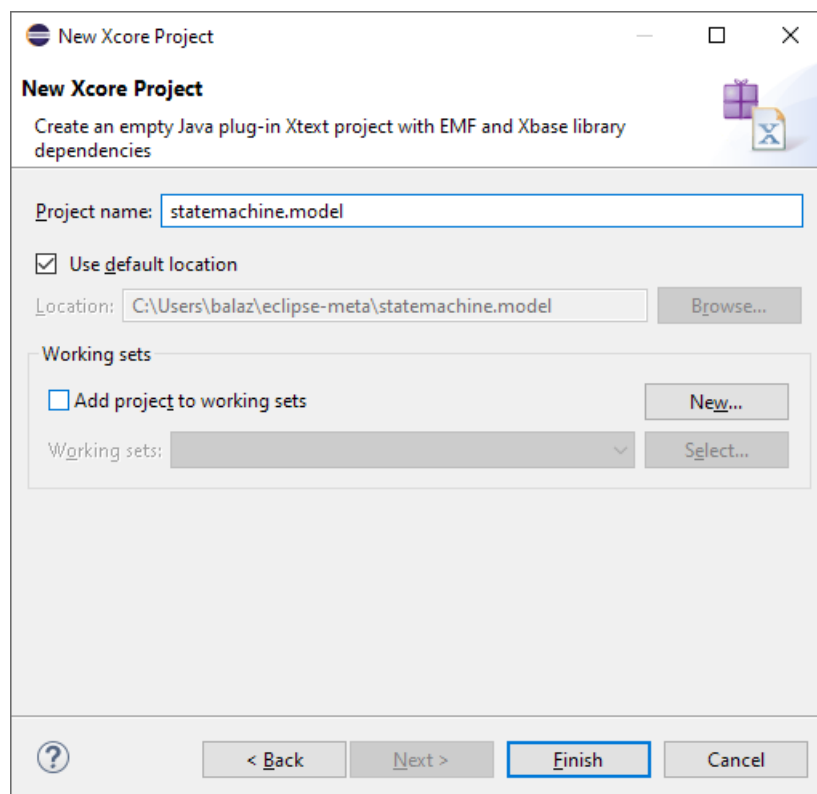
2 Goal

The goal is to create a compiler and tools support for a simple state machine description language. To prepare for this, let's create the projects as described in the following chapters!

3 Project for the meta-model: Xcore

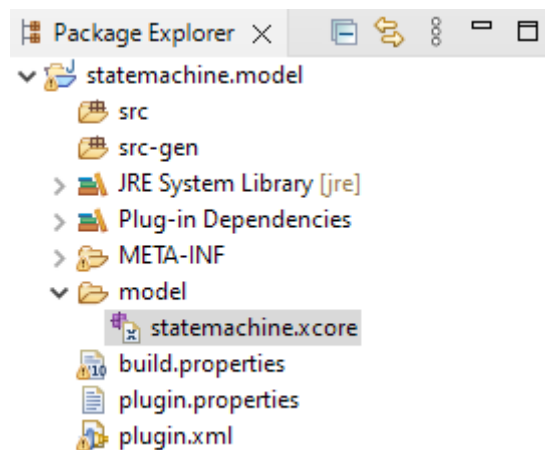Create a new **Xcore** project by clicking on **File > New > Project…**:
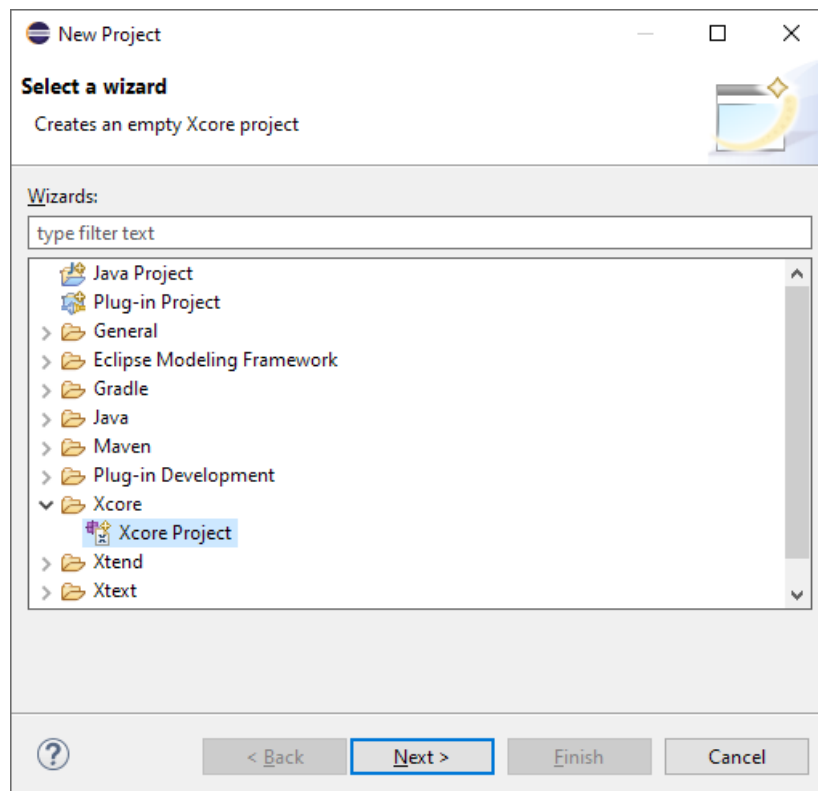
Click **Next**. The name of the project should be **statemachine.model**:
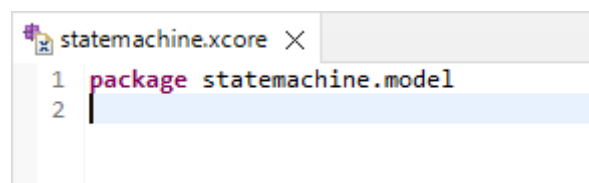

