



Grant ANR-12-INSE-0011

ANR INS GEMOC

D1.2.1 - DSML behavioral semantics definition tools (SOFTWARE)

Task 1.2.1

Version 2.0

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

DOCUMENT CONTROL

	–: 2013/11/22	A: 2013/11/28	B: 2014/11/03	C: 2015/05/29	D:
Written by	Didier Vojtisek	Didier Vojtisek	Xavier Crégut	Xavier Crégut	
Signature					
Approved by					
Signature					

Revision index	Modifications
–	version 0.1
A	version 0.2 - addition of workflow chapter
B	version 1.0 - addition of tutorial section
C	version 2.0 - addition of Dashboard section, update the overall document.
D	

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

Authors

Author	Partner	Role
Didier Vojtisek	INRIA	Lead author
Mélanie Bats	OBEO	Contributor
Joël Champeau	ENSTA Bretagne	Contributor
Benoit Combemale	INRIA	Contributor
Xavier Crégut	IRIT – Université de Toulouse/INP	Contributor
Papa Issa DIALLO	ENSTA Bretagne	Contributor
Florent Latombe	IRIT – Université de Toulouse/INP	Contributor

ANR INS GEMOC / Task 1.2.1	Version:	2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date:	May 30, 2015
D1.2.1		

Contents

1	Introduction	6
1.1	Purpose	6
1.2	Perimeter	6
1.3	Definitions, Acronyms and Abbreviations	6
1.4	Summary	7
2	Architecture of the DSML behavioral semantics definition tools	9
2.1	Implementation of an eExecutable DSML (Executable Domain-Specific Modeling Language (xDSML))	9
2.2	xDSML development method	9
2.3	xDSML development tool	10
2.3.1	xDSML Project	10
2.3.2	Internal structure of the GEMOC Language Workbench	11
2.3.3	xDSML Metamodel	11
3	Gemoc Language Workbench: xDSML development tool workflow	12
3.1	xDSML definition	12
3.1.1	New Gemoc Language project Command	12
3.2	Domain Model definition (AS)	14
3.2.1	New EMF Project Command	15
3.2.2	Select existing EMF Project Command	15
3.3	Modelers definition (CS)	15
3.3.1	New Tree editor Command	17
3.3.2	Select existing Tree editor Command	17
3.3.3	New Xtext editor Command	17
3.3.4	Select existing Xtext editor Command	18
3.3.5	New Sirius editor Command	18
3.3.6	Select existing Sirius editor Command	18
3.4	Domain-Specific Actions	19
3.4.1	New Kermeta 3 project Command	19
3.4.2	Select existing Kermeta 3 project Command	20
3.5	Model of Concurrency and Communication (Model of Concurrency and Communication (MoCC)) definition: reusable part	20
3.5.1	New MoCC Modeling Language (MoCCML) project Command	20
3.5.2	Select existing MoCCML project Command	21
3.6	Model of Concurrency and Communication (MoCC) definition: mapping to the Abstract Syntax (AS)	21
3.6.1	New ECL file Command	21
3.6.2	New Modelh'x project Command	22
3.6.3	Select existing ECL file Command	22
3.7	Animators definition	23
4	Language workbench dashboard	24
4.1	Open the dashboard	24
4.2	Overview	24
4.3	Sections	25
4.4	Description	25

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

4.5	Links	26
4.6	Projects view	26
4.7	Synchronize	26
4.8	Open existing projects	26
5	Current implementation	27
6	Tutorials	29
6.1	Tutorial Description	29
6.2	Process Description	30
7	Conclusion	31
A	Getting started with the Language Workbench	32

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

1. Introduction

1.1 Purpose

This document presents the tools supporting the methodology for defining an Executable Domain-Specific Modeling Language (xDSML). The developed tools orchestrate a set of features proposed by the GEMOC language workbench to help the language designer in the various activities leading to the definition of an xDSML.

We describe the various steps of the process and the tools that support it.

The tools and the wizards supporting the process are integrated and deployed into the GEMOC Studio. Tutorials have been proposed to illustrate the process and the use of the GEMOC Studio.

The GEMOC Studio can be found on the download page of the GEMOC project: <http://gemoc.org/studio-download/>. The source code is available on the git server: `git+ssh://developer-name@scm.gforge.inria.fr/gitroot/gemoc-dev/gemoc-dev.git`. The version described in this deliverable is the v1.0.

The tutorials are available at <http://gemoc.github.io/gemoc-studio/>. The sources of the tutorials are available at <https://github.com/gemoc/gemoc-studio/wiki/>. The Eclipse projects of the tutorial are available on git server in `git+ssh://developer-name@scm.gforge.inria.fr/gitroot/gemoc-dev/org/gemoc/sample/`.

1.2 Perimeter

This document describes the version v2 (last version) of the software deliverable D1.2.1 (*DSML behavioral semantics definition tools*). It presents the current state of the tools provided in the Language Workbench of the GEMOC studio. These tools ease the application of the methodology defined in Task 1.1. It also includes the definition of the technical workflow associated to the tools, a dashboard to allow the language designer to manage the xDSML artifacts through the steps of the process, as well as a tutorial to explain the definition of an xDSML on a concrete example.

1.3 Definitions, Acronyms and Abbreviations

- **AS:** Abstract Syntax.
- **API:** Application Programming Interface.
- **Behavioral Semantics:** see *Execution semantics*.
- **C CSL:** Clock-Constraint Specification Language.
- **CS:** Concrete Syntax.
- **Domain Engineer:** user of the Modeling Workbench.
- **DSA:** Domain-Specific Action.
- **DSE:** Domain-Specific Event.
- **DSL:** Domain-Specific Language.
- **DSML:** Domain-Specific (Modeling) Language.
- **Dynamic Semantics:** see *Execution semantics*.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

- **Eclipse Plugin:** an Eclipse plugin is a Java project with associated metadata that can be bundled and deployed as a contribution to an Eclipse-based IDE.
- **ED:** Execution Data (part of DSA).
- **EF:** Execution Function (part of DSA).
- **Execution Semantics:** Defines when and how elements of a language will produce a model behavior.
- **GEL:** GEMOC Event Language which defines the two way protocol between the MoCC and DSA.
- **GEMOC Studio:** Eclipse-based studio integrating both a language workbench and the corresponding modeling workbenches.
- **GUI:** Graphical User Interface.
- **Language Workbench:** a language workbench offers the facilities for designing and implementing modeling languages.
- **Language Designer:** a language designer is the user of the language workbench.
- **MoCC:** Model of Concurrency and Communication.
- **MoCCML:** MoCC Modeling Language.
- **Model:** model which contributes to the content of a View.
- **Modeling Workbench:** a modeling workbench offers all the required facilities for editing and animating domain specific models according to a given modeling language.
- **MSA:** Model-Specific Action.
- **MSE:** Model-Specific Event.
- **Operational semantics:** Constraints on a model that defines its step-by-step execution semantics (see **Execution Semantics**).
- **RTD:** RunTime Data.
- **Static semantics:** Constraints on a model that cannot be expressed in the metamodel. For example, static semantics can be expressed as OCL invariants.
- **TESL:** Tagged Events Specification Language.
- **xDSML:** Executable Domain-Specific Modeling Language.

1.4 Summary

This document describes the current state of the tools provided in the Language Workbench of the GEMOC studio as part of the D1.2.1 deliverable.

The GEMOC language workbench is intended to Language Engineer who wish to design a new Executable Domain-Specific Modeling Language (xDSML).

Building an xDSML consists in defining the various parts of a language (abstract syntax, concrete syntaxes and behavioral semantics), aggregated together thanks to a model of their aggregation (xDSML model) as part of a dedicated project (xDSML project). The combination of these parts provides a concrete tooling supporting the execution of the conforming models. It will also provide a convenient way to deploy the resulting language via a modeling workbench. In this case, the System Engineers will be able to create and execute models conforming to this language.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

The parts required to build an xDSML are: a domain model with its AS, the Domain-Specific Action (DSA) including both the execution function and data, a MoCC, the Domain-Specific Event (DSE), one or several editors (modelers) and optionally one or several graphical animators.

The GEMOC studio offers a workflow that is supported by a dashboard and a suite of wizards for helping the Language Engineers while designing an xDSML.

The workflow explicitly supports various technologies for implementing the required parts of an xDSML.

The concrete technologies explicitly supported by the studio for building the xDSML parts are:

- Domain Model:
 - Ecore + EMF genmodel
- Domain-Specific Actions:
 - Kermeta 3
 - direct modification of ecore and generated code
- MoCC:
 - MoCCML with the MoCCML solver.
 - ECL to define the mapping to the AS
- Editors:
 - EMF tree editor
 - Xtext text editor
 - Sirius viewpoint
- Animators (Backends):
 - Obeo Designer for Simulation

Other technologies can be used by creating them separately and including them manually.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

2. Architecture of the DSML behavioral semantics definition tools

2.1 Implementation of an eXecutable DSML (xDSML)

The implementation of an xDSML consists in implementing the required components of the language so that it can support the execution of the conforming models. Figure 2.1 illustrates the overall process to define the different components, namely the abstract syntax, the concrete syntax (both for edition and animation), and the behavioral semantics (incl., the DSA, the MoCC and the DSE).

