Grant ANR-12-INSE-0011

GEMOC

ANR Project, Program INS

WP5 – GEMOC EXPERIMENTATIONS

D5.2.1 DSML and MoC for Use Cases (SOFTWARE)
Task 5.2

Version 1.0

DOCUMENT CONTROL

	- :	A :	B :	C :	D :
Written by Signature	Marc Pantel (IRIT)				
Approved by Signature					

Revision index	Modifications
A	
B	
C	
D	

Authors

Author	Partner	Role
Marc Pantel	IRIT	Lead author
Xavier Crégut	IRIT	Contributor
Ali Koudri	TRT	Contributor
Jérôme Le Noir	TRT	Contributor
Benoît Combemale	INRIA	Contributor
Julien De Antoni	I3S	Reviewer
Joël Champeau	ENSTA-B	Reviewer

Table of content

D5.2.1 DSML and MoC for Use Cases

1. Introduction

The GEMOC project targets a language design studio providing methods and tools to ease the design and integration of new MoCCs and executable DSMLs (xDSMLs) relying on the use of proven technologies developed in previous research projects such as Cometa, MOCML, Kermeta and the metamodeling pattern to build xDSML in order to define:

- Modeling languages with associated methods and tools for the modeling of both MoCCs and xDSMLs;
- A single cooperative heterogeneous execution framework parameterized by the MoCCs and xDSMLs definitions;
- A global MoCCs and xDSMLs design methodology encompassing these two items.

A formal specification of the previous cooperation framework to prove its completeness, consistency and correctness with respect to the cooperative heterogeneous model execution needed.

1.1 Purpose

This document describes the experiments conducted with the language workbench from the toolset developed in GEMOC for the three use cases in WP5. There will be several versions of the document providing slightly different contents. The first version targets the early development conducted with the first version of the toolset. It provides the components of the xDSML needed to build the use cases: AS, DSA, MoCC, DSE. The second version will bring the whole set of components developed with the toolset during the project.

The content is different between the use cases developed by the Academic partners INRIA and IRIT (in cooperation with Airbus); and the one developed by the industrial partner THALES. INRIA and IRIT are relying on common public languages and thus provide software and a minimal description of the implementation. THALES is relying on its internal proprietary language that will be publically available on https://www.polarsys.org/projects/polarsys.capella.

Three case studies are implemented for the validation of the solutions established in GEMOC. The industrial case studies rely on several heterogeneous cooperative xDSMLs relying on various MoCCs that are implemented using the GEMOC Studio provided by WP4, using the processes, methods and tools defined in WP1 and WP2 and the formal framework provided by WP3. The experiments first define the languages, then the models and validate the proposed technologies through the heterogeneous simulation of models.

1.2 Definitions & acronyms

- **AS:** Abstract Syntax.
- **API:** Application Programming Interface.
- **Behavioral Semantics:** see **Execution semantics.**
- **CCSL: Clock-Constraint Specification Language**.
- **MoCML: Model of Computation Modeling 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.
- **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.
- **Model:** model which contributes to the convent 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.
- **TESL:** Tagged Events Specification Language.
- **xDSML:** Executable Domain-Specific Modeling Language

.

1.3 Intended Audience

This document mainly targets GEMOC in WP1, WP2, WP3 and WP4 partners.