Click **Finish**.

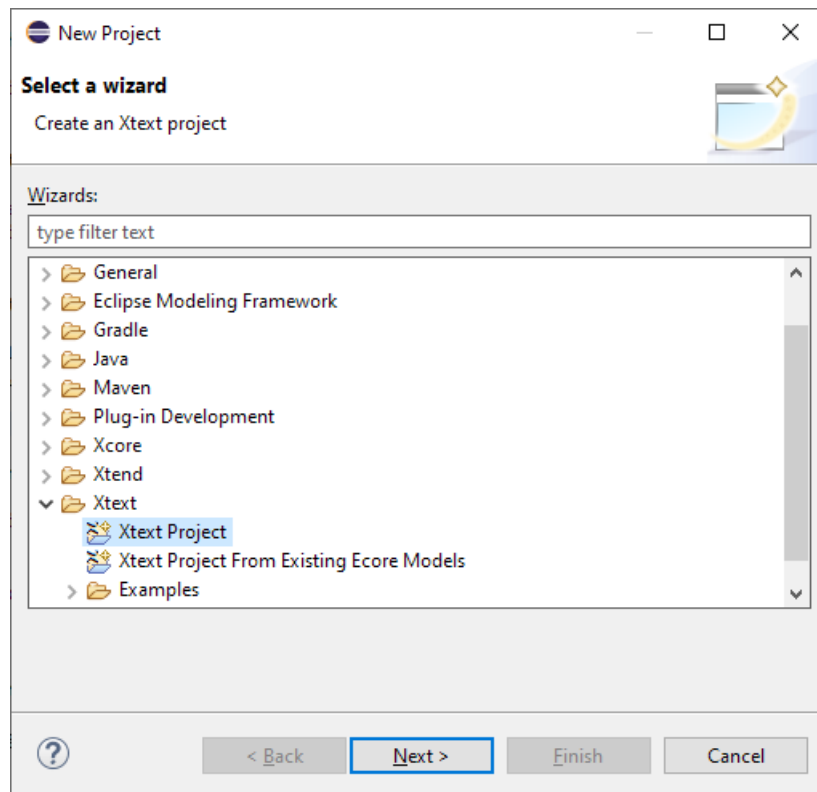Under the **model** folder, create a file called **statemachine.xcore**:

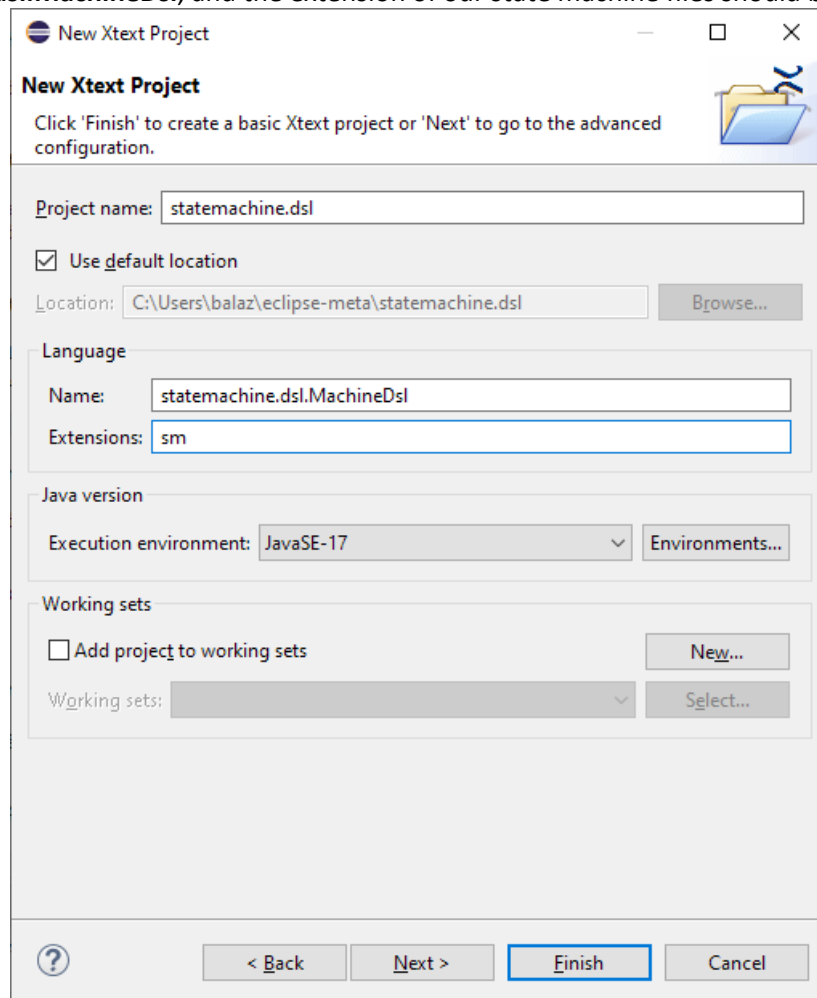The contents of the file should be the following:

4   Project for the compiler: Xtext
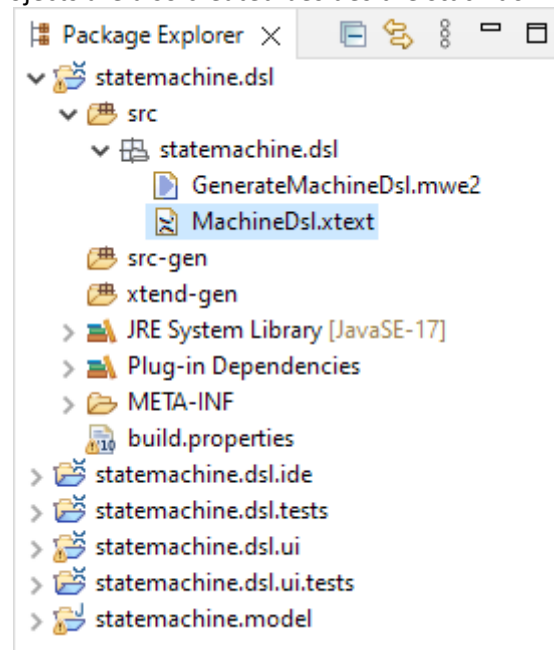
Create a new **Xtext** project by clicking on **File > New > Project…**:

Click **Next**. The name of the project should be **statemachine.dsl**, the name of the language should be **statemachine.dsl.MachineDsl**, and the extension of our state machine files should be **sm**:

Click **Finish**. Some other projects are also created besides the **statmachine.dsl** project:

For now, we are only interested in the **entities.dsl** project. The other project is useful for the Eclipse IDE support and testing:

- **statemachine.dsl.ide**: main IDE functions which are independent of the graphical IDE interface
- **statemachine.dsl.ui**: IDE functions dependent on the graphical IDE interface
- **statemachine.dsl.test**: unit tests for our DSL
- **statemachine.dsl.ui.tests**: unit tests for the IDE support of our DSL

