

GEMOC Studio Guide

**Francois Tanguy <francois.tanguy@inria.fr>,
Didier Vojtisek <didier.vojtisek@inria.fr>**

GEMOC Studio Guide

Francois Tanguy <francois.tanguy@inria.fr>, Didier Vojtisek <didier.vojtisek@inria.fr>

Table of Contents

Introduction	v
1. Gemoc Language workbench	1
1.1. Language Workbench overview	1
1.2. Gemoc Language project	1
1.3. Defining Abstract Syntax (or domain concepts)	1
1.4. Defining RunTime Data	2
1.5. Defining Domain-Specific Actions (DSAs)	2
1.6. Defining Domain-Specific Constraints	2
1.7. Defining a concrete syntax	2
1.7.1. Defining a concrete syntax with xText	2
1.7.2. Defining a concrete syntax with Sirius	2
1.8. Defining Model of Concurrency (MoC)	2
1.9. Defining Domain Specific Events (DSE)	2
1.10. Defining an animation view	2
1.11. Process support view	2
2. Gemoc Modeling workbench	3
2.1. Modeling workbench overview	3
2.2. Editing model	3
2.3. Executing model	3
2.3.1. Launch configuration	3
2.3.2. Engine view	3
2.3.3. Logical step view	3
2.3.4. Event manager view	3
2.3.5. Animation view	3
Bibliography	4
Glossary	5
Index	7

List of Figures

1.1. Screenshot of GEMOC Studio Language Workbench on the TFSM (Timed Finite State Machine) example. 1

2.1. Screenshot of GEMOC Studio Modeling Workbench on the TFSM example (execution and animation). 3

Introduction

The GEMOC Studio offers 2 main usages:

- Building and composing new executable DSML. This mode is intended to be used by language designers (aka domain experts).
- Creating and executing models conformant to executable DSMLs. This mode is intended to be used by domain designers.

Each of these usage has it own set of tools that are referenced as **Gemoc Language workbench** for the tools for *language designers* and **Gemoc Modeling Workbench** for the tools for *domain designers*.

Chapter 1. Gemoc Language workbench

1.1. Language Workbench overview

The GEMOC Language Workbench, intended to be used by language designers: it allows building and composing new executable DSMLs.

Figure 1.1. Screenshot of GEMOC Studio Language Workbench on the TFMS (Timed Finite State Machine) example.

1.2. Gemoc Language project

In the menu go to: File> New> Projects...> Gemoc Language project

This project will be the aggregator of the other Eclipse projects that will constitute the components of a Language Unit of the xDSML.

It mainly defines a project.xdxml file that references these other components.

The xDSML file isn't supposed to be edited manually. The pop up menu on the project (available on right click) or the Section 1.11, "Process support view" provides a set of wizard to fill this file.

From these information, the project will generate additional code that will be used by the execution engine when the language will be deployed and used by the Chapter 2, *Gemoc Modeling workbench*.

1.3. Defining Abstract Syntax (or domain concepts)

Defining the abstract syntax (AS) in the Language Workbench is done thanks to an EMF Ecore project.

All ecore editors can be used to edit the concepts in the ecore file.

TODO new project wizard (via popup, via process) TODO select existing project TODO Ecore editor

1.4. Defining RunTime Data

1.5. Defining Domain-Specific Actions (DSAs)

1.6. Defining Domain-Specific Constraints

1.7. Defining a concrete syntax

The xDSML can support different concrete syntaxes. Most EMF based editors should work however Gemoc provides additional support for some editors. Editors explicitly supported are: EMF tree editor, xText editor, Sirius editor.

1.7.1. Defining a concrete syntax with xText

1.7.2. Defining a concrete syntax with Sirius

1.8. Defining Model of Concurrency (MoC)

1.9. Defining Domain Specific Events (DSE)

1.10. Defining an animation view

The animation layer is an extension on top of a graphical editor defined with Sirius.

TODO Debug layer, Animation layer

1.11. Process support view

TODO present process view

Chapter 2. Gemoc Modeling workbench

2.1. Modeling workbench overview

The GEMOC Modeling Workbench, intended to be used by domain designers: it allows creating and executing models conformant to executable DSMLs.

Figure 2.1. Screenshot of GEMOC Studio Modeling Workbench on the TFSM example (execution and animation).

2.2. Editing model

2.3. Executing model

2.3.1. Launch configuration

Description of the Gemoc launch configuration in Run and Debug mode.

2.3.2. Engine view

2.3.3. Logical step view

2.3.4. Event manager view

2.3.5. Animation view

Bibliography

The bibliography lists some useful external documents. For a more complete list, please refer to the publications section on <http://gemoc.org> site.

Articles

[globalizing-modeling-languages] Globalizing Modeling Languages [<http://hal.inria.fr/hal-00994551>] (Benoit Combemale, Julien Deantoni, Benoit Baudry, Robert France, Jean-Marc Jezequel, Jeff Gray), In Computer, IEEE, 2014.

Glossary

AS

Abstract Syntax.

API

Application Programming Interface.

Behavioral Semantics

see Execution semantics.

CCSL

Clock-Constraint Specification Language.

Domain Engineer

user of the Modeling Workbench.

DSA

Domain-Specific Action.

DSE

Domain-Specific Event.

DSML

Domain-Specific (Modeling) Language.

Dynamic Semantics

see Execution semantics.

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.

Execution Semantics

Defines when and how elements of a language will produce a model behavior.

GEMOC Studio

Eclipse-based studio integrating both a language workbench and the corresponding modeling workbenches

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

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.

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.

xDSML

Executable Domain-Specific Modeling Language.

Index

L

language designer, 1
Language workbench, 1

M

Modeling workbench, 3

S

Sirius, 2, 2

T

TFSM, 1, 3
 Language workbench, 1
 Modeling workbench, 3