1.4 References

- D5.1.1: Technical Requirements, uses-cases specification and metrics for experimentations
- [UML2012] The Unified Modeling Language release 2.4.1, Infrastructure & Superstructure, The Object Management Group, 2012 (http://www.omg.org/spec/UML)
- [fUML2013] The Semantics Of A Foundational Subset For Executable UML Models, release 1.1, The Object Management Group, 2013 (http://www.omg.org/spec/FUML)
- [SysML2013] The System Modeling Language, release 1.3, The Object Management Group, 2013 (http://www.omg.org/spec/SysML)
- [OCL2012] The Object Constraint Language, release 2.3.1, The Object Management Group, 2012 (http://www.omg.org/spec/OCL)
- [Simulink2014] Simulink Reference Documentation, release 2014a, The MathWorks, 2014 (http://www.mathworks.fr/fr/help/simulink/index.html)
- [Stateflow2014] Stateflow Reference Documentation, release 2014a, The MathWorks, 2014 (http://www.mathworks.fr/fr/help/stateflow/index.html)
- [AADL2012] Architecture Analysis and Design Language, release 2.0, Society of Automotive Engineers, 2012 (http://standards.sae.org/as5506b/)
- [D1.13:GA2011] D1.13: Functional Modeling Guidelines, 2011, GeneAuto consortium (https://gforge.enseeiht.fr/docman/view.php/25/4247/D1.13+Functional+Modelling+Guidelines_1.3_110126_AnTo.pdf)
- [D1.14:GA2009] D1.14: Modeling Guidelines for State Diagrams, 2009, GeneAuto consortium (https://gforge.enseeiht.fr/docman/view.php/25/3399/D1.14_Modelling_guidelines_for_state_diagrams_v1.1_090130_AnTo.pdf)
- [BL2014] GeneAuto/HiMoCo/P Block Library Specification Language, work in progress, 2014, P/HiMoCo consortium (http://block-library.enseeiht.fr/bl/)
- [Kahn74] G. Kahn, "The semantics of a simple language for parallel programming," in Proc. IFIP Cong. '74, Amsterdam: NorthHolland, 1974.
- [Petri66] Carl Adam Petri. Communication with Automata. PhD Thesis, Universitat Hamburg, 1966.

- [Lee87] Edward A. Lee, David G. Messerschmitt. Proceedings of the IEEE, vol. 75, no. 9, p 1235-1245, September, 1987.
- [Harel84] David Harel. StateCharts: A Visual Formalism for Complex Systems. 1984.
- [Murthy02] Praven K. Murthy, Edward A. Lee: Multidimensional Synchronous Dataflow. IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 7, JULY 2002

1.5 Summary

This document defines the content of the software delivered in Task 5.2 and the location of the associated elements (source code, binary code, plugins, features, documentation, tests...).

Section 1 defines the scope of the document and presents the document structure.
Section 2 describes the content of the Thales use case.
Section 3 describes the content of the INRIA use case.
Section 4 describes the content of the Airbus and IRIT use case.

2. THALES: Radar Simulation use case

This use case is relying on internal THALES software parts that will be publically available on https://www.polarsys.org/projects/polarsys.capella. Parts specific to the GEMOC project are available on the project repository in org/gemoc/sample/capella.
The Arcadia engineering method mainly focuses on functional analysis, complex architecture definition and early validation. It is highly extensible and customizable through viewpoints providing integrated specialty engineering support. Both the method (which is going to be published and standardized) and the Capella ecosystem are already operationally deployed within Thales in defense, aerospace, space, transportation and security business domains, across several countries, thanks to a large-scale rollout of model-based approaches, with hundreds of daily users worldwide, on critical operational projects.

2.1 Abstract syntax

The Thales use case relies on the the Capella metamodel and more precisely the Data Flow and Finite State machine parts.

2.2 Intended semantics

The Data Flow part of the language follows the Synchronous Data Flow model of computation as defined by [Lee87]. We plan to extend it to handle multi-dimensional data. The Finite State Machine part of the language follows the Harel Statechart proposal given in [Harel84]. We plan to extend it to handle real time constraints. More details are given in the THALES specific annex.

2.3 xDSML components for Capella

The xDSML components for Capella have been developed according to instantiation of the GEMOC methodology, which has been defined in the WP1.

