ENSTA Bretagne
2, rue François Verny
29806 BREST cedex
FRANCE
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

SPID TSe

promo 2017

October 8, 2016

UML-DSimulator

Internship report

IETA MICHAËL RIGAUD

Intership company: University of Antwerpen

Stage chief: Prof. Hans Vangheluwe

Tutor: Simon Van Mierlo

Abstract

UML-DSimulator is an open source[1] plugin which adds a simulator to UML Designer. This plugin has been entirely produced during this internship. It is based on a simulator developed by Ciprian Teodorov although other simulator can be used. This report explains the condition of this internship and the technical work done.

[1] under GNU GPL license v3

Acknowledgement

First of all, I would like to express my deep gratitude to Professor Hans Vangheluwe and Mr Simon Van Mierlo, my research supervisors, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to thank Professor Champeau, for his advice and for giving me the opportunity to do this internship. My grateful thanks are also extended to Professor Teodorov for his help in the manipulation of his simulator.

I would also like to extend my thanks to all members of the MSDL laboratory for their help in offering me the resources in running the program, and their welcome. And more particularly to Yentl Van Tendeloo for his technical help and Claudio Gonçalves Gomes for taking time to read my report.

Table of contents

Introduction

During my second year school at ENSTA Bretagne, Mr Champeau taught us UML Diagrams. During this lesson, he shown us the possibility to create Codes from UML Diagram and the possibility to simulate UML Diagrams such as an overview of the running. But to do that, he needed a tool to create UML Models and simulate them. The two more user-friendly tools which permit that are: Rhapsody and Papyrus.

Papyrus uses Moka to simulate UML Model and it was not well adapted for his lesson, so he chose Rhapsody. However, problems are that Rhapsody is not an open source software, it is only for Windows OS, and it is not free. That is why many students said that they would not use this software outside the lesson.

Therefore, Mr Champeau has proposed this internship to fill in the lack of simulator in open source UML Modelers.

This report is going to present the evolution of the project during this internship. To begin, I will present the host structure. Then, I will explain the situation before the beginning of the project so issues, aims, and the tools choices to preserve the organization of the project. Afterwards, I will exhibit technical choices given and the result of the technical part. To finish, I will highlight the contribution of this internship for my professional project.

Part I

Context

CHAPTER 1

Description of the host structure

My position

I did my internship in the University of Antwerp in the MSDL[1] laboratory. My supervisor was Prof. Hans Vangheluwe. But, because he is always busy and not always in the university for professional reasons, I was attached to Simon Van Mierlo for this internship. As a PhD student at the University of Antwerp, he has created a debugger for the simulator of Statechart SCCD.

Simon was chosen to be my tutor for two reasons. Firstly, because all my work would be very similar to a debugger interface. Secondly, because Prof. Vangheluwe expected me to use the SCCD as simulator and compare it with the simulator of Mr. Teodorov

During my internship I worked in the office of Simon and Yentl Van Tendeloo. However, I had the opportunity to interact with all members of MSDL.[2]

Position of MSDL

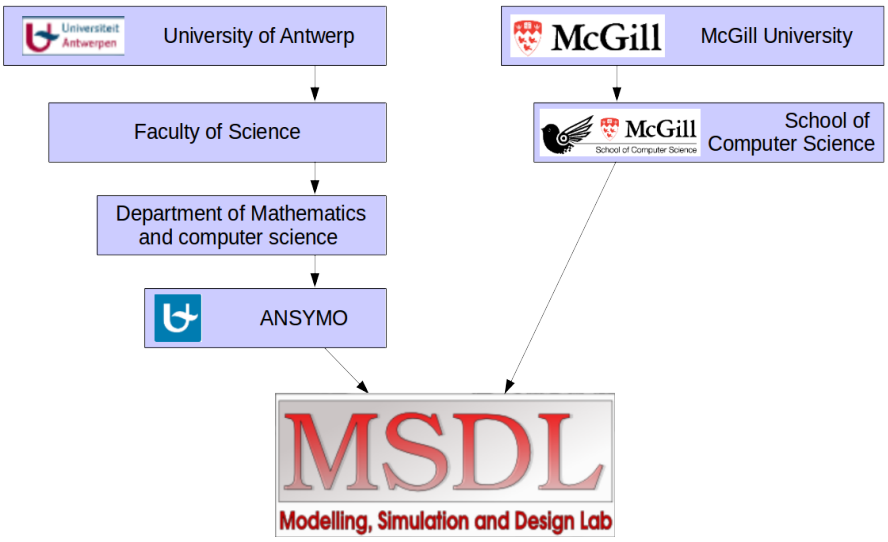


Figure 1.1: Position of MSDL

The MSDL laboratory has members in two schools, most part of these members are in the University of Antwerp but there are some PhD students in the McGill University. McGill University is an English-language public research university in Montreal, Canada.

[1] Modelling, Simulation and Design Lab

[2] It is possible to find a description of all MSDL members in `http://msdl.cs.mcgill.ca/people`

You can see on the Figure 1.1 a description of the organization of these universities[3] and the position of MSDL in this organization.

Description of MSDL

Figure 1.2: MSDL banner

> «The Modelling, Simulation and Design lab (MSDL) headed by Prof. Hans Vangheluwe is part of the School of Computer Science of McGill University in Montreal, Quebec, Canada and of the AnSyMo (Antwerp Systems and software Modelling) group in the department of Mathematics and Computer Science of the University of Antwerp, Antwerp, Belgium. The MSDL has projects, researchers and students in both locations.»[5]

So MSDL is a laboratory of Modelling. Modelling is an important field of computer science because it captures ideas in complex systems without all the accidental complexity of traditional software development. Models that have a well defined semantics also permit to create software without programming. I had the opportunity during my internship to participate in a presentation of all research subject. The Figure 1.3 is a not exhaustive list of all research topics discussed in MSDL. The ones most related to my subject are *Simulation* and *Analysis, Validation, Verification, Testing, and Accreditation*.

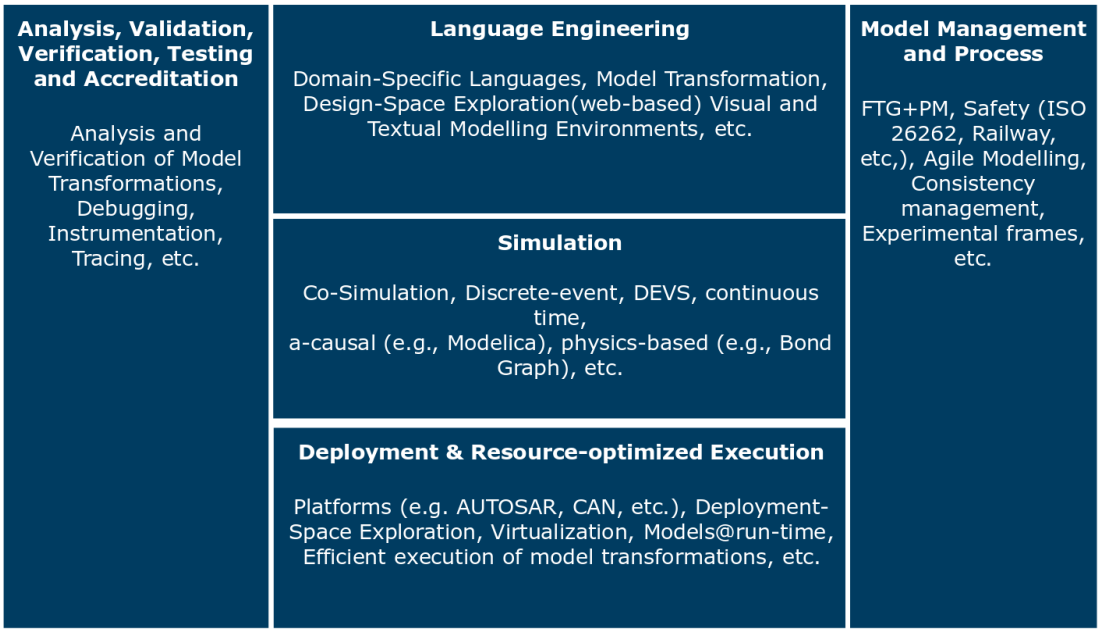


Figure 1.3: Research topics

[3]Only departments which belongs msdl were presented.