Each of these components can be implemented using different tools if they are compatible with the Eclipse platform and EMF. For example, the editors can be tree editors (from EMF), Sirius viewpoints, Xtext editors or combined editors.

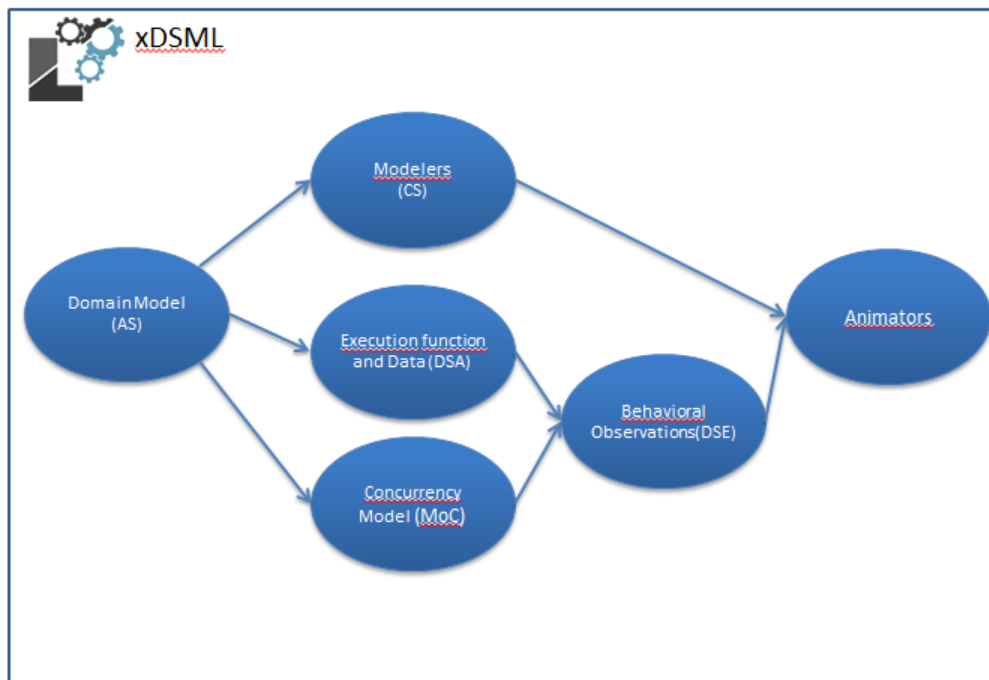


Figure 2.1: xDSML components

2.2 xDSML development method

It is possible to develop an xDSML by directly linking all the components together. However, in order to ease the integration and to tackle the possible variability of the technology used for implementing the different components, the GEMOC Studio offers a development method that provides guidances and drives the language designer.

The method implements a workflow corresponding to the guidelines specified in D1.1.1.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

2.3 xDSML development tool

In the GEMOC Studio, the workflow is equipped with a tooling support that offers GUI interactions for launching the concrete wizards. This tooling strongly relies on a special eclipse project: the xDSML project, that includes in a dedicated model (an xDSML model) an explicit description of the aggregation of the various xDSML components. The xDSML model contains the aggregation definition that will be used by the various tools to update the xDSML project accordingly.

2.3.1 xDSML Project

The role of the xDSML project is twofold:

- to support the interactions with the user while building an xDSML,
- to offer the technical services used by the Modeling Workbench tooling.

2.3.1.1 Typical structure

An xDSML project contains the following elements:

project.xdsml This is the xDSML model containing the xDSML definition of the current language. This is actually a model conforming to the metamodel `gemoc_language_workbench_conf.ecore` (cf. Section 2.3.3)

plugin.xml This file declares the available services which can be used by other components (*e.g.*, components of the GEMOC Modeling Workbench).

xdsml-java-gen This folder is automatically generated by the language workbench from the xDSML model. It implements the technical services.

manifest.mf The dependencies to the other language components are automatically added to the `manifest.mf`.

Other technology-dependent artefacts Depending on the tool used for a given xDSML component, some additional artifact might be created and included into the project (*e.g.*, DSEs written in ECL will produce a QVTo file that will be used by the solver).

2.3.1.2 User interactions

The xDSML project is the support for various user interactions.

- visualization of the current implementation of the xDSML. A view allows to see what are the language components aggregated into the xDSML.
- create a new component. It allows to launch wizards able to build the new components.
- set a new component from an existing one. It allows to launch wizards able to select an existing compatible component and associate it to the xDSML implementation.
- open the language components. It allows to open the project or the file for a language component already aggregated into the xDSML.

These interactions are driven by the process described in chapter 3.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

2.3.1.3 Technical services

The xDSML project offers the following technical services for the Modeling Workbench tooling (e.g., to be used by the Execution Engine).

- A *Model Loading service* which is able to load a model described in (one of) the available syntax(es) (i.e., the one implemented for the xDSML and specified in the xDSML model).
- A *DSA execution service* which is responsible for locating the methods and objects referenced by instances of DSEs in the DSAs (i.e., bytecode).
- A *Solver service* which is responsible for interpreting the solver input (resulting from the compilation of the Domain-Specific Events for a given model) and for returning a scheduling trace that can be interpreted by the Execution Engine.
- A *Feedback Policy service* which specifies how the returning values of Execution Functions should impact the MoCC. For example, the evaluation of the condition of a decision node will impact which branches of the decision node can happen in the future of the execution.

Depending on the tool used for a given language component, some additional services might be associated to the project.

2.3.2 Internal structure of the GEMOC Language Workbench

All the tools required for developing a new xDSML are gathered and deployed in the GEMOC Language Workbench.

It also supports the definition of the xDMSL project, as well as the interactions and automated tasks relying on the xDMSL project. This workbench is based on the following main internal components:

xDSML metamodel (`gemoc.language.workbench.conf.ecore`) This is the metamodel for defining xDSML models. This is an Ecore model specifying the information about a given language. This metamodel is supported by the tool ensuring the aggregation of the various language components.

xDSML view of `project.xml` This is an editor implementing the view on a given xDSML.

Actions supporting the xDSML design process Depending on the state of the xDSML model (i.e., language description), the GUI proposes context-dependent actions to support the user in the language definition. It launches Eclipse commands and Eclipse wizards.

xDSML builder This Eclipse service tracks the changes on the xDSML model. Whenever a change occurs, it automatically updates or generates the relevant artefacts in the xDSML project.

2.3.3 xDSML Metamodel

An xDSML model conforms to the xDSML metamodel. The metamodel is presented in Figure 2.2 It allows to represent resources such as files or eclipse projects in the language workbench in order to aggregate them as an xDSML.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

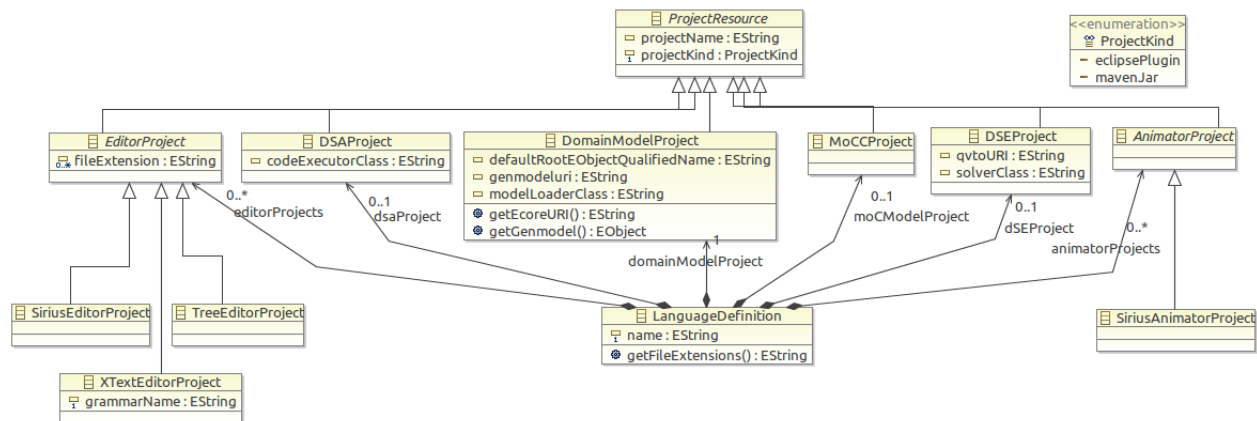






Figure 2.2: xDSML metamodel

3. Gemoc Language Workbench: xDSML development tool workflow

The GEMOC Studio offers several services for defining an xDSML. The underlying tools can be called in a predetermined sequence in order to achieve the various activities that are required to build an xDSML. A specific workflow tool has been implemented to help the user in performing the activities (or steps) of the GEMOC xDSML development method. Figure 3.1 presents the global view of the tools that implements the different activities of the Language Workbench workflow.

In the following sections, each activity will be fully detailed, including the major concrete artifacts resulting from the commands.

Caption:

-  Activity: an activity is a step of the workflow. Each activity is supported by at least one concrete Command
-  Command: a command is a concrete action of the studio, usually implemented as a wizard.
-  Artifact creation: Artifact created as the result of a command.
-  Artifact update: Artifact updated as the result of a command.