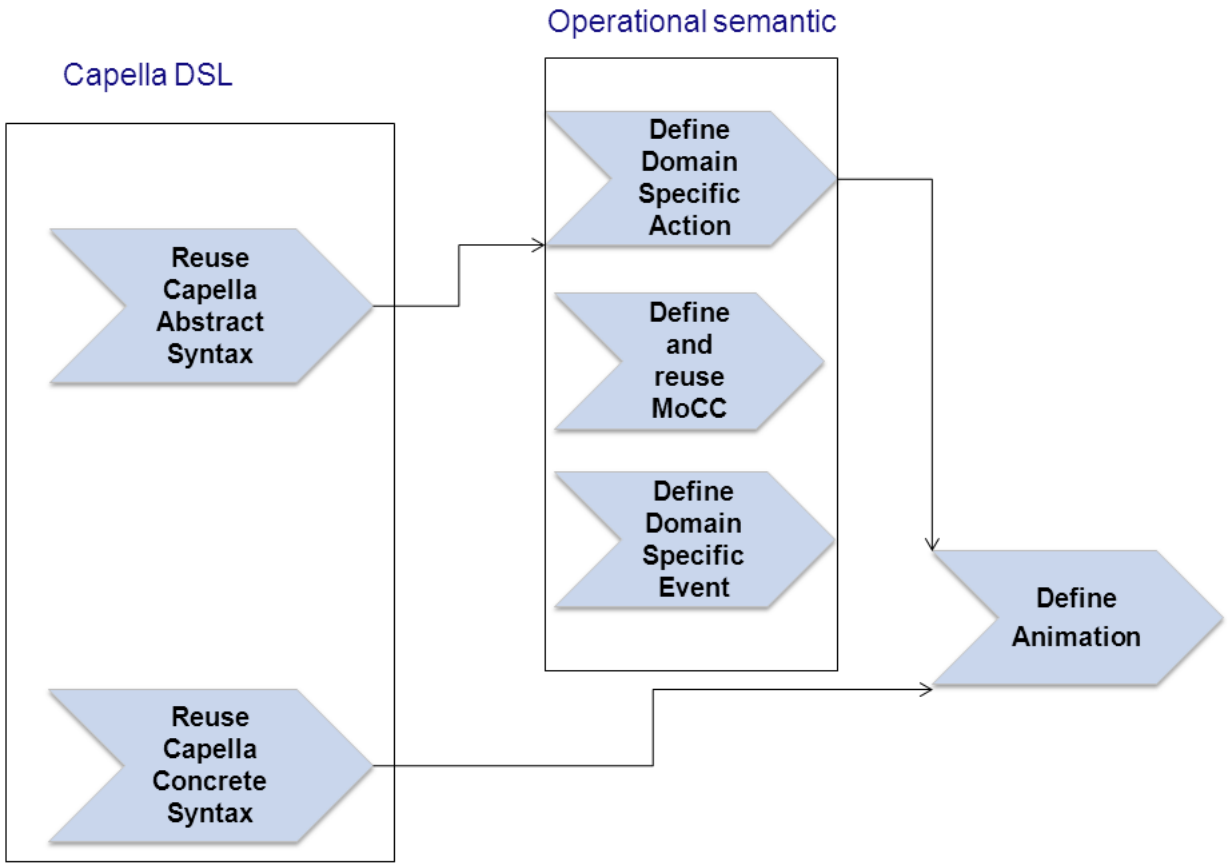

Figure 1: Gemoc method for specifying the Capella xDSML components

The figure below represents the definition of ours Capella xDSML components developed thanks to the GEMOC language workbench provided by the WP4