Part II

Presentation of the project

CHAPTER 2

Goals

As it was explained in the introduction, the aim of this project is to propose a visualization of UML Model for UML Designer.

Short term goals

In the short term, this plugin should permit Mr. Champeau and Mr. Teodorov to use UML Designer in the classroom, as a free and portable alternative to Rhapsody. In this way, every student will be able to install on their own computer the tool use in classroom.

Advantages of UML Designer when compared to Rhapsody: free, open source, works on Windows, Linux, and Apple.

Long term goals

Because this plugin is open source and downloadable on a Git Server[1], it can be used by anybody. My hope is that this plugin can be improved by the community, student, teacher, *etc*... and becomes a serious alternative to Rhapsody and Papyrus simulator.

Figure 2.1: Rational Rhapsody

Figure 2.2: Papyrus

[1] `https://msdl.uantwerpen.be/git/michael/UML-DSimulator`

CHAPTER 3

The plans of the project

Definition of my project

After some interviews with Mr. Champeau, I wrote the table of requirements (Table 3.1). This table contains all requirements that satisfy the goals of this project define before.

FS means service function and *C* means constraints.

Table of requirements		
Number	Type of Designation	Designation
FS1	UI	Represent the visualization in the UML Model with graphical tools
FS2	UML Designer	Works by default with the the Ciprian simulator
FS3	Simulator	Have the possibility to change the simulator with other simulator, for example SCCD
FS4	Simulation	Give some debugging tools, for example play, stop, *etc*...
C1	License	Open Source
C2	Compatibility	Be multiplatform. Works on Linux, Windows, and Apple.
C3	Documentation	Have documentation, code readable, modularity, etc. In that way, this project could be improved during another internship.
C4	User	Be user-friendly

Table 3.1: Table of requirements

Then, it is also possible to represent the project with a octopus diagram (Figure 3.1). This diagram permit to show the interaction of the simulator with the outside world that the client expected.

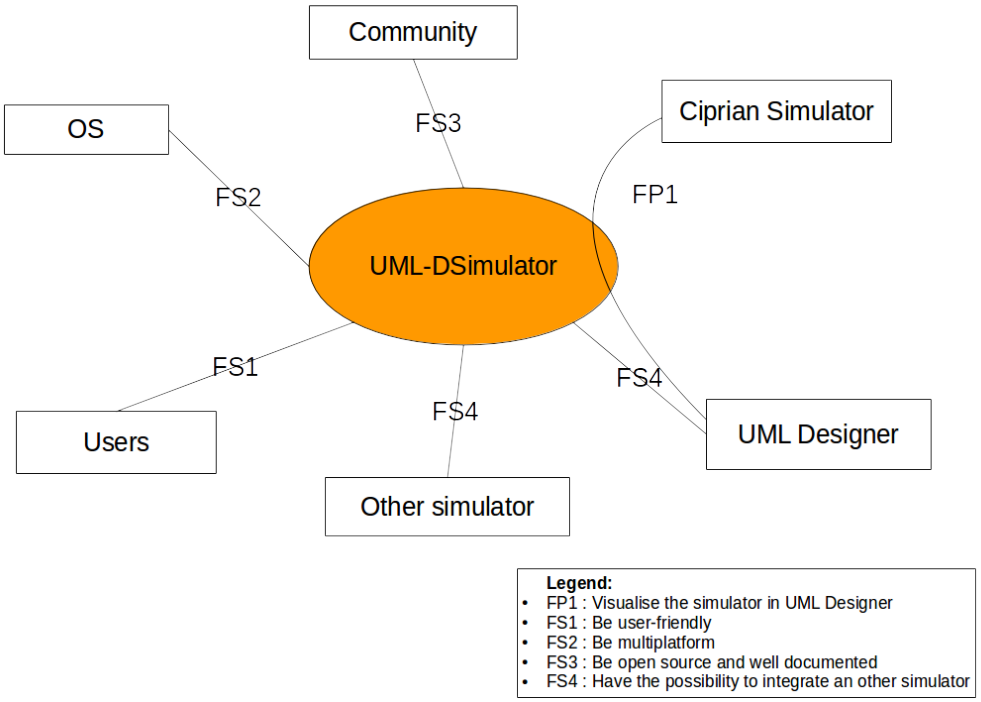


Figure 3.1: Octopus diagram

Goals

As it was explain in the introduction, the main goal of this project is integrate a simulator in UML Designer. To do that Mr. Ciprian put at my disposal his own Simulator. So it is possible to describe my goals in this order:

1. Find the way to add plugin in UML Designer, and understand how it is possible to add more features.
2. Understand how the Ciprian simulator works.
3. Find a way to integrate the simulator but keep the possibility to change it. Moreover it should be kept in mind that the simulator was not finished so the integration of the simulator need to preserve modularity.
4. Propose some debugger tools. For example: a play button, a stop button, *etc*.
5. Write documentation and comment in the code to be reusable. Mr Champeau wanted to keep the choice to make some improvements after the end of this internship.
6. Try other simulator and compare its performance with the Ciprian simulator.

Tools at my disposal

This section presents a short description of UML Designer and the Ciprian Simulator. More details can be found in the appendix A and B.

UML Designer

Figure 3.2: UML Designer logo

At the beginning of this project, some tools were at my disposal. First of all, it was UML Designer created by the french company *Obeo*. It is an open source software documented so I could download the source and work on it. It is a UML modeler with a user interface. It is based on Eclipse and Sirius. It follows the UML2 standard which is well known and documented.

Simulator

Then, Mr Ciprian Teodorov, one of my professors, has developed a simulator for UML Model. This simulator needed some improvements, but it was a good starting point for this project.

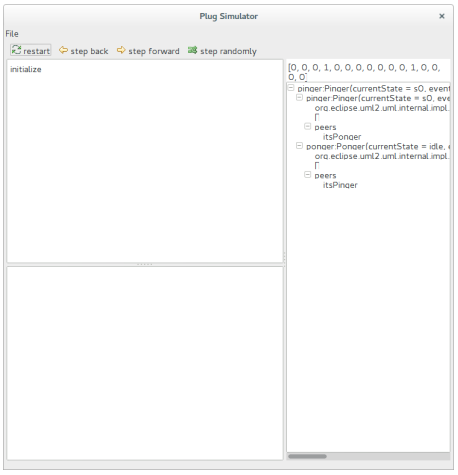


Figure 3.3: Mr Teodorov simulator

CHAPTER 4

Organization

At the beginning of this internship I made many choices in the organization of my work. This chapter details those choices.

Calendar

First of all, in June, I made a calendar of my planning works. This calendar has given me a good overview of the time that I am allowed to spend in every task.

Tasks/weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14
State of the art	-	-								X				
Create a plugin			-							X				
Visualize the simulation				-	-	-	-	-		X				
Unit tests								-		X				
Integration tests									-	X				
Try an other simulator										X	-	-		
Redaction		-	-	-	-	-	-	-	-	X	-	-	-	
Oral						-				X				-

Table 4.1: Calendar

During the week 10, the University was closed.

Record

The main client of this project is Mr Champeau. So I did every week a record of my work by e-mail to him. In this e-mail I presented the work done during this week and the work planned for the next week. Whenever possible, I included a short video to present my improvements.