3.1 xDSML definition

The *Create language workbench definition* Activity is the global activity of assembling the various components of a xDSML tool suite.

The figure 3.2 presents the activity and its supporting Commands.

3.1.1 New Gemoc Language project Command

The *New Gemoc Language project* Command is a wizard that creates the eclipse project hosting the xDSML definition and the glue code connecting the components for this definition.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

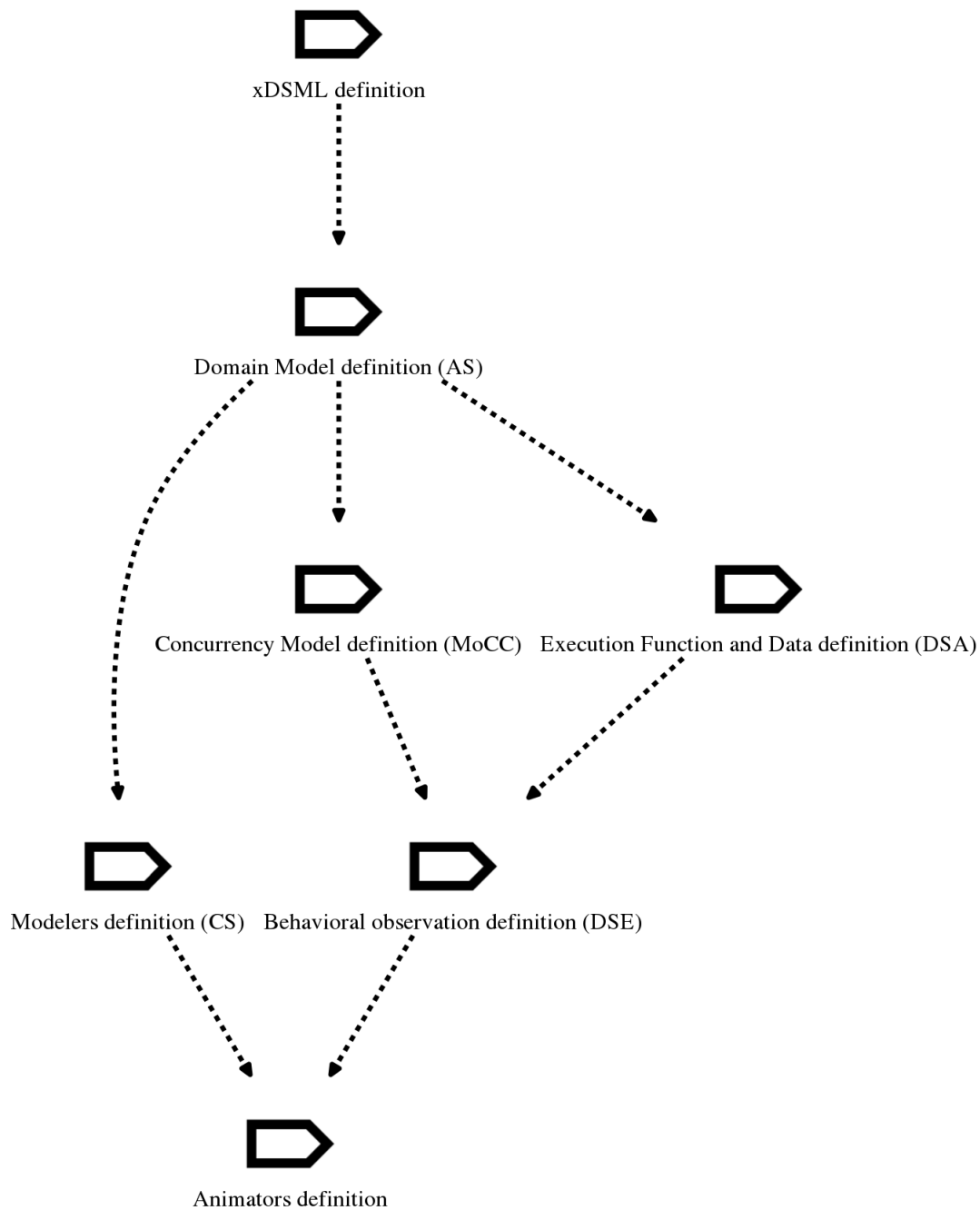


Figure 3.1: Workflow proposed in the GEMOC Language Workbench

3.1.1.1 Created artefacts

Artifacts created by the New Gemoc Language project Command:

Gemoc Language project This is the eclipse project hosting the xDSML definition and the glue code connecting the components for this definition.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

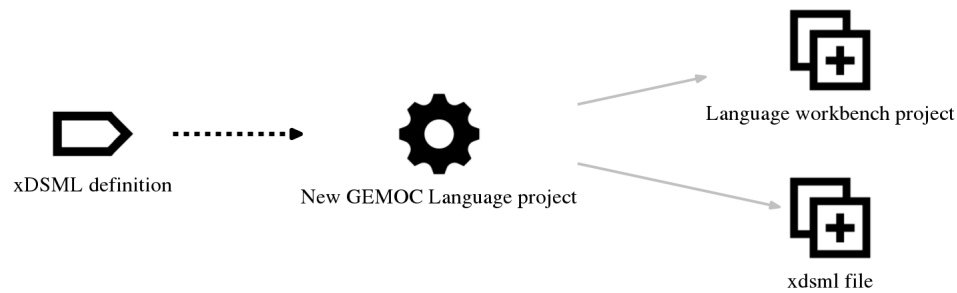


Figure 3.2: xDSML definition activity

xdxml file This is the definition of the language assembly. It acts as a dashboard presenting the implementation choices and provides a link to the concrete implementations components.

3.1.1.2 Updated artefacts

Artifacts updated by the New Gemoc Language project Command:
None

3.2 Domain Model definition (AS)

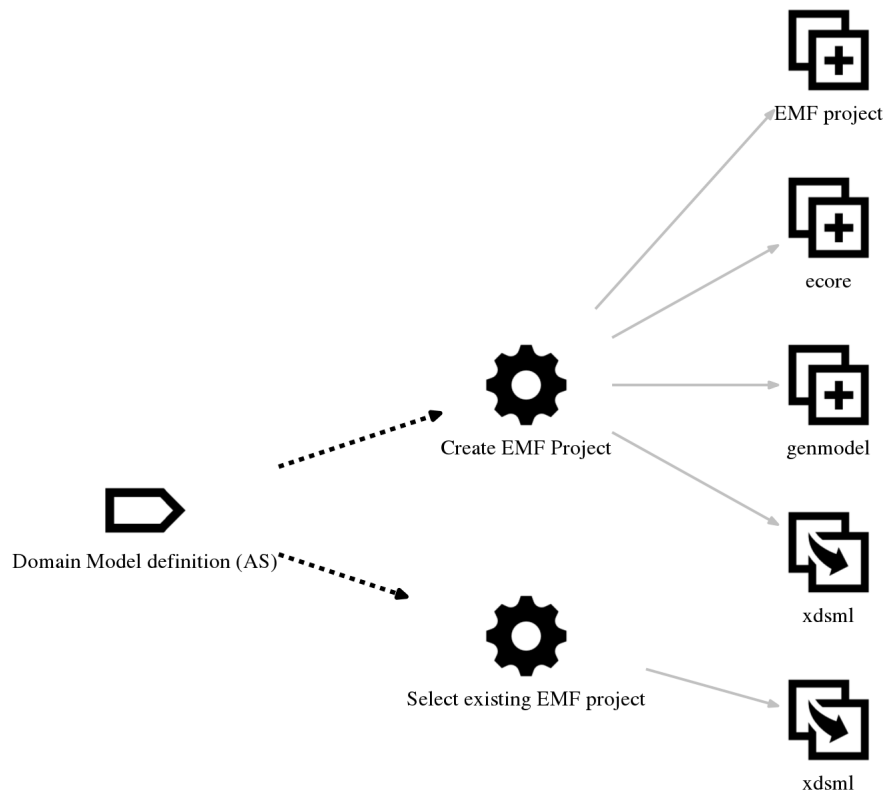


Figure 3.3: Domain Model definition (AS) activity

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

The *Domain Model definition* Activity is the activity of creating the domain model and the components that implement this model.

The figure 3.3 presents the activity and its supporting Commands.

3.2.1 New EMF Project Command

The *New EMF Project* Command is a wizard that creates a new eclipse project hosting the ecore file of the domain definition and its java implementation in an eclipse plugin.

3.2.1.1 Created artefacts

Artefacts created by the New EMF Project Command:

EMF project This artifact is an eclipse plugin project that hosts an ecore file and its java implementation.

ecore This artifact is an ecore model representing the domain of the xDSML.

genmodel This artifact is a genmodel file that is used internally by the eclipse plugin to build the java implementation.

3.2.1.2 Updated artefacts

Artefacts updated by the New EMF Project Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.2.2 Select existing EMF Project Command

The *New EMF Project* Command is a wizard that selects an eclipse plugin project hosting the ecore file of the domain definition and its java implementation.