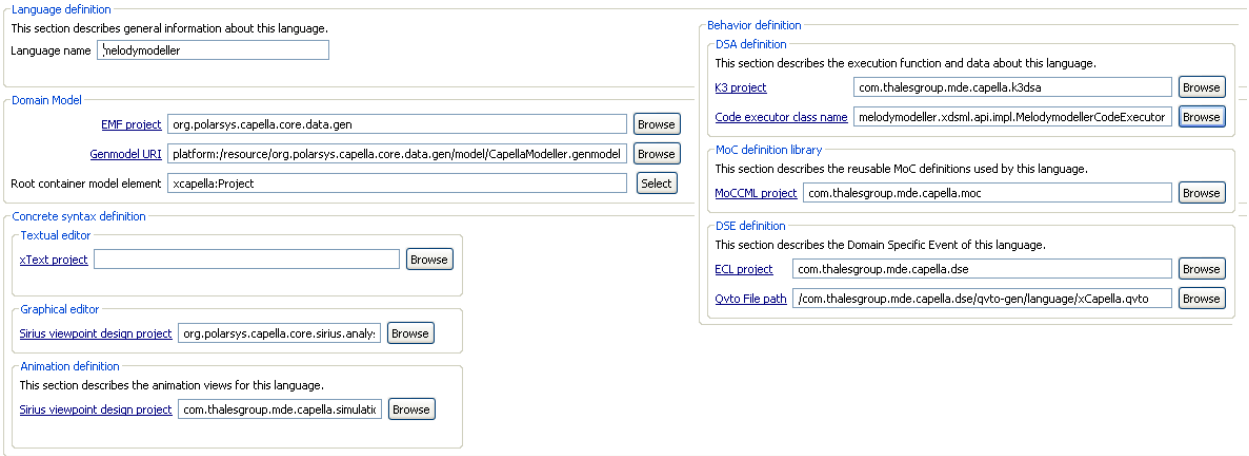

Figure 2: Capella xDSML file

In the following sections, we present briefly each sub-project (see Figure 3: Capella projects organisation) required to achieve the executability of our existing models.

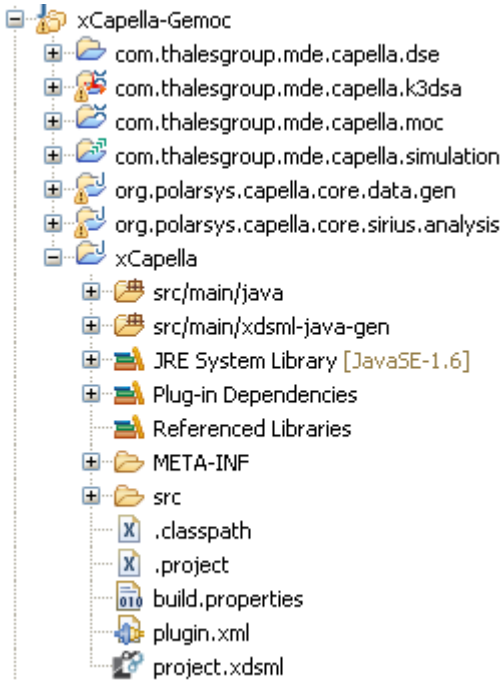

Figure 3: Capella projects organisation

2.3.1 AS

The "org.polarsys.capella.core.data.gen" project deal with the Abstract Syntax of our language. It contains the java implementation of the metamodel generated by the EMF tools chain.

2.3.2 DSA

The "com.thalesgroup.mde.capella.k3dsa" contains the definition of specific actions that are called during the execution of models. Those specification actions are weaved to the metamodel thanks to the use of KerMeta 3 aspects. Those operations are called by the MoCC through the DSE.

2.3.3 MoCC

The MoCC defines the valid succession of events (or simultaneity of events) during any execution. We use MoCML to specify constraints between logical / physical clock. From this specification, the MOCML solver tries to find a valid execution satisfying the constraints. The specification of temporal constraints in MOCML is completely independent from the definition of the AS. The definition of the relevant MoCCs has been done in the "com.thalesgroup.mde.capella.moc" project. The link between the AS, the DSA and the MoCC is performed in the DSE project.

2.3.4 DSE

Finally, when specific actions have been weaved to the metamodel and the MoCC defined, we can make the link between them. This is done in the "com.thalesgroup.mde.capella.dse" project.

2.3.5 Concrete Syntax

When the language and its semantics have been defined, we need a concrete syntax to edit new/existing models. For this purpose, the "org.polarsys.capella.core.sirius.analysis" project specifies how model elements shall be rendered in a graphical editor.

3. INRIA: fUML use case

The purpose of this use case was the early validation of the proposed technologies. It was developed with the earliest versions of the tools. It has reached its purpose and has not been adapted to the more recent versions of the toolset as it was on the one hand not complex enough to assess the new features of the toolset and on the other hand too complex to be used in tutorials or beginners examples.

3.1 Abstract syntax

The abstract syntax is taken from the fUML standard metamodel [fUML2013].

3.2 Intended semantics

The intended semantics is the one from the fUML standard [fUML2013]. It should be compliant with the fUML reference implementation (http://portal.modeldriven.org/content/fuml-reference-implementation-download).