I also did a video conference with Mr Champeau and Mr Teodorov in the 1st of July. This video conference was very useful to know if I was on the right track and to clarify some important points.

Tools used for the project

After the begin of this project I decided to use some tools to arrange my work.

I employ Framaboard: a web application which permit to create kanban board[1]. I used it because that allow to have a good visual overview of the project. The Figure 4.1 show my framaboard.

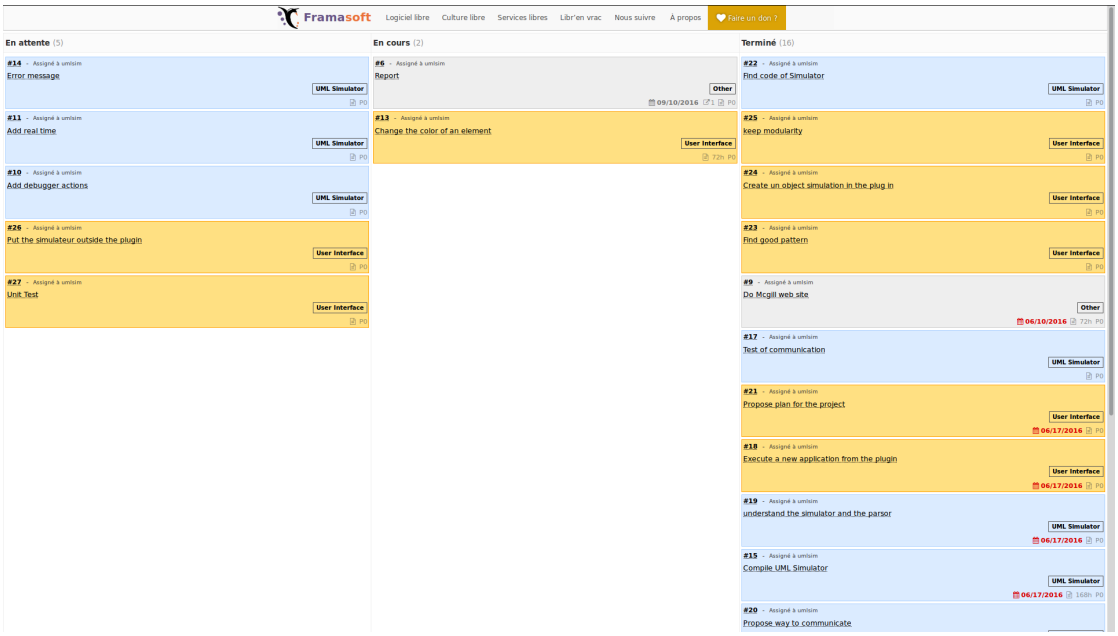


Figure 4.1: Screen shot of the framaboard

Then the MSDL laboratory give me a web page to present my work and my progress. I also put on this web page a resume of all my records. The Figure 4.2 exhibit my personal web page.

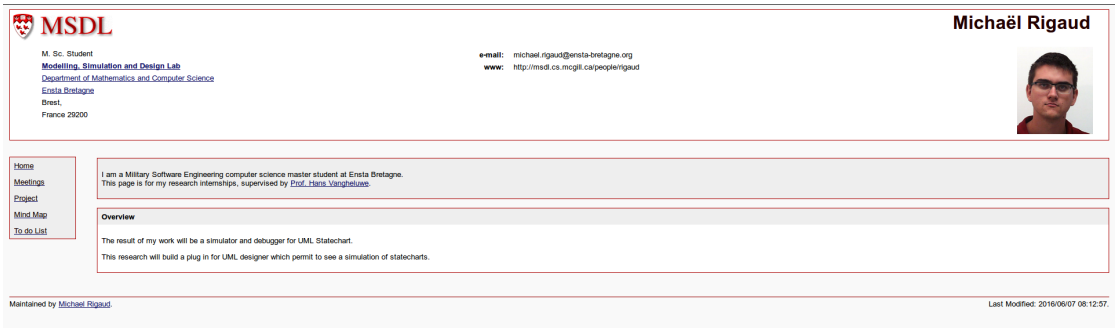


Figure 4.2: MSDL web site

To finish they likewise give me an access to their git server. That permit to lodge my work on a secure server and prevent a data-loss in case my computer broke. Figure 4.3 shows my git repository.

[1]Kanban is a method for managing knowledge used by software teams practicing agile software development.[9]

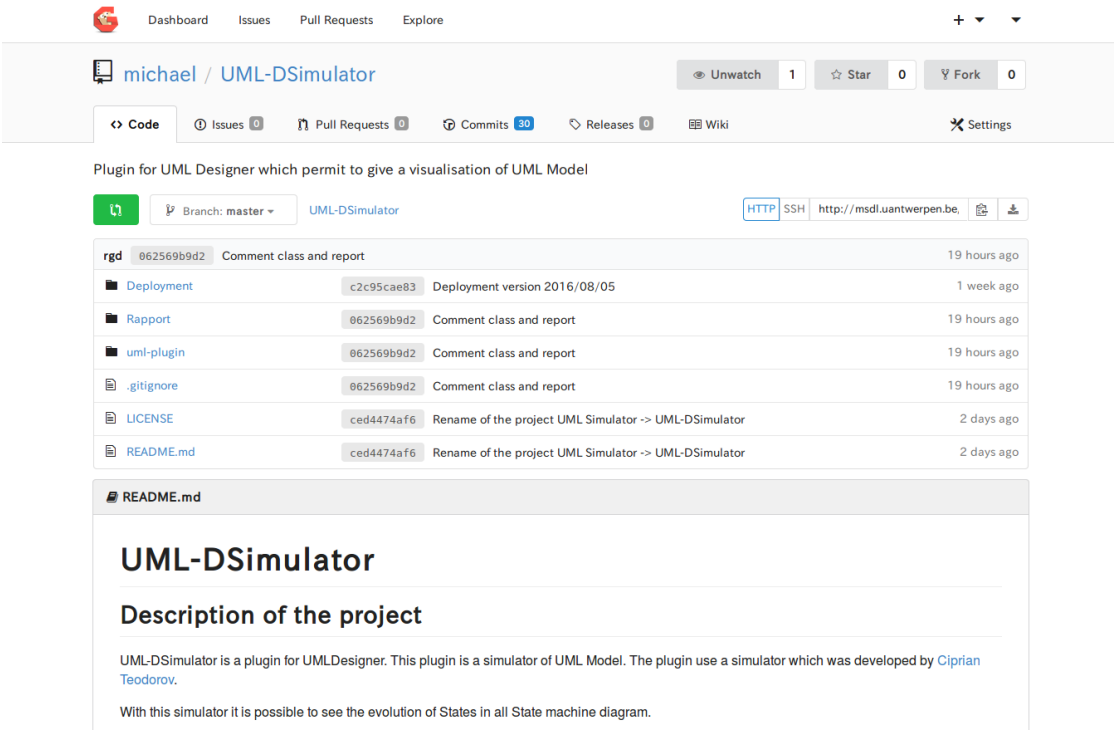


Figure 4.3: git repository

Part III

Results of the internship

CHAPTER 5

Technical choice

During this project I made some technical choice. I will explain in this chapter which choice I take and why.

Type of communication

First of all, I had to find the best way to integrate the simulator in UML Designer. Because I want to keep the possibility to change the simulator I have decided to put the simulator in a separate thread with a communication by socket. In this way, I had the possibility to change the simulator without changing everything in the plugin, and even start the simulator outside the plugin.

Afterwards, I had to find the best way to realize the communication enter the simulator and the plugin. After list all kinds of inter-process communication with their advantages and their drawbacks, I decided to chose a socket communication.