3.2.2.1 Created artefacts

Artefacts created by the Select existing EMF Project Command:
None

3.2.2.2 Updated artefacts

Artefacts updated by the Select existing EMF Project Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.3 Modelers definition (CS)

This activity will create or associate implementations of concrete syntaxes for the domain Model.

The figure 3.4 presents the activity and its supporting Commands.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

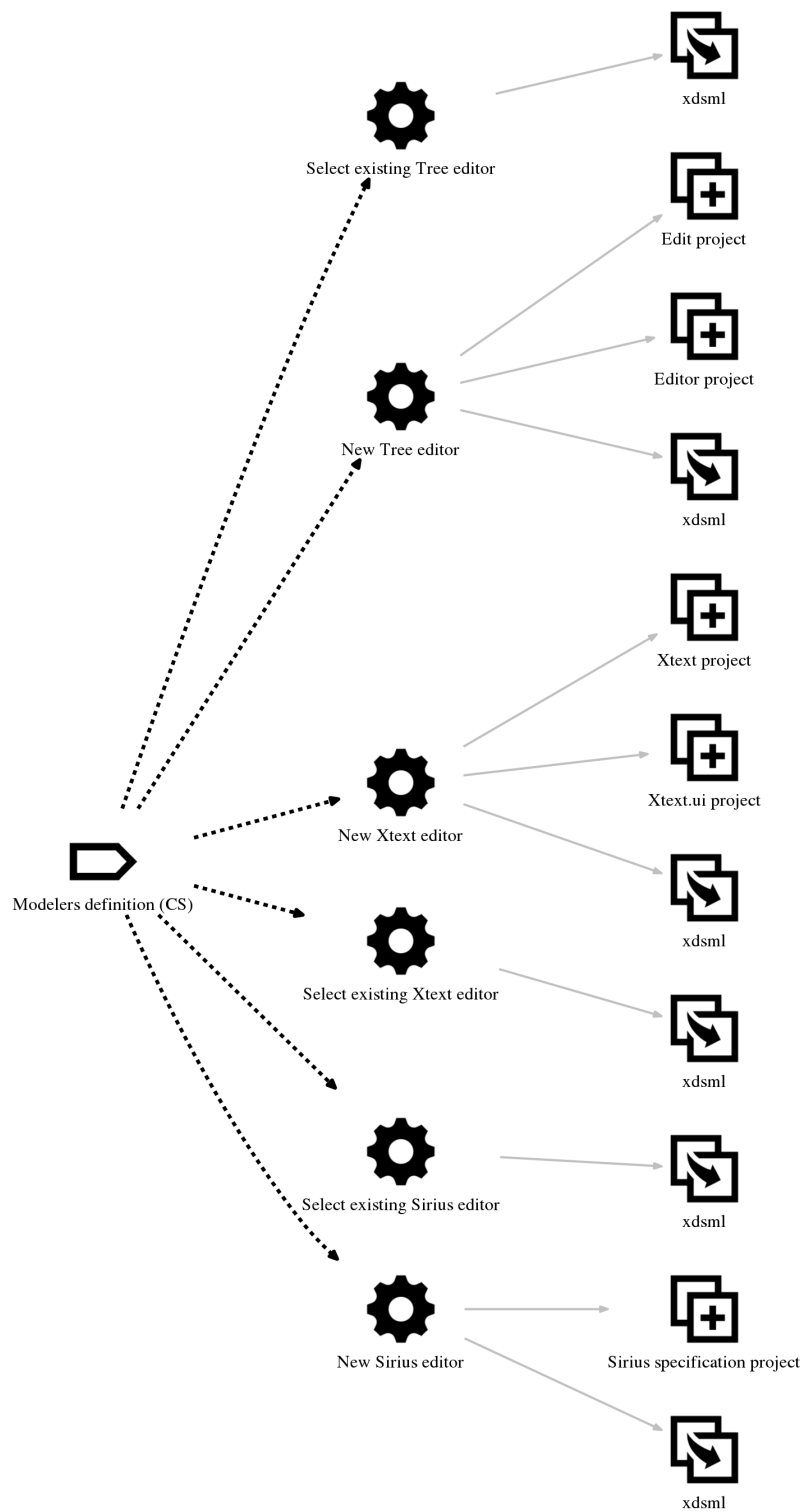


Figure 3.4: Modelers definition (CS) activity

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

3.3.1 *New Tree editor Command*

The *New tree editor* Command is in charge of creating the plugins implementing a tree editor using EMF based on the Domain Model.

3.3.1.1 Created artefacts

Artifacts created by the New Tree editor Command:

Edit project The *Edit project* is an eclipse project that is part of a tree editor.

Editor project The *Editor project* is an eclipse project that is part of a tree editor.

3.3.1.2 Updated artefacts

Artifacts updated by the New Tree editor Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.3.2 *Select existing Tree editor Command*

This command is a wizard that selects existing eclipse plugin projects hosting a tree editor implementation.

3.3.2.1 Created artefacts

Artifacts created by the Select existing Tree editor Command:
None

3.3.2.2 Updated artefacts

Artifacts updated by the Select existing Tree editor Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.3.3 *New Xtext editor Command*

The *Create Xtext editor* Command is in charge of creating the plugins implementing a textual editor using Xtext based on the Domain Model.

3.3.3.1 Created artefacts

Artifacts created by the New Xtext editor Command:

Xtext project The *Xtext project* is an eclipse project that is part of a textual editor editor.

Xtext.ui project The *Xtext.ui project* is an eclipse project that is part of a textual editor editor.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

3.3.3.2 Updated artefacts

Artifacts updated by the New Xtext editor Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.3.4 *Select existing Xtext editor Command*

This Command is a wizard that selects existing eclipse plugin projects hosting an Xtext editor implementation.

3.3.4.1 Created artefacts

Artifacts created by the Select existing Xtext editor Command:
None

3.3.4.2 Updated artefacts

Artifacts updated by the Select existing Xtext editor Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.3.5 *New Sirius editor Command*

The *New Sirius editor* Command is in charge of creating the plugins implementing a graphical editor using Sirius framework based on the Domain Model.

3.3.5.1 Created artefacts

Artifacts created by the New Sirius editor Command:

Sirius specification project The *Sirius specification project* is an eclipse project that implements a Sirius editor.

3.3.5.2 Updated artefacts

Artifacts updated by the New Sirius editor Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.3.6 *Select existing Sirius editor Command*

This Command is a wizard that selects existing eclipse plugin projects hosting a Sirius editor implementation.

3.3.6.1 Created artefacts

Artifacts created by the Select existing Sirius editor Command:
None

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

3.3.6.2 Updated artefacts

Artifacts updated by the Select existing Sirius editor Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.4 Domain-Specific Actions

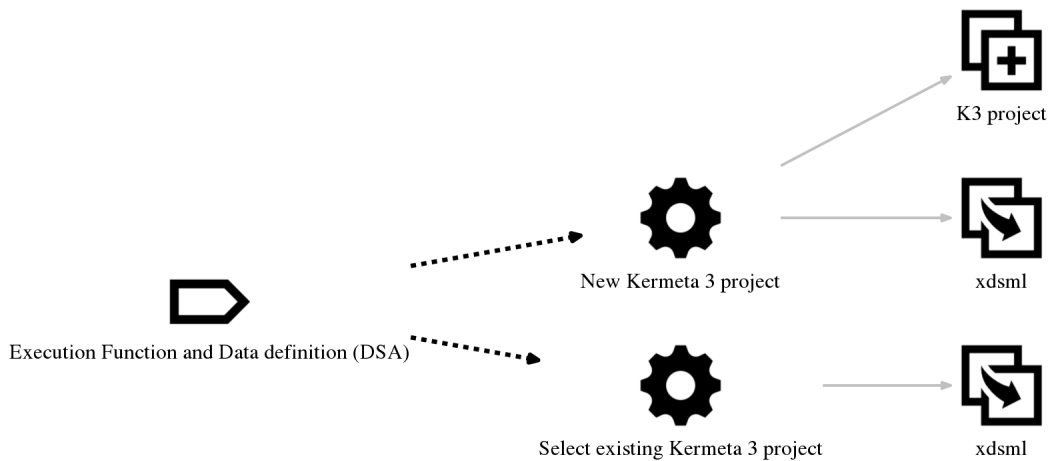


Figure 3.5: RTD and DSAs definition activity

This is the activity of creating the components that implement the Execution Data and Functions of the xDSML. These components are based on the Abstract Syntax, as explained in Deliverable 1.1.1.

The figure 3.5 presents the activity and its supporting Commands.

3.4.1 New Kermeta 3 project Command

This command is a wizard that is in charge of selecting an existing plugin implementing the DSA using Kermeta 3.

3.4.1.1 Created artefacts

Artifacts created by the New Kermeta 3 project Command:

K3 project This is an eclipse project written in Kermeta 3 that implements the DSA for the xDSML.

3.4.1.2 Updated artefacts

Artifacts updated by the New Kermeta 3 project Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

3.4.2 Select existing Kermeta 3 project Command

This command is a wizard that is in charge of selecting an existing plugin implementing the DSA using Kermeta 3.

3.4.2.1 Created artefacts

Artifacts created by the Select existing Kermeta 3 project Command:
None

3.4.2.2 Updated artefacts

Artifacts updated by the Select existing Kermeta 3 project Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the selected artifacts so the glue can be automatically adapted to use them.