For the compiler to be able to see our meta-model, we have to add the following line with yellow background to the **GenerateEntityDsl.mwe2** file:

```
module statemachine.dsl.GenerateMachineDsl

import org.eclipse.xtext.xtext.generator.*
import org.eclipse.xtext.xtext.generator.model.project.*

var rootPath = ".."

Workflow {

  component = XtextGenerator {
    configuration = {
      project = StandardProjectConfig {
        baseName = "statemachine.dsl"
        rootPath = rootPath
        runtimeTest = {
          enabled = true
        }
        eclipsePlugin = {
          enabled = true
        }
        eclipsePluginTest = {
          enabled = true
        }
        createEclipseMetaData = true
      }
      code = {
        encoding = "UTF-8"
        lineDelimiter = "\r\n"
        fileHeader = "/*\n * generated by Xtext \${version}\n */"
        preferXtendStubs = false
      }
    }
    language = StandardLanguage {
      name = "statemachine.dsl.MachineDsl"
      fileExtensions = "sm"
      referencedResource      =      "platform:/resource/statemachine.model/model/statemachine.xcore"

      serializer = {
        generateStub = false
      }
      validator = {
        // composedCheck = "org.eclipse.xtext.validation.NamesAreUniqueValidator"
```

```
        // Generates checks for @Deprecated grammar annotations, an IssueProvider
and a corresponding PropertyPage
        generateDeprecationValidation = true
      }
      generator = {
        generateXtendStub = true
      }
      junitSupport = {
        junitVersion = "5"
      }
    }
  }
}
```

In addition, the **META-INF/MANIFEST.MF** file must be extended with the following lines:

```
Manifest-Version: 1.0
Automatic-Module-Name: statemachine.dsl
Bundle-ManifestVersion: 2
Bundle-Name: statemachine.dsl
Bundle-Vendor: My Company
Bundle-Version: 1.0.0.qualifier
Bundle-SymbolicName: statemachine.dsl; singleton:=true
Bundle-ActivationPolicy: lazy
Require-Bundle: org.eclipse.xtext,
 org.eclipse.xtext.xbase,
 org.eclipse.equinox.common;bundle-version="3.16.0",
 org.eclipse.emf.ecore.xcore.lib,
 org.eclipse.emf.ecore.xcore,
  statemachine.model
Bundle-RequiredExecutionEnvironment: JavaSE-17
```

Also, in the **statemachine.dsl.tests** project, the **META-INF/MANIFEST.MF** file must be extended:
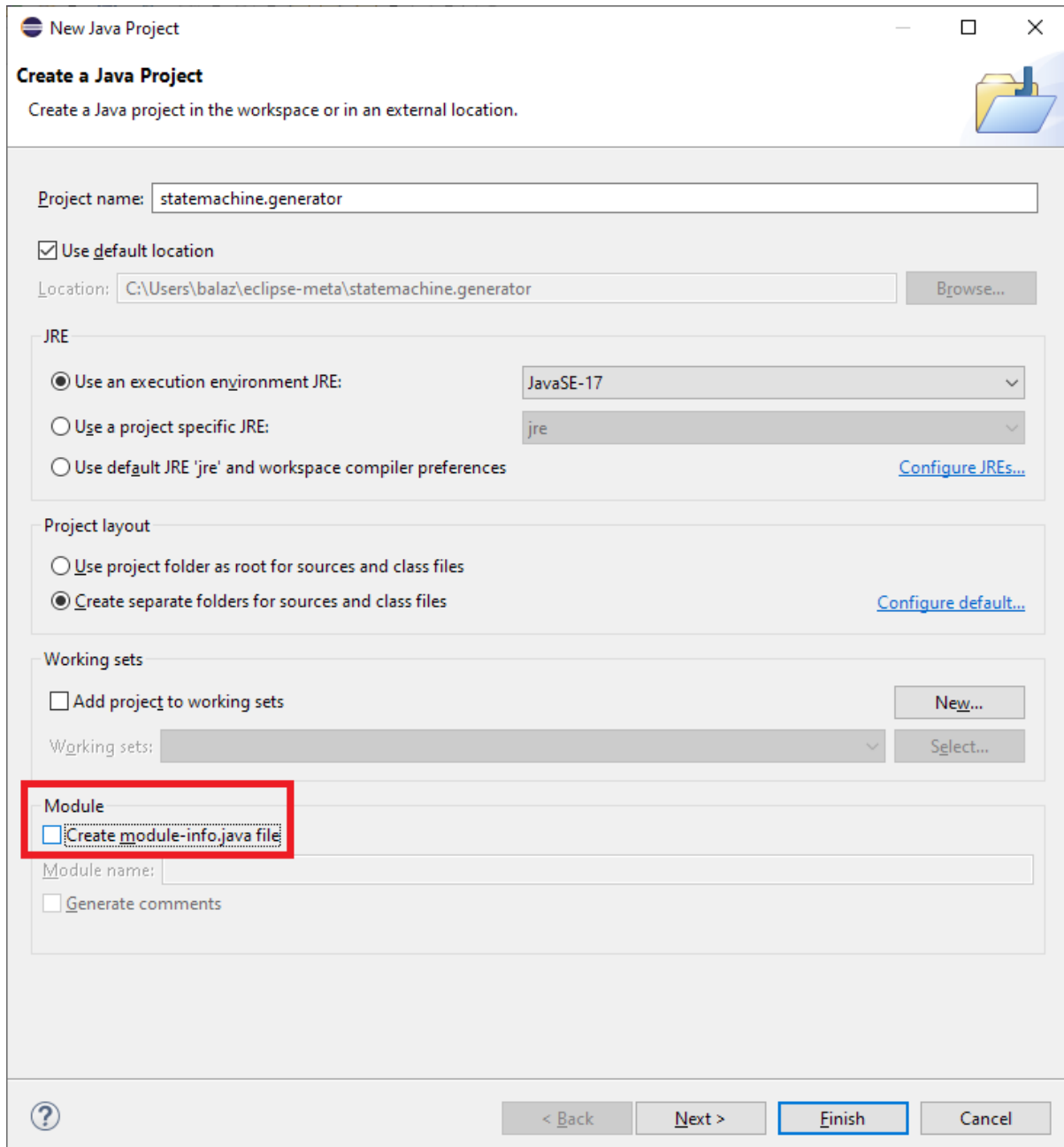
```
Manifest-Version: 1.0
Automatic-Module-Name: statemachine.dsl.tests
Bundle-ManifestVersion: 2
Bundle-Name: statemachine.dsl.tests
Bundle-Vendor: My Company
Bundle-Version: 1.0.0.qualifier
Bundle-SymbolicName: statemachine.dsl.tests; singleton:=true
Bundle-ActivationPolicy: lazy
Require-Bundle: statemachine.dsl,
 org.eclipse.xtext.testing,
 org.eclipse.xtext.xbase.testing,
  statemachine.model
Import-Package: org.junit.jupiter.api;version="[5.1.0,6.0.0)",
 org.junit.jupiter.api.extension;version="[5.1.0,6.0.0)"
Bundle-RequiredExecutionEnvironment: JavaSE-17
```