Socket

This next table exhibit advantages and drawbacks of socket. You can find on appendix C the list of all other type of communication consider. I chose socket because it has a better ratio of Advantages/Drawbacks for this work.

Advantages	**Drawback**
Work with every language (python, java, ...)	Message need to be formatted
Works over the network	Not very fast
Work on all platform (Windows, Linux, OSX)	

Type of message

Now I have chosen socket to communicate enter the plugin and the simulator, I have to choose which type of object I will send by this socket. In fact, socket message need to be formatted and it exists many way to do that.

In the same way, I list the type of message that I could send, and only three were relevant.

- String
- Java object
- Json message

Because String doesn't permit modularity and Java object require to use java for the simulator layer, I chose Json. Moreover, Json object are sent like String but with a formatted type.

To do that I use a library which permit to manipulate Json object in Java. I found it on Github [10].

Our json object are constructed like this:

plugin $\rightarrow$ simulator

```
JsonPluginToSimulator = {
  initialize : boolean
  play : boolean
  stop : boolean
  restart : boolean
  random : boolean
  reload : boolean
  reloadPath : string
  state : string
}
```

simulator $\rightarrow$ plugin

```
JsonSimulatorToPlugin = {
  transitions : ["transition1", ...]
  error : boolean
  errorMessage : string
  currentClass : string
  currentStates : [
  {
    class : string
    instance : [
    {
      nom : string
      state : ["state1",...]
    }
    ...
    ]
  }
  ...
  ]
}
```

Overview of the project

With this choice, it is possible to better understand how the project is constructed. The project consist in creating a plugin for UML Designer and a communication layer for the simulator. The plugin communicates with the communication layer with socket and json. The plugin receives the data sent from the simulator, analyzes it, and sends instructions to the simulator.

The Figure 5.1 summarizes the structure. By default, the simulator and the communication layer are started by the plugin. So, they are inside UML Designer but it is possible to put them outside because after the start of their thread they communicate only by

socket. In this way, we have a default behavior which is user-friendly because the user does not have to launch an other software, but we can consider that the simulator is outside the plugin. That is why on the Figure 5.1 the simulator is represented outside.

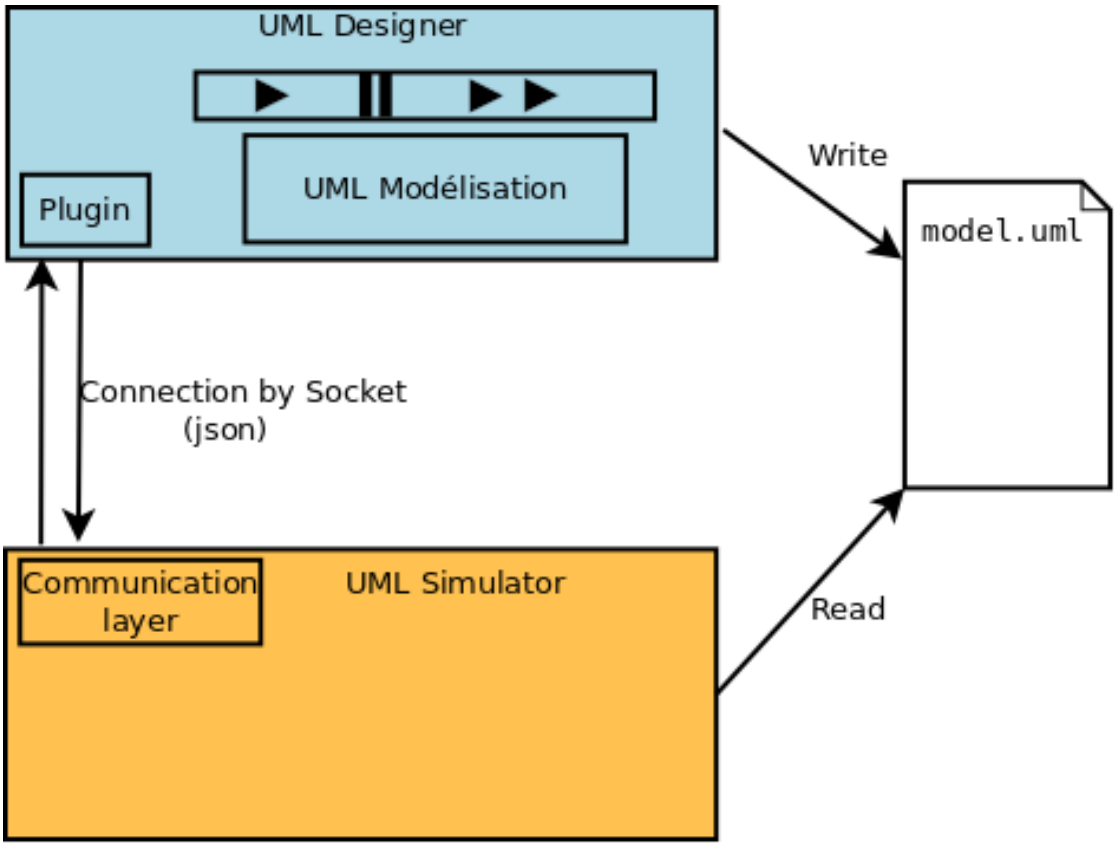


Figure 5.1: overview of the project

CHAPTER 6

The simulator

Afterwards, I worked on the simulator to understand how it works and implement the layer of communication.

Specificity of the UML Model

The Ciprian simulator simulates a UML model but this UML model need to have some specificities in the architecture and in the language.

UML Designer use 2 files to save a uml project. The first is named *model.uml* it contains all UML elements and the declaration of UML diagrams. The second is named *representation.aird*, it contains the specification of the graphical representation, which consists of the positions, colors, and other graphical attributes of the elements described in model.uml file.

To work, the simulator needs the *model.uml* file. Moreover, this file need to contain some specifics features. It needs a class **SUS** which contains the declaration of all other classes and all other classes need to have a State Machine diagram associated, as you can see on the Figure 6.1.

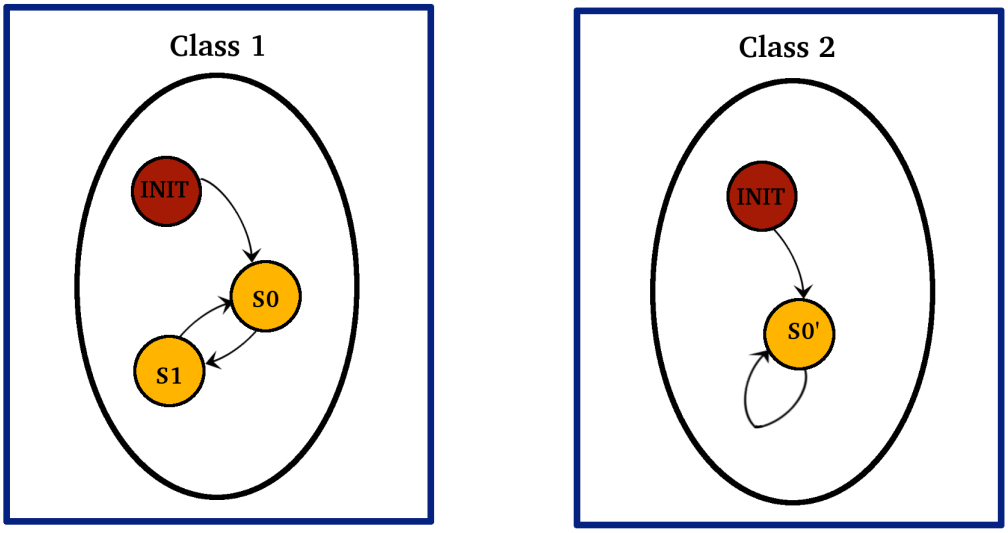


Figure 6.1: Representation of the most important elements for the simulator

Additions to Teodorov Simulator

Then, I made some research on the code of the simulator. I realized that the simulator was not develop to communicate and notify any changing. So I changed the code of the simulator to add an Observer pattern at the class *SimulationModel* because this class is a controller of the simulator.