3.5 Model of Concurrency and Communication (MoCC) definition: reusable part

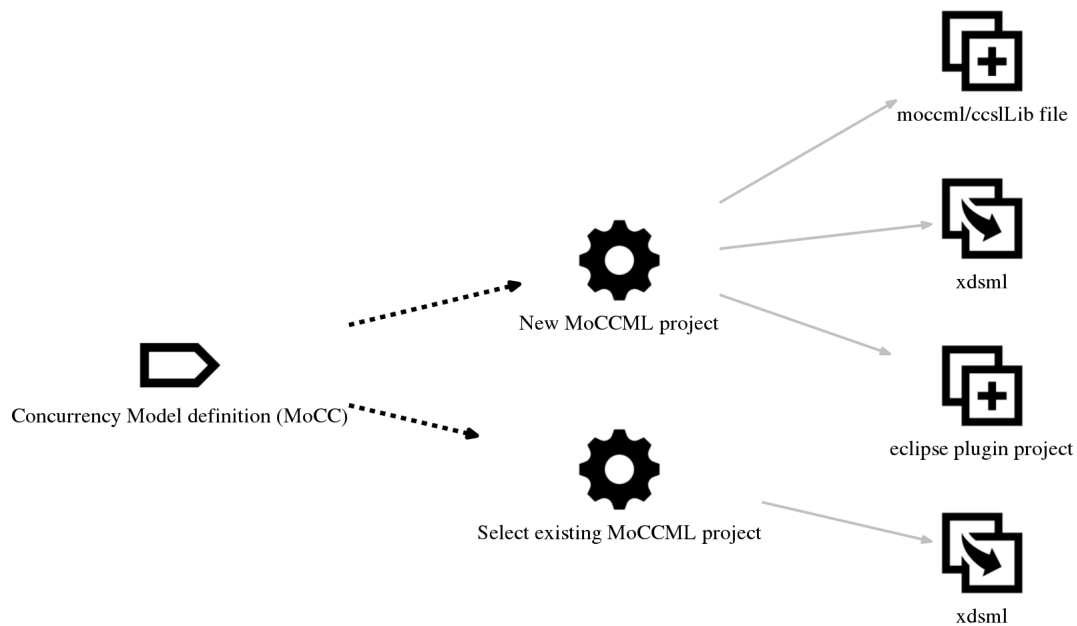


Figure 3.6: Model of Concurrency and Communication definition activity

This activity is in charge of creating the components that implement the model of computation of the Domain Model for the xDSML.

The figure 3.6 presents the activity and its supporting Commands.

3.5.1 New MoCCML project Command

The current implementation supposes that the DSE written in ECL have an import to the MoCC definition. This MoCC definition is supposed to be moccml and ccsLib files that are bundled in an eclipse plugin.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

3.5.1.1 Created artefacts

Artifacts created by the New MOCCML project Command:

eclipse plugin project This is an eclipse plugin project that hosts and deploys the moccml and ccsLib files.

moccml/ccsLib file This is the moccml/ccsLib file implementing the MoCC in MoCCML.

3.5.1.2 Updated artefacts

Artifacts updated by the New MoCCML project Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.5.2 *Select existing MoCCML project Command*

This command is a wizard that is in charge of selecting an existing plugin implementing the MoCC using MoCCML.

3.5.2.1 Created artefacts

Artifacts created by the Select existing Kermeta 3 project Command:

None

3.5.2.2 Updated artefacts

Artifacts updated by the Select existing MoCCML project Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the selected artifacts so the glue can be automatically adapted to use them.

3.6 **Model of Concurrency and Communication (MoCC) definition: mapping to the AS**

This is the activity of creating the components that implement the Domain Specific Event for the xDSML.

The figure 3.7 presents the activity and its supporting Commands.

3.6.1 *New ECL file Command*

The command is in charge of creating the plugin hosting the DSEs. This plugin is written using ECL (Event-Constraint Language).

3.6.1.1 Created artefacts

Artifacts created by the New ECL file Command:

ecl file The *ecl file* is the implementation of the DSEs written in ECL.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

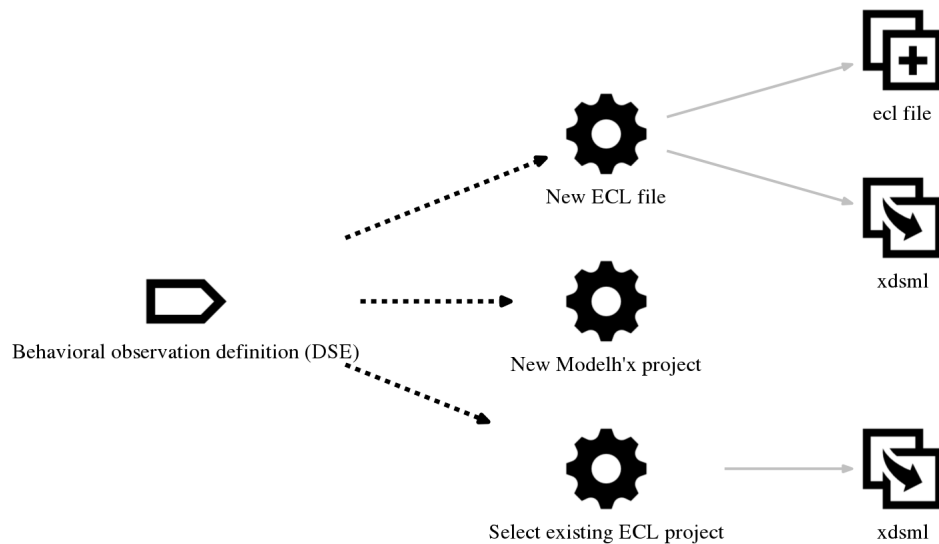


Figure 3.7: MoCC mapping to the AS activity

3.6.1.2 Updated artefacts

Artifacts updated by the New ECL file Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the created artifacts so the glue can be automatically adapted to use them.

3.6.2 New Modelh'x project Command

Not implemented in this version of the Studio.

3.6.2.1 Created artefacts

Artifacts created by the New Modelh'x project Command:

None

3.6.2.2 Updated artefacts

Artifacts updated by the New Modelh'x project Command:

None

3.6.3 Select existing ECL file Command

This command is a wizard that is in charge of selecting an existing plugin implementing the DSEs using ECL.

3.6.3.1 Created artefacts

Artifacts created by the Select existing ECL file Command:

None

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

3.6.3.2 Updated artefacts

Artifacts updated by the Select existing ECL file Command:

xdsml The command also updates the xDSML artifact in order to indicate the various locations of the ECL file so the glue can automatically create a qvto transformation from it and deploy it.

3.7 Animators definition

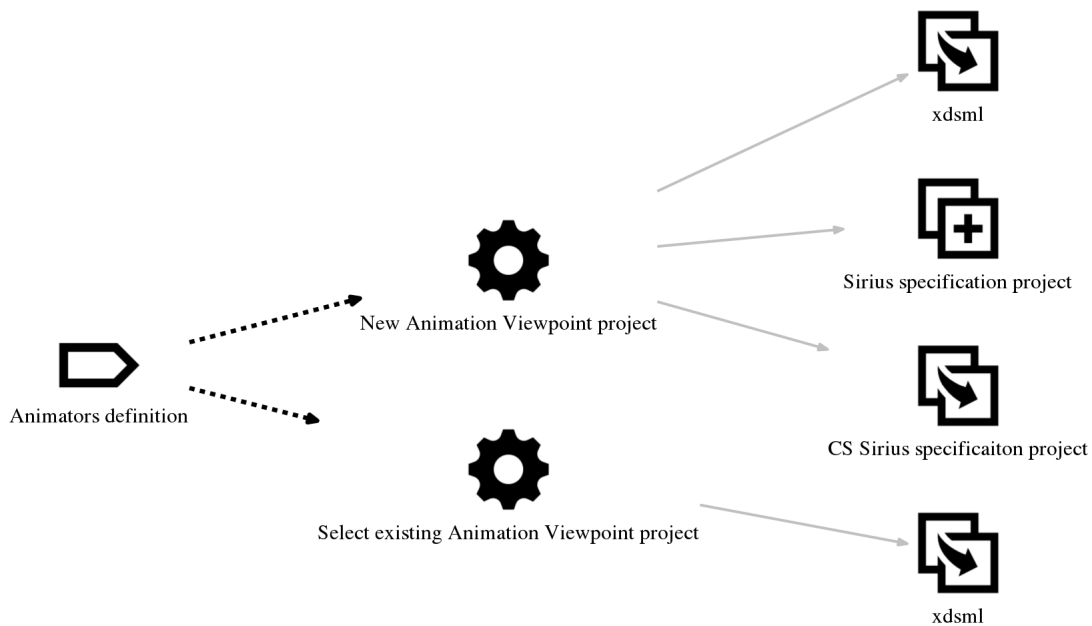


Figure 3.8: Animators definition activity

The figure 3.8 presents the activity and its supporting Commands.