3.3 xDSML components for fUML

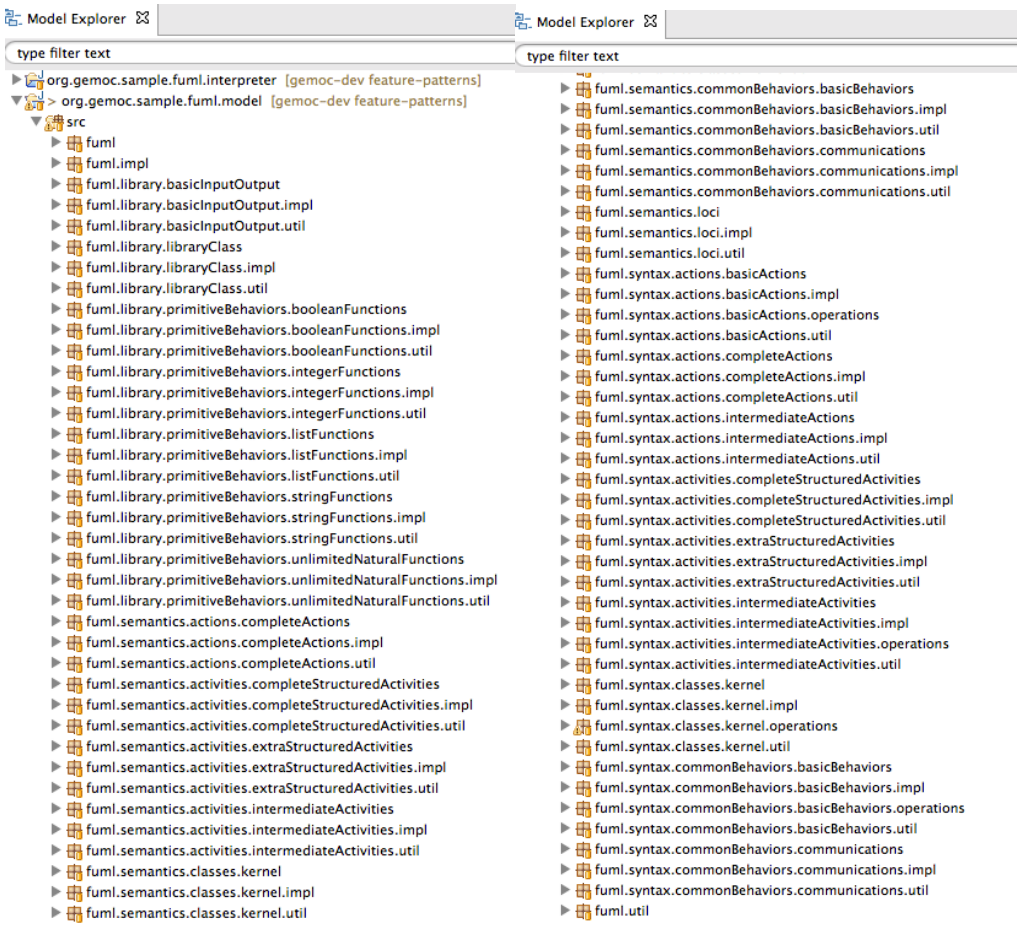

Figure 4: Plugins in the Kermeta 2 fUML implementation

The xDSML project is located on the git repository at: gemoc-dev/org/gemoc/sample/fuml. This first version has been implemented using the Kermeta language version 2 (see Figure 4).

3.3.1 AS

The abstract syntax is the standard one provided by the OMG as an UML class diagram. It has been converted to the ECORE format from the Eclipse Modeling Framework.

3.3.2 DSA

The DSA are defined using the aspect construct from Kermeta as operations and attributes extended the AS..

3.3.3 MoCC

The MoCC is implemented with MOCML which directly triggers the Kermeta operations..

3.3.4 DSE

The DSE are defined as metaclasses that extends the AS.

3.3.5 Concrete syntax

The purpose of this experiment is not to animate models thus no graphical concrete syntax has been defined.