> «The Observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods.» [11]

I used this pattern to notify the communication class of a changing.

Communication

The communication Layer was written in Java. As it was explain in the chapter before, the simulator and its layer were implemented, without loss of generality, inside the plugin. The main class of the plugin initiates the Simulator in a new thread and then there is only a communication enter the plugin and the simulator though the localhost loop by socket.

Moreover, to be sure that it is possible to change the default simulator with an outside simulator I did some test where the thread is initiated by an other software, and that worked.

CHAPTER 7

Plugin

How to write a plugin

The first thing that I looked for in the UML Designer documentation was how add something in the software. It is a real challenge in software development. Fortunately, I quickly found on the UML Designer developer Guide, that there is a the developer environment to do that. This environment was an Eclipse environment, which is looking as the Figure 7.1.

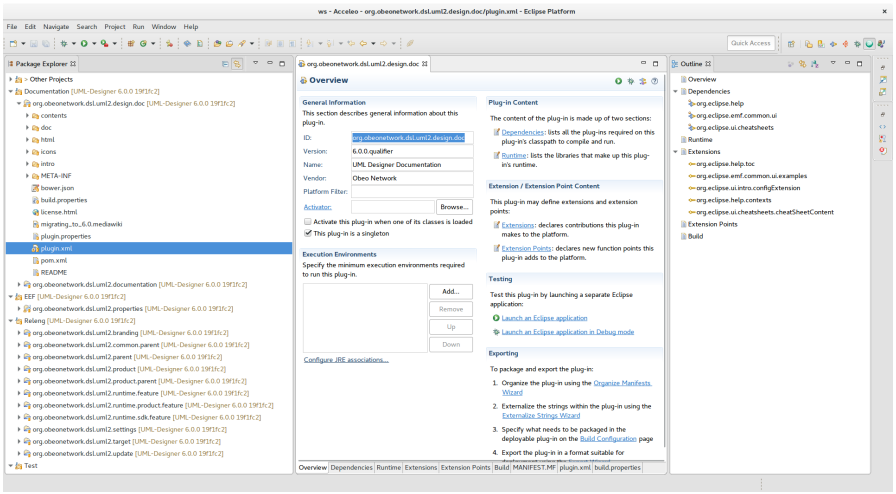


Figure 7.1: Eclipse environment

Then, I use the fact that UML Designer is based on Eclipse to create the plugin. The creation of a plugin is the same as an Eclipse plugin. So I made some research on how write an Eclipse plugin. I learn that there is a special structure. This structure permit to keep the modularity and the possibility to install the plugin in all platform. The structure of an eclipse plugin follow the structure show on the Figure 7.2.

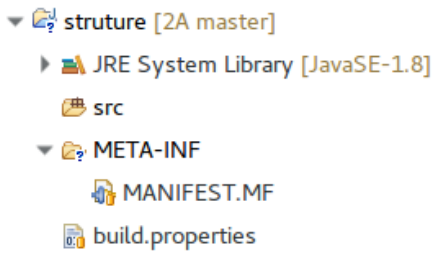


Figure 7.2: Structure of an eclipse plugin

- The directory *JRE System Library* contains all libraries used to run the plugin
- The directory *src* contains sources of the plugin
- The file *MANIFEST.MF* contains addition information for the integration of the plugin
- The file *build.properties* contains information to export the project

General presentation

During the development of the plugin, I tried to separate elements of the UML Designer API, the elements of the UI, and the communication Layer.

The architecture of my plugin follow the UML class diagram presented on the Figure 7.3. However, all link enter classes are not represented on it to not overloaded the model. A short description of all sub package:

MainView It is the main class, it is the first class call by UML Designer when it load the plugin
features This package contains all graphical elements of the plugins
communication This package contains the layer of communication of the plugin
tools This package contains some tools for all class
design This package contains all classes to change the appearance of the UML Model
model This package contain a class with all information of the status of the model during the simulation

Then, it is possible to underline that I use a *Observer* pattern enter classes *MainView* and *CommunicationP* as with the Simulator. This pattern permit to actualize the view of the model when a new communication is received.

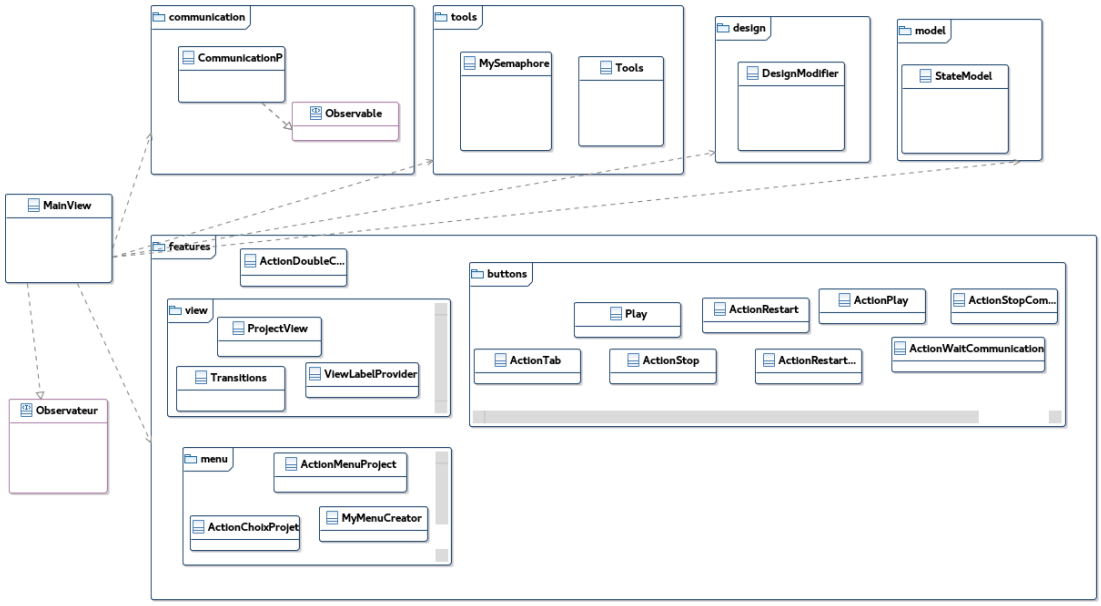


Figure 7.3: plugin class diagram

Functionality implemented

During this project I had time to implement a lot of functionality for the simulation.

On the Figure 7.4, you can see the result of my project. It is an Eclipse view, on the top, there is a list of all transition possible, and on the bottom, a tree of the project view

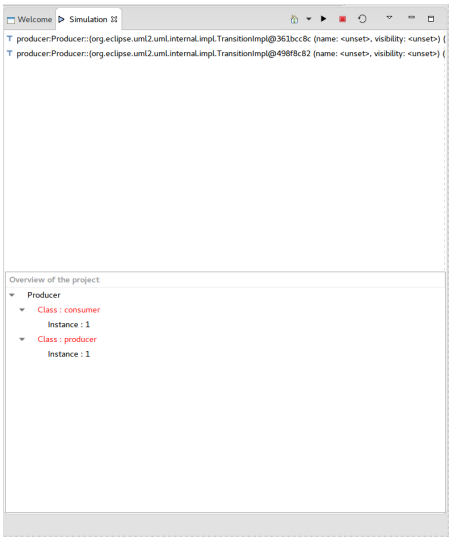


Figure 7.4: result in UML Designer

is shown. The tree has the name of the project simulated, the name of all classes, and the name of all instances.