In the xDSML editor we can launch a wizard to create the Viewpoint specification — either as a new project or as an extension of the project used for the graphical concrete syntax — or select an existing one.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

4. Language workbench dashboard

This chapter presents the GEMOC Language Workbench dashboard. It aims at providing the language designer with a view on the steps to be performed. When a step is selected the already created artifacts are shown as well as the concrete action which can be performed. The main elements of the dashboard and the way to use it are presented hereafter.

The dashboard does not enforce any process, it only presents the steps of the process and the related actions as well as the already defined artifacts. The language designer keeps full control of the way to define his executable DSML.

4.1 Open the dashboard

The dashboard is available on .xdsml files from the Project Explorer or the Model Explorer views (fig 4.1).

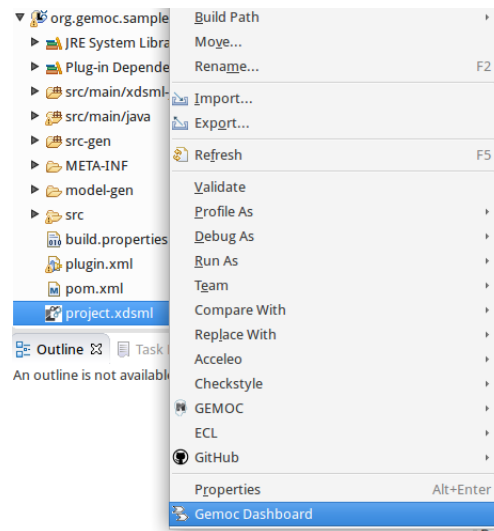


Figure 4.1: Dashboard menu

When the “Gemoc Dashboard” menu is selected a new editor opens (fig. 4.2). This editor is divided in three main parts:

1. Overview part
2. Sections part
3. Projects view

4.2 Overview

The overview allows to access to the different parts of the GEMOC Language Workbench process:

- Create your language
- Make your language executable

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

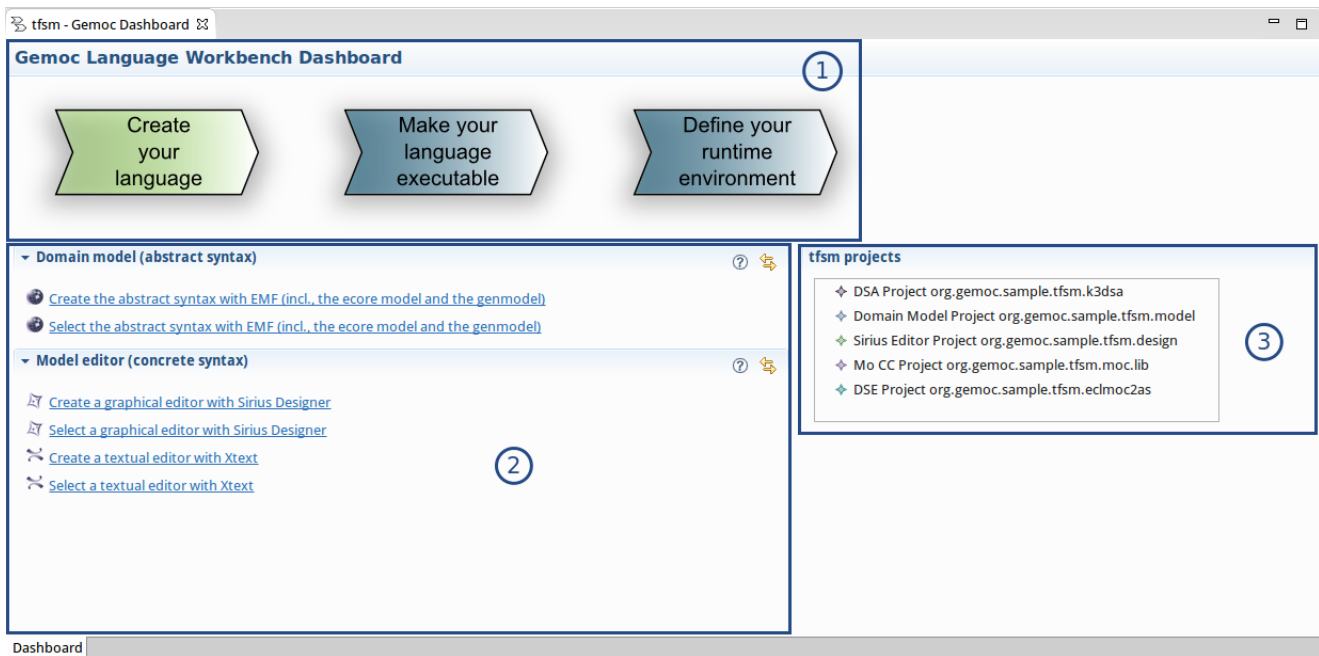


Figure 4.2: Dashboard editor

- Define your runtime environment

When you click on a step in the overview the related actions are automatically presented in the sections part.

4.3 Sections

This part provides sections which depend on the selected overview. The sections contain links that can be used to activate the different actions : create the domain model project, edit the DSA project, define a concrete syntax...

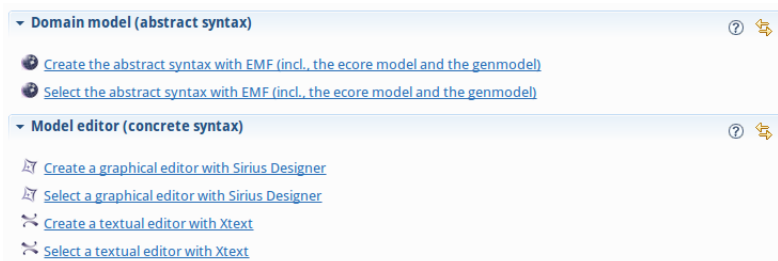


Figure 4.3: Sections description

4.4 Description

Each section provides a help button which shows the description associated to the section and explains the process step and which actions are available.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

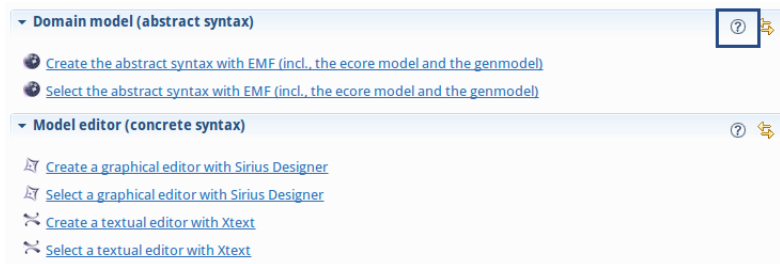


Figure 4.4: Dashboard description

4.5 Links

When a link is selected, the corresponding action is launched : open a wizard, open the documentation...

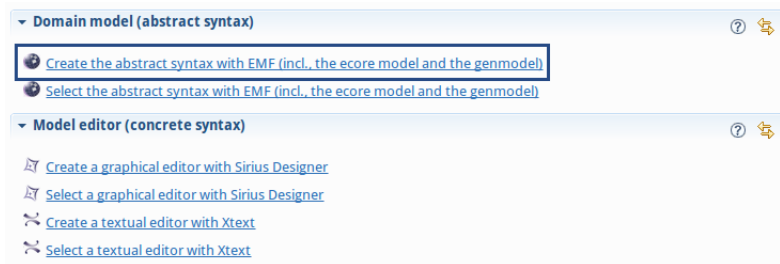


Figure 4.5: Link

4.6 Projects view

On the right side of the dashboard, a tree viewer shows the projects associated to the current xDSML project.

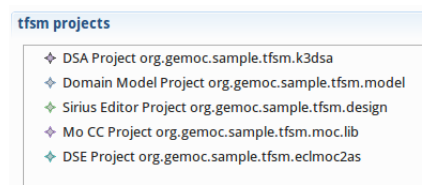


Figure 4.6: Projects viewer

4.7 Synchronize

The section toolbar provides a 'synchronize' action (fig. 4.7). This action filters the content of the tree viewer to keep only projects relevant according to the current process step represented by the section.