4. Airbus and IRIT: Maintenance Scenario Simulation use case.

4.1 Abstract syntax

The abstract syntaxes are taken on the one hand from the GeneAuto (http://www.geneauto.org) and HiMoCo/P (http://www.open-do.org/projects/p) projects metamodel for Data flow and State machine models related to Simulink/Stateflow; and on the other hand on the UML standard and its implementation in the Eclipse UML plugins.

4.2 Intended semantics

The semantics is, on the one hand, the one described in the GeneAuto [D1.13:GA:2011], [D1.14:GA:2009] and HiMoCo/P projects Tool Requirements including the Simulink/Stateflow and UML parts and the associated Tool Operational Requirements; and, on the other hand, the one from the SAE AADL standard [AADL2012] including the behavioural annex. This second one will be derived from the AADL2Fiacre formal semantics of AADL.

4.3 xDSML components for Simulink/Stateflow

The xDSML project is located on the git repository at: org/gemoc/usecases/geneauto.

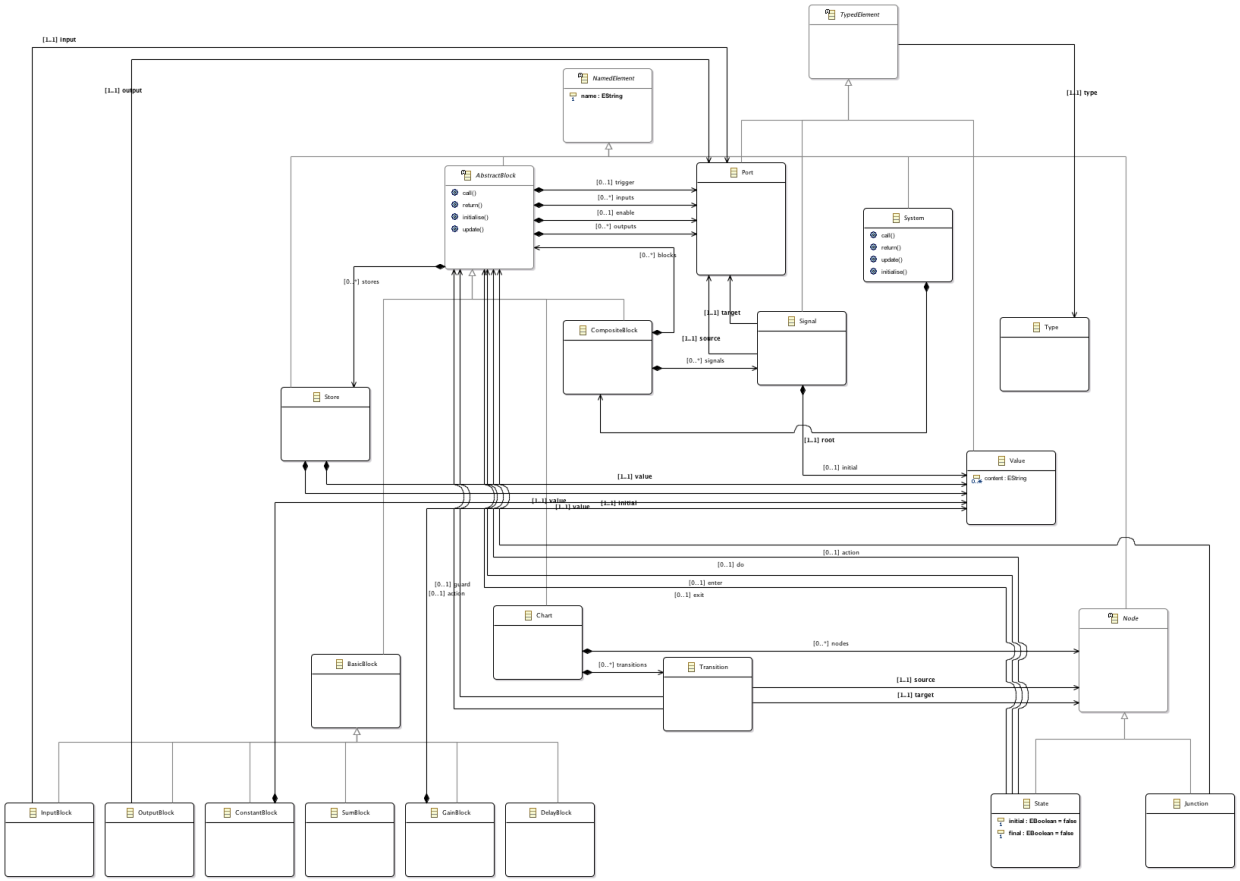

Figure 5: Excerpt from the GeneAuto/HiMoCo/P metamodel

4.3.1 AS

The AS are available in the org.geneauto.language.geneauto (Figure 5) and org.gemoc.usecases.statecharts and in the org.eclipse.uml2 plugins.

4.3.2 DSA

GeneAutoAspect.xtend

```
package org.gemoc.usecases.geneauto.k3dsa

import static org.gemoc.usecases.geneauto.k3dsa.BlockAspect.*

abstract class BlockAspect {

@Aspect(className=typeof(ConstantBlock))
class ConstantBlockAspect extends BlockAspect {

    def public void initialise() {
        _self.value = ModelFactory.eINSTANCE.createValue()
        _self.value.content.add("1")
    }

    def public void compute() {
        (_self.outputs.get(0) as Port).publish(_self.value)
    }

}

@Aspect(className=typeof(Port))
class PortAspect {
    public Value value
    def public void publish(Value _value) {
        _self.value = _value
    }
}

@Aspect(className=typeof(Signal))
class SignalAspect {
    public Value value
    def public void forward () {
        _self.target.publish(_self.source.value)
    }
}
```

Figure 6: Excerpts from the DSA specification

The DSA are defined in the org.gemoc.usecases.geneauto.k3dsa and org.gemoc.usecases.statecharts.k3dsa projects (excerpts are given in Figure 6). Parts of the DSA are automatically generated from the specification of the block library from the Project P/HiMoCo projects [BL2014].