In the same figure, under the «Simulation» tab, if the user clicks on a transition this transition is selected as the next transition. On the bottom, if he click on an instance of a class, it is chosen as the visible instance of the model, and if he click on the class name all instances of this class are visible (useful only if there is multiple instance for a same class). Current instances visible are in red.

On the top, it is possible to see that there are many buttons. These buttons define the following actions:

home permit to change the project that we want to simulated
play launch a simulation where a random step is chosen every 1s.
stop stop the simulation launch with play
replay restart the simulation of the project
menu contain all previous buttons but also:

- **stop simulator** stop the communication with current simulator
- **wait a simulator** start a new server and wait a communication of a new simulator
- **restart simulator** restart a communication with the default simulator

Finally, I implemented the possibility to see the current state of all state machine diagram in the UML model. You can see on the Figure 7.5 a test that I did on one model given by Ciprian Teodorov. The red state is the current state.

The table 7.1 resume all functionality implemented and also present some functionality suggested during the project but not implemented.

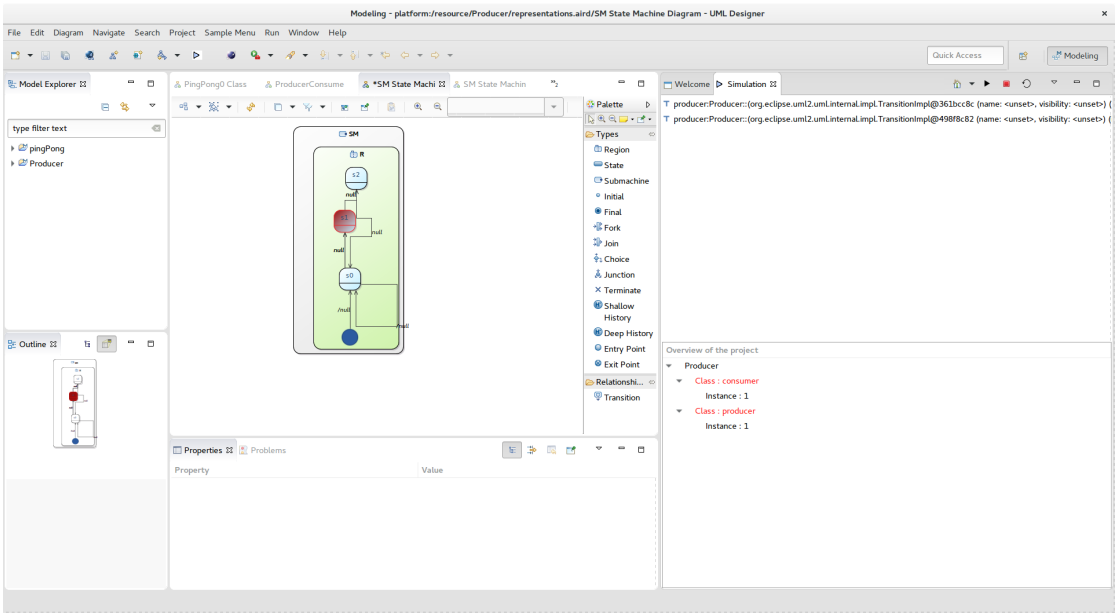


Figure 7.5: result of a simulation

	Functionality	if not implemented why
✓	Visualization of the simulation in State Machine Diagrams and Class Diagrams	
✓	Possibility to select the next transition	
✓	Possibility to change the simulator	
✓	Possibility to start the simulator outside UML Designer	
✓	Possibility to change the current project simulated	
✓	Possibility to restart the simulation for the current project	
✓	Play mode (choose a random step every 1s)	
~	Multiple instances for a unique class	The plugin interface take care about multiple instances but the simulator not
~	Parallel state	The plugin interface can show parallel state but it is not implemented on the simulator
✗	Historical state	It is not yet implemented on the simulator
✗	Generation of the sequence diagram automatically	Not enough time
✗	Select the next transition on the UML Model	Not enough time
✗	Show multiple instances in multiple tabs	Make the project too heavy
✗	Choose a state for debugging	Not enough time

Table 7.1: Functionality implemented

CHAPTER 8

SCCD

In this chapter, I will talk about SCCD. In the MSDL laboratory, they use SCCD to simulate Statechart models. After a presentation of my project in July, they suggested me to try their simulator and compare it to the Ciprian simulator. It is possible to find more explanation about SCCD on the Master's thesis published by Glenn De Jonghe [4] and on the Master's thesis published by Joeri Exelmans [3].

Short description

SCCD[1] is an open source simulator and code generator of Statechart developed by the MSDL laboratory. An SCCD model consists of a class Diagram and a StateChart attached on each classes. It also uses some special semantics to instantiate classes with an object manager.

Because a state machine is a statechart[2], for the rest of this chapter we can consider that our models conform to the SCCD language.

Transformer

As we saw before, the project has been written to have a ability to change the simulator. However, the model needs to be written in scxml standard to be interpreted by SCCD. So the first things that I have to do is a transformer in XSLT. XSLT is a language for transforming XML documents into other XML documents.

After some research on the internet I found only one transformer written by apache on Github[1]. The last commit of this project was in 2009, so we can assume that the project is abandoned. I had tried to use it but it didn't work. For this reason I have created a new transformer, but I used some part of this project.

My transformer has the ability to transform a xmi file as a scxml file. However, model given by my professor had some specificities so I take care to adapt to its.

For example, when there is a script in ABCD language and the script is «send eventA to itsPinger» the translation in the state machine is «raise eventA to itsPinger». Then I use the fact that our project always have a *SUS* class which contains all other classes, so in the scxml model the *SUS* class start all other classes.

[1] signified StateChart and Class Diagram

[2] Statechart is an extension of state machine, invented by David Harel

Utilization

SCCD debugger

In my project I want to visualize states of all state machine. To do that, I need some informations of the status of the model. SCCD don't permit this type of utilization, but the SCCD debugger written by Simon Van Mierlo can do it.

However, during my internship this debugger wasn't finished. The debugger can create only one class because the object manager doesn't work.

To prove that my scxml model created automatically will work when the debugger will be finished, I tried to do a proof of concept.

SCCD

To do this prove of contest, I achieve some tests on the pure SCCD. I use the last version of SCCD published in the beginning of august.

I quickly realize that my model had infinite loop. These loops are explain because in the model there is some state which have transition on itself, and this transitions are always verify. In the Ciprian simulator it was not a problem because the user has to choose the next transition and so he was the condition. To fix this problem I add on these transitions a timer of 1s.

The Figure 8.1 resume the work done to execute xmi model with SCCD. You can also see on the Figure 8.2 the result of the Ping-Pong example on SCCD. As it was expected there is an event which is send from ping to pong and then from pong to ping.