4.8 Open existing projects

By double-clicking on a project in the tree viewer (fig. 4.8), it is possible to open and update the corresponding file.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

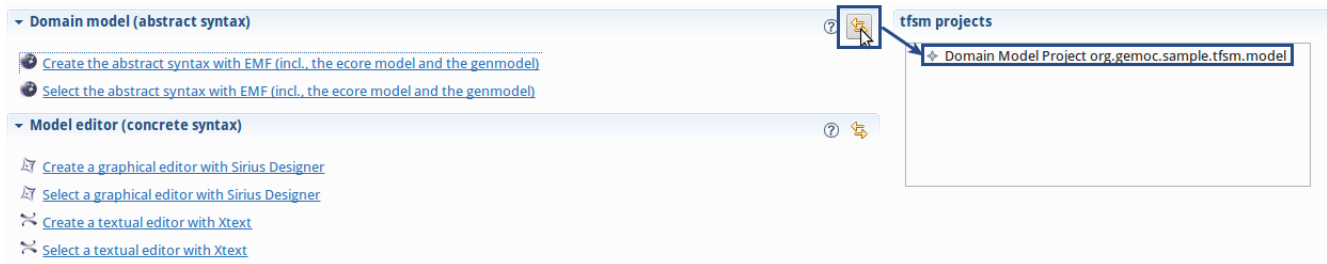


Figure 4.7: Synchronize projects

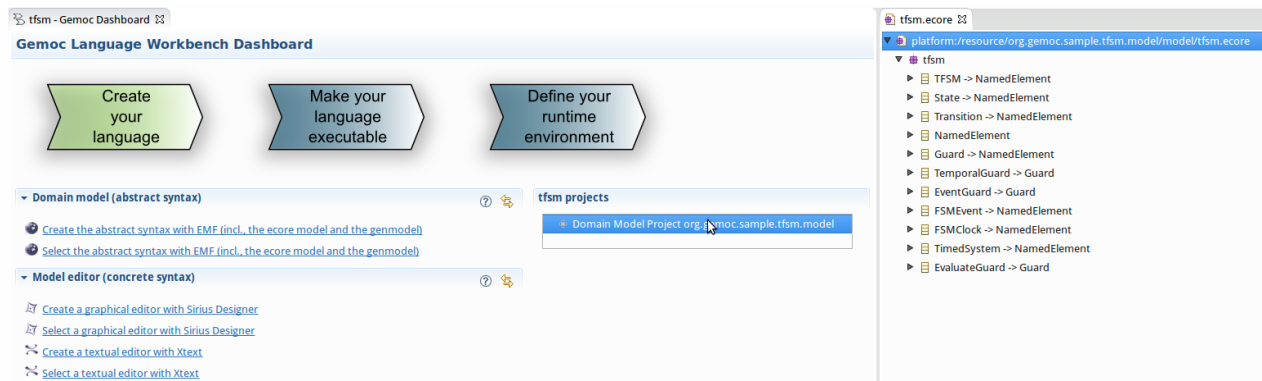


Figure 4.8: Open and update files from the selected project

5. Current implementation

The GEMOC Language Workbench is currently composed of the following eclipse plugins and development projects:

org.gemoc.gemoc_language_workbench.api Eclipse plugin exposing the interfaces that a xDSML project must offer.

org.gemoc.gemoc_language_workbench.conf.model EMF Eclipse plugin defining the xdsml.ecore and its implementation classes.

org.gemoc.gemoc_language_workbench.conf.model.edit Simple tree editor support for xdsml models.

org.gemoc.gemoc_language_workbench.conf.model.editor an editor with support for launching the wizards.

org.gemoc.gemoc_language_workbench.documentation Eclipse help associated to the Gemoc Language workbench.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

org.gemoc.gemoc_language_workbench.extensions.k3 Eclipse plugin offering an implementation of the execution service for DSA built with Kermeta 3.

org.gemoc.gemoc_language_workbench.feature Eclipse feature used to deploy the language workbench plugins.

org.gemoc.gemoc_language_workbench.p2updatesite Description of the update site deploying all these features and plugins. (used by continuous integration).

org.gemoc.gemoc_language_workbench.root maven root description used by continuous integration to build all the other parts.

org.gemoc.gemoc_language_workbench.sample.deployer Eclipse plugin deploying the examples illustrating the Language workbench. It currently deploys the tfsm example.

org.gemoc.gemoc_language_workbench.ui Main eclipse plugin for the language workbench. It implements the workflow and its commands. It also implements the builder that updates the xDSML project according to the definition.

org.gemoc.gemoc_language_workbench.utils Eclipse plugin containing various reusable code.

org.gemoc....extensions.sirius Sirius is an Eclipse project which allows to easily creating graphical modelling workbenches easily by leveraging the Eclipse Modeling technologies, It is built on top of EMF and GMF.

fr.inria.aoste.timesquare.* TimeSquare is an MDK (Model Development Kit) provided as a set of Eclipse plugins that can be downloaded or installed over an existing Eclipse. TimeSquare provides an engine, a solver and editors for CCSL specifications (Clock Constraint Specification Language).

org.gemoc.gemoc_language_workbench.dashboard Eclipse plugin for the GEMOC Language Workbench Dashboard.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

6. Tutorials

The workflow presented in section 3 is a technical workflow that explains the concrete actions which have to be performed in the GEMOC Language Workbench in order to define an xDSML. The dashboard describes the main steps and actions for defining an xDSML but does not enforce any process. Thus, it lacks more abstraction on why and in which order to build the different artefacts which are parts of the definition of an xDSML. The deliverable D1.1.1 describes a possible high level process to guide the Language Engineer in the definition of an xDSML. The starting point of this process is the identification of the Language Engineer's expectations and then describes a possible way to build the artefacts and validate them against the initial expectations.

The main advantage of this process is to present a way to build an xDSML which outlines the purpose of each step from the Language Engineer viewpoint and not from the technical viewpoint (how the elementary steps are achieved in the GEMOC Language Workbench is not described). Unfortunately, this process cannot be implemented as this in the Language Workbench because :

1. It is only a possible way to perform the required steps. We believe that this process must not be enforced but only be used as guidelines by the Language Engineer.
2. It implies dealing with artifacts which are not part of GEMOC xDSML's artefacts. For example expectations of Language Engineer are not formalized in GEMOC. Furthermore, this process implies to fill in only parts of the proposed artefacts. For examples, it advocates to identify DSE before defining their mappings to the MoCC on one side and the DSA on the other side.
3. It is quite a complex process with a great number of tasks which cannot be easily presented to the Language Engineer in a convenient way for him.

In consequence, we believe that a better choice than implementing this process is to include in the GEMOC Language Workbench :

1. A tutorial to explain how to use the different tools to build an xDSML while focusing on the methodological aspects and not on the technical action to perform (this technical actions will obviously be explained).
2. A description of the process defined in the deliverable D1.1.1 as part of Eclipse help in the GEMOC Language Workbench.

6.1 Tutorial Description

The tutorial is in fact composed of several tutorials.

A first set of tutorials is composed of small and focused tutorials which describe how to use the GEMOC Studio to make an executable Domain-Specific (Modeling) Language (DSML), the starting point being its abstract and concrete syntaxes. At the moment short tutorials are defined for Marked Graph and SIGPML DSML. The structure of this two tutorials is the following:

1. Overview
2. Install the projects which defines the abstract and concrete syntaxes
3. Set up an xDSML project
4. Define the execution semantics
5. Animate a model

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

Another tutorial is defined with the purpose to explain the main steps to build an xDSML with a focus on software engineering techniques like unit testing and illustrating the GEMOC process described in D1.1.1. It gradually presents main concepts of the GEMOC Language Workbench. Here is the description of the main parts of this tutorial.

1. Description of the considered DSML
2. Recall of the GEMOC Process
3. Definition of the requirements on the execution semantics of the xDSML

The first step consists in describing the system engineers' expectations in terms of execution semantics and visualization at runtime. As it is often a complex task for system engineers to formally describe their expectations, we advocate to define them through examples.

An example is composed of:

- (a) A **model** which is conform to the DSML AS.
- (b) A **scenario** which describes a particular use of the model. A scenario is composed of events, i.e. stimuli that trigger evolution of the model.
- (c) **Expected results** while the scenario is played. Expected results include values of runtime data, possible next events, etc.

4. Definition of DSA and MoCC

The purpose of this first increment is to explain how to define the MoCC and the DSA as well as the DSE.

5. Graphical visualization

This part shows how to build a graphical visualization of the model at runtime.

6. Illustration of the feedback mechanism

The purpose is to illustrate the feedback mechanism: information from the DSA are used to select among the possible futures proposed by the MoCC the ones that are compatible with the real execution of the model.

7. Call of User actions

This part explains how to connect an xDSML to an outside system.

This tutorial will be updated to follow the advances made in the GEMOC project.

6.2 Process Description

The process has been defined in the deliverable D1.1.1. It will be described as an Eclipse help so that the Language Designer can have access to it, anytime during the design process.

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

7. Conclusion

The software deliverable D1.2.1 is a set of tools supporting the process to define an xDSML. It is completed with Eclipse help which includes a tutorial to demonstrate the use of the GEMOC Language Workbench and the description of the methodological process described in D1.1.1.

The various steps of the process for developing an xDSML are summarized both in the xDSML project editor and in a dashboard. Both tools present as a specific view of the xDSML project that gives direct access to the main artefacts of an xDSML project and corresponding editors and wizards.

The tools and the wizards supporting the process are integrated and deployed into the GEMOC Studio. A tutorial has been proposed to illustrate the process and the use of the GEMOC Studio.

The GEMOC Studio can be found on the download page of the GEMOC project: <http://gemoc.org/studio-download/>. The source code is available on the git server: `git+ssh://developer-name@scm.gforge.inria.fr//gitroot/gemoc-dev/gemoc-dev.git`. The version described in this deliverable is the v1.0.

The tutorials are available at <http://gemoc.github.io/gemoc-studio/>. The sources of the tutorials are available at <https://github.com/gemoc/gemoc-studio/wiki/>. The Eclipse projects of the tutorial are available on git server in `git+ssh://developer-name@scm.gforge.inria.fr//gitroot/gemoc-dev/org/gemoc/sample/`.