5   Project for the code generator: Xtend

Click **File > New > Java Project** to create a new Java project called **statemachine.generator**! Remove the checkmark before the „Create module-info.java file" option:

Click **Finish** to create the project! Under the **src** folder inside the **statemachine.generator** package create a file **MachineToJavaGenerator.xtend** with the following content:

Right click on the **statemachine.generator** project, and select **Configure > Convert to Xtext Project**!

Right click again on the **statemachine.generator** project, and select **Configure > Convert to Plug-in Projects**! Close the appearing window with the **Finish** button.

Add the following lines to the **META-INF/MANIFEST.MF** file:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Generator
Bundle-SymbolicName: statemachine.generator
Bundle-Version: 1.0.0.qualifier
Bundle-RequiredExecutionEnvironment: JavaSE-17
Automatic-Module-Name: statemachine.generator
Require-Bundle: org.eclipse.xtext,
 org.eclipse.xtext.xbase,
 statemachine.model
Export-Package: statemachine.generator
```

Inside the **statemachine.dsl** project modify the **META-INF/MANIFEST.MF** file as follows:

```
Manifest-Version: 1.0
Automatic-Module-Name: statemachine.dsl
Bundle-ManifestVersion: 2
Bundle-Name: statemachine.dsl
Bundle-Vendor: My Company
Bundle-Version: 1.0.0.qualifier
Bundle-SymbolicName: statemachine.dsl; singleton:=true
Bundle-ActivationPolicy: lazy
Require-Bundle: org.eclipse.xtext,
 org.eclipse.xtext.xbase,
 org.eclipse.equinox.common;bundle-version="3.16.0",
 org.eclipse.emf.ecore.xcore.lib,
 org.eclipse.emf.ecore.xcore,
 statemachine.model,
 statemachine.generator,
 org.eclipse.xtext.xbase.lib;bundle-version="2.14.0",
 org.eclipse.xtext.util,
 org.antlr.runtime;bundle-version="[3.2.0,3.2.1)"
Bundle-RequiredExecutionEnvironment: JavaSE-17
Export-Package: statemachine.dsl.serializer,
 statemachine.dsl.parser.antlr,
 statemachine.dsl.generator,
 statemachine.dsl.validation,
 statemachine.dsl,
 statemachine.dsl.services,
 statemachine.dsl.scoping,
 statemachine.dsl.parser.antlr.internal
Import-Package: org.apache.log4j
```