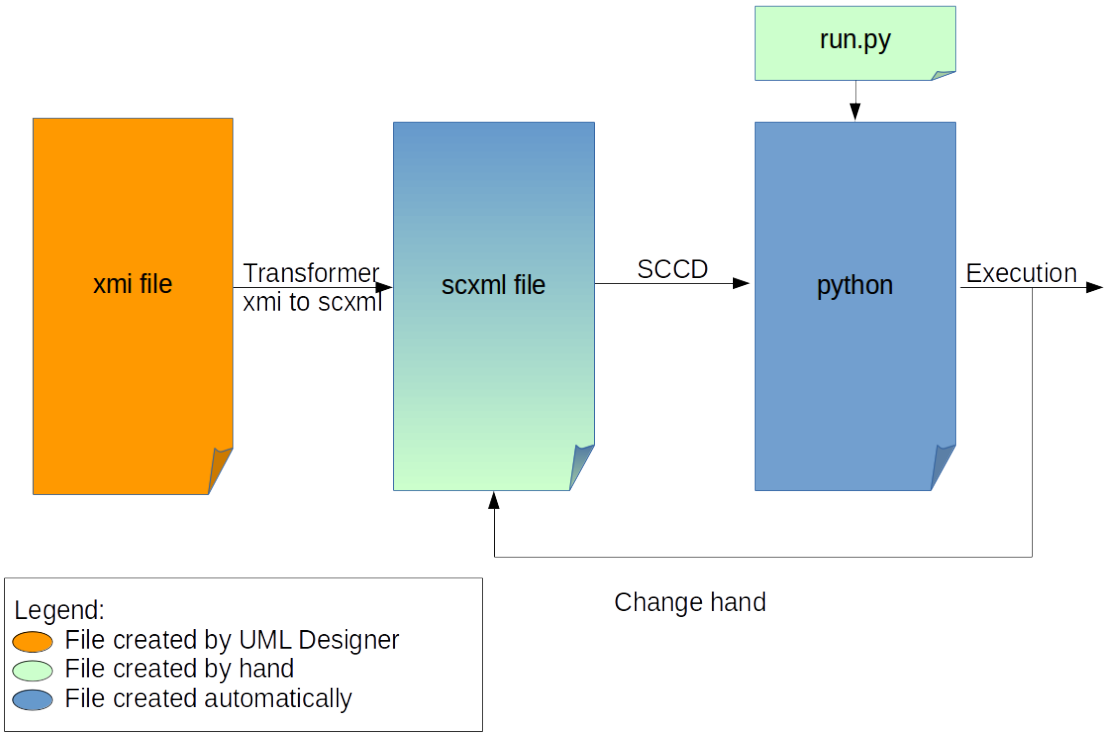


Figure 8.1: Resume

```
send ping to itsPonger;
etat: idle
send pong to itsPinger;
etat: idle
send ping to itsPonger;
etat: idle
send pong to itsPinger;
etat: idle
send ping to itsPonger;
etat: idle
send pong to itsPinger;
etat: idle
send ping to itsPonger;
etat: idle
send pong to itsPinger;
etat: idle
```

Figure 8.2: pingpong simulation on SCCD

CHAPTER 9

Tests

Unit tests

During this project I did some unit tests to preserve the code during the development. But, I had a lot of difficulties to tests user interface features, so I chose, on Simon advice, to don't tests them. However, the architecture of the plugin clearly separates UI code from essentially functionalities, which where tested.

To do this unit tests, I used junit and an eclipse feature EclEmma which permit to see the coverage of code during unit tests. On the Figure 9.1, you can see the result of the coverage show by EclEmma about my project.

First of all, you can see the package json was not well tested. This package was written by stleary[10], so I assumed it was already well tested.

Then, packages org.ensta.uml.sim.views.features and org.ensta.uml.sim.views.design only contain class which do action on the user interface. So I didn't tests them.

After this remarks, it is possible to notice that I tested more then 80%[1] of the code use in other classes.

org.ensta.uml.sim	36,0 %	168	299	467
src	25,4 %	100	293	393
json	18,9 %	46	197	243
org.ensta.uml.sim.views	0,0 %	0	26	26
org.ensta.uml.sim.views.features.buttons	0,0 %	0	22	22
org.ensta.uml.sim.views.features.menu	0,0 %	0	14	14
org.ensta.uml.sim.views.features.view	0,0 %	0	13	13
org.ensta.uml.sim.views.design	0,0 %	0	10	10
org.ensta.uml.sim.views.features	0,0 %	0	3	3
org.ensta.uml.sim.views.model	76,9 %	10	3	13
org.ensta.uml.sim.simulateur	90,9 %	20	2	22
org.ensta.uml.sim.views.communication	88,9 %	16	2	18
org.ensta.uml.sim.views.tools	88,9 %	8	1	9
test	90,9 %	50	5	55
mock	94,7 %	18	1	19

Figure 9.1: Coverage view of my project

Integration tests

I also did some integration tests to verify that the plugin runs in all major platforms (Windows, Linux, Apple).

[1] It is the usual value of acceptable coverage

During all my project I verified that my plugin could be use on my own computer without the eclipse developer environment. I have a Ubuntu 16.04 LTS.

Then, at the end of my project, I tried to use this plugin on other platform. To do that, I use virtual machine with VirtualBox. I tested on a Windows virtual machine with W7 (Figure 9.3), and a Kali virtual machine based on Debian (Figure 9.2). However, I didn't try on OSX because I didn't find a OSX machine.