ANR INS GEMOC / Task 1.2.1	Version:	2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date:	May 30, 2015
D1.2.1		

A. Getting started with the Language Workbench

This annex presents some of the major steps to get started with the current GUI of the workflow tooling. It will use one of the example that is bundled in the GEMOC Studio (TFSM). It shows how to install the example. It also shows some of the basic GUI commands used to create or navigate an xDSML by aggregating components with an xDSML project and model.

Understanding the TFSM example itself and how its aggregated components are designed internally is not in the scope of this document. To understand the design of the xDSML TFSM and play with it, we refer the reader to <http://gemoc.org/sle13/> (including videos).

- Download the GEMOC Studio from <http://gemoc.org/studio-download/>.
- Install the TFSM example definition. This is done by navigating in eclipse menu: File > new > Example... > GEMOC TFSM Language example (see figureA.1).
- In this example you can open the project.xdxml file in order to have a dashboard that synthetizes the current components used by the xDSML definition. As shown on figure A.2, in the current version, the dashboard is presented as a view on the different components of the xDSML. The component type (Eclipse project or resource) is displayed with a clickable underlined red text. On its right is displayed the location of the component — if it has been initialized. A browse button allows to select an existing component.

Clicking on the component type has an effect which depends on whether the component type has been initialized or not. When initialized, it opens the corresponding resource. If several resources of the expected type are available, a popup allow the user to select one of them. If the resource has not been initialized yet, clicking on the component type launches the wizard to create it.

- The workflow actions are accessible by clicking on the component type (underlined red terms on figure A.2) from the xDSML view of the project.xdxml file. These actions are also in the menu labeled *Gemoc Language*, (see figure A.3.)
- Using the wizard via the actions will automatically update the project.t fsm accordingly. The project.xdxml file will in turn be used in the background in order to update the org.gemoc.sample.t fsm project so it can act as the glue between the elements that are part of the xDSML (update of the dependencies, creation of factories, ...).

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

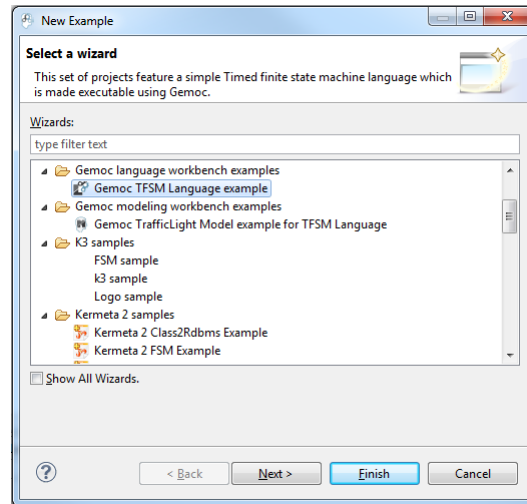


Figure A.1: install TFSM language example screenshot

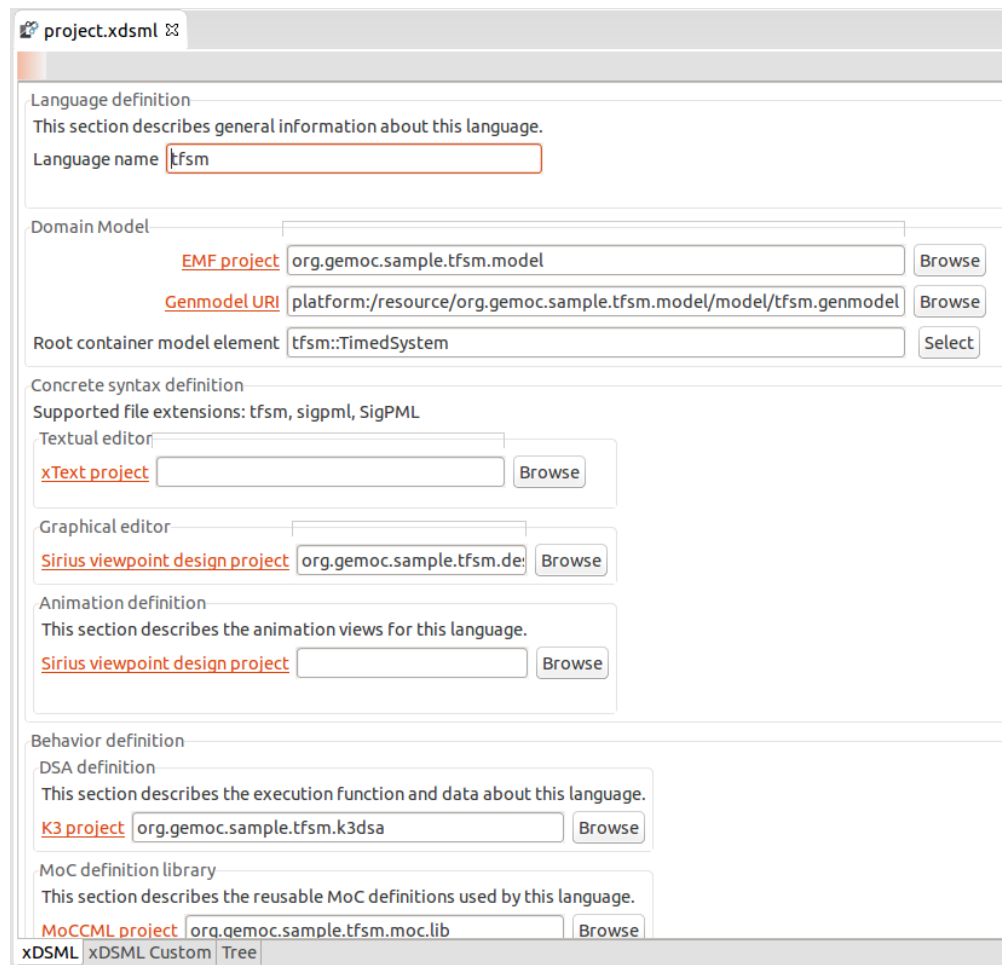


Figure A.2: xDSML editor

ANR INS GEMOC / Task 1.2.1	Version: 2.0
DSML behavioral semantics definition tools (SOFTWARE)	Date: May 30, 2015
D1.2.1	

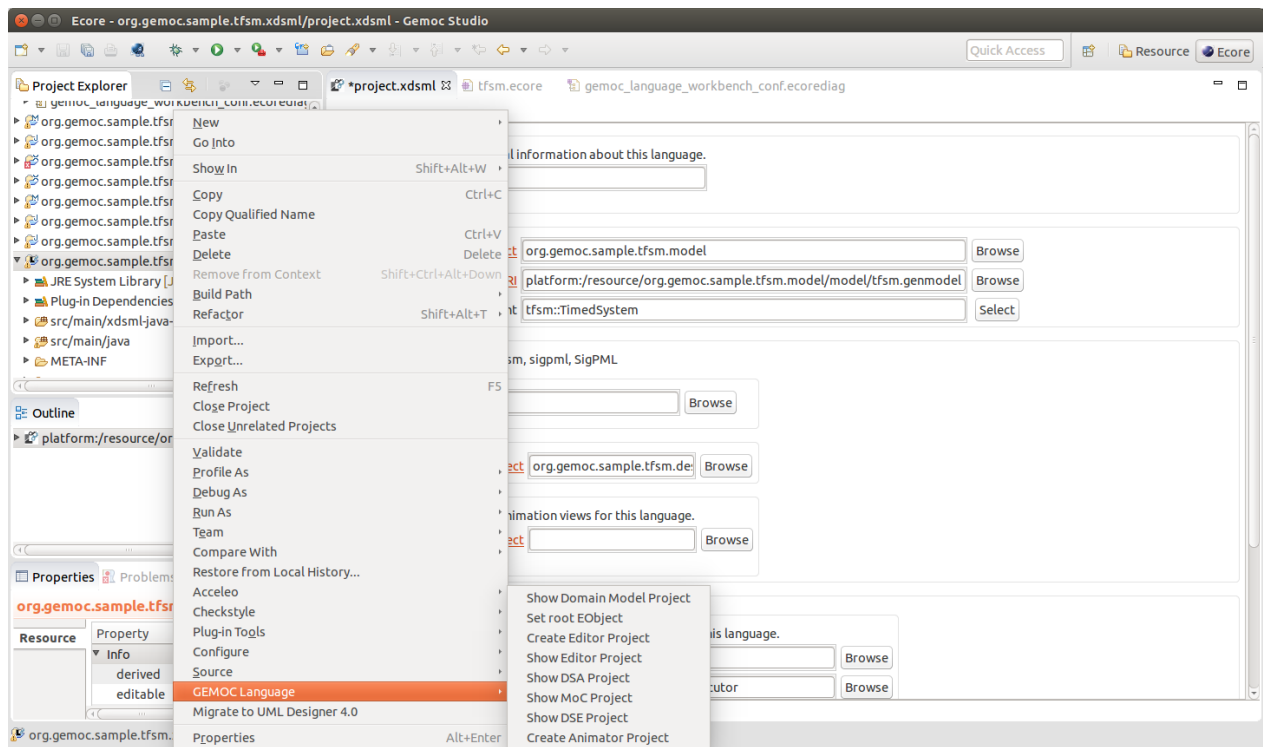


Figure A.3: GEMOC Language menu