4.3.3 MoCC

The MoCC is defined in the org.gemoc.usecases.geneauto.mocc and org.gemoc.usecases.statecharts.mocc projects.

4.3.4 DSE

geneauto.ecl

```
import 'platform:/plugin/org.gemoc.usecases.geneauto.model/model/model.ecore'

ECLimport "platform:/plugin/fr.inria.aoste.timesquare.ccslkernel.model/ccsllibrary/kernel.ccslLib"
ECLimport "platform:/plugin/fr.inria.aoste.timesquare.ccslkernel.model/ccsllibrary/CCSL.ccslLib"

package model

context System
    inv: Relation Alternates(self.blocks.compute,self.blocks.update)
    inv: let lastInitialise : Event = Expression Sup(self.blocks.initialise) in
        let firstCompute : Event = Expression Inf(self.blocks.compute) in
        Relation Precedes(lastInitialise,firstCompute)

context Block
    def: initialise : Event = self.initialise()
    def: compute : Event = self.compute()
    def: update : Event = self.update()
    inv: Relation Precedes(self.initialise,self.compute)
    inv: Relation Precedes(self.compute,self.update)

context Signal
    def: forward : Event = self.forward()
    inv: Relation Precedes(self.source.block.compute,self.forward)
    inv: Relation Precedes(self.forward,self.target.block.compute)

endpackage
```

Figure 7: Excerpts of the DSE specification

The DSE are defined in the org.gemoc.usecases.geneauto.dse and org.gemoc.usecases.statecharts.dse project (excerpt are given in.Figure 7)

4.3.5 Concrete syntax

The current purpose of the first part of this experiment is not to animate models and thus no graphical concrete syntax has been defined. A textual syntax has been given to ease the definition of validation models in org.geneauto.xtext.geneauto. User models will be imported from Simulink/Stateflow relying on GeneAuto/Projet P toolsets.

5. Conclusion

This report provided a synthetic account of the executable domain specific modelling language provided as use cases in the GEMOC project. These use cases are provided as software artefacts on the project repository. The Thales use case relies on elements from the Capella project available on the PolarSys IWG repository. The fUML use case was used for early validation of the proposed technologies and first versions of the studio and is currently not upgraded to the new versions of the studio.