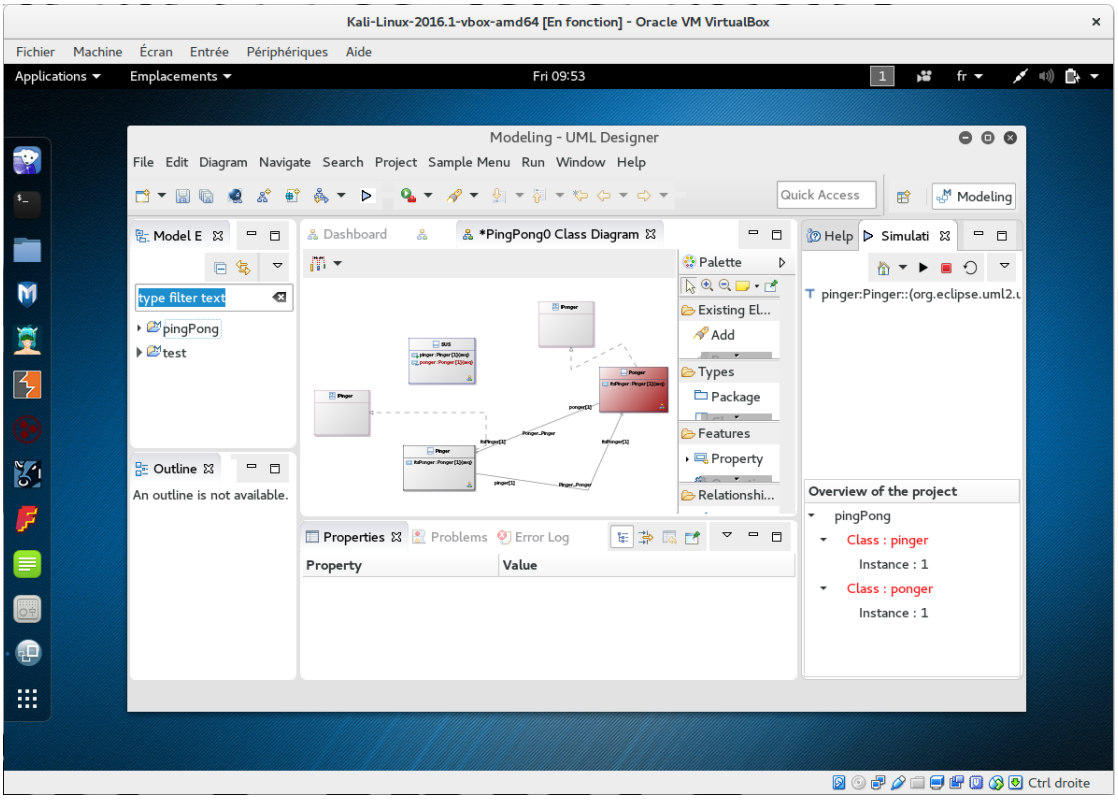


Figure 9.2: Screen-shot of the kali virtual machine

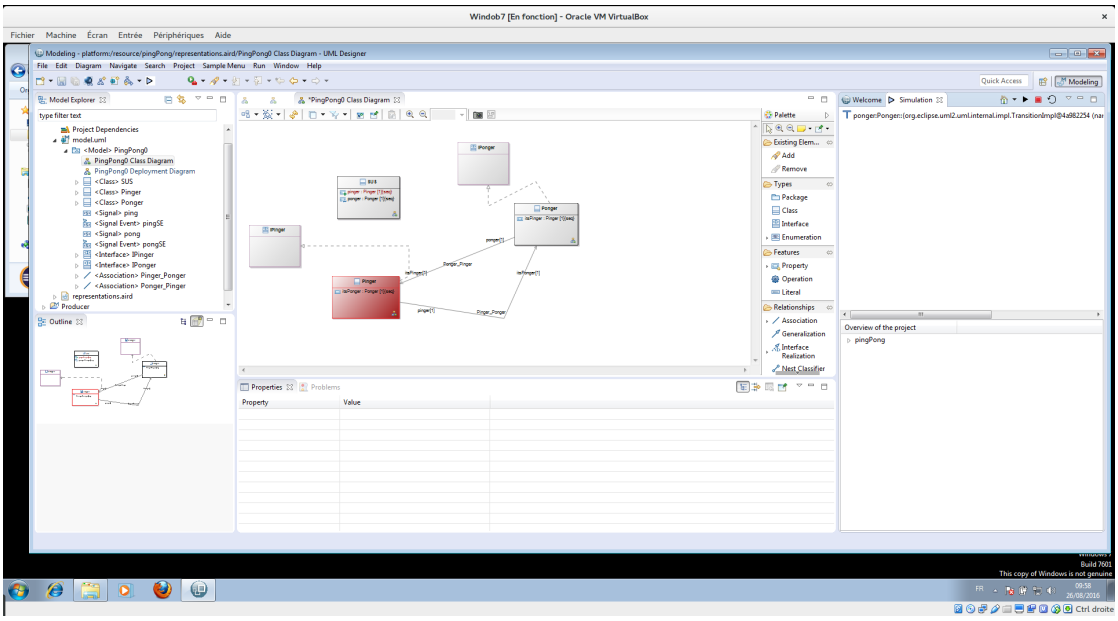


Figure 9.3: Screen-shot of the Windows virtual machine

Part IV

Contribution of this internship for my professional project

CHAPTER 10

Professional and personal balance sheet

Profits from this experience

The acquisition of new technical skills

During this internship I have discovered new technical skills, and I have consolidated skills acquired at ENSTA Bretagne. My project was developed in Java, and I choose to use a managing tools Framaboard (to create Kanban board) and a version control system Git. I understand better why it is important to use these tools. Framaboard was very important to keep an overview of my target and keep in mind the task that I have to implement after. Git was very useful to keep track of my work.

Moreover, I gained a deeper understanding of Statecharts. In fact, my only experience at ENSTA Bretagne with Statechart was through UML, but during my internship I follow some presentation about Statechart and I try to use SCCD which is a simulator of Statechart.

The discovery of a research center

During this internship I discover the operation of a research center. Even if I will work for the DGA[1] which is different of a research center, It was very interesting to discover how it works.

I had the opportunity to see some Master Thesis presentation, and follow an introduction of all MSDL researches. It gave me the possibility to better understand their research on Modeling and why they are important.

Encountered difficulties

During this internship I meet one of the most common problem in software development: the problem of integration. In fact, for me it was very difficult to find a API[2] to do action on UML Designer. In fact, UML Designer is based on Sirius, EMF, and the Eclipse Kernel, so I had to understand their API to interact on the user interface of UML Designer. It was very difficult, because when I wanted to do something I had first to find on which API I had to be connected and then how use it.

An other problem was that the simulator given by Mr Teodorov was stable but not finished. In fact, this simulator has some constraints which block the advanced of new

[1] Direction générale de l'armement
[2] Application Programming Interface

abilities. For example, This simulator doesn't take care about instances. It is a real problem because in OOP[3] there is often object which are instantiated several times. Even if, my plugin was written to have the possibility to show multiple instances, it was not possible to test it on real model.

[3]Object-Oriented programming

Conclusion

I made my project called UML-DSimulator. This project is a plugin for UML Designer to simulate UML Models. This project was written to work with the Ciprian Simulator. Currently, the plugin show on the UML Model a visualization of the simulation, give some debugger tools, and have the ability to take an other simulator. However, this plugin was only tested with the Ciprian Simulator which is not finished, and this simulator imposes many constraints.

So, I did a plugin stable and modular which full fill the initial requirements, but is not adapted for the UML community because of the constraints of the simulator. Furthermore, It is possible to notice that this plugin need some improvement especially in the field of debugging.

Despite this, I am very happy because this internship give me the possibility to learn new technical skills, to overwhelm myself, to overcome many challenges, and to use in a real project managing tools that I learn during my scholarship.

Appendix

APPENDIX **A**

UML Designer

Description

UML Designer is an open-source tool to edit and visualize UML2 models created by the French company: *Obeo*. The project is licensed under the EPL[1]

Figure A.1: UML Designer logo

Utilization

UML Designer is a graphical modeling tool for UML2 as defined by OMG[2]. As you can see on the figure A.2, it permit to create diagram on which ones it is possible to add some elements. The type of the elements proposed depend on the types of the diagram chosen. For example, if you choose a *User case diagram* it is possible to add 'user' component that is impossible in *Class diagram*.

So with graphical action it is possible to create many UML diagram which have transverse elements.

To finish, it is possible to create the code of the application that you have develop from the model.

List of diagram supported

- Packages diagram
- Use case diagram
- Activity diagram

[1] Eclipse public license
[2] Object Management Group[8]

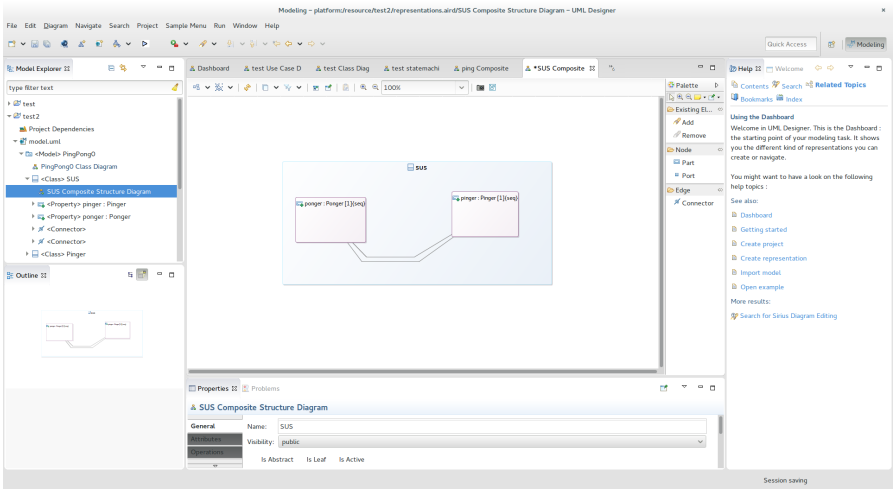


Figure A.2: Screen shot of UML Designer

- Class diagram
- Component diagram
- Composite Structure diagram
- Sequence diagram
- State Machine diagram
- Documentation table
- Use Case cross table
- Package containment diagram
- Profile diagram

Released

Version	Release Date
1.0.0	2012
2.0.0	17 January 2013
2.1.0	1 February 2013
2.2.0	12 April 2013
2.3.0	13 June 2013
2.4.0	13 September 2013
3.0.0	17 January 2014
4.0.0	8 July 2014
4.0.1	5 August 2014
5.0.0	29 May 2015
6.0.0	19 October 2015

Legend:
Latest stable release

Base on

UML Designer is based on a Eclipse and Sirius. It is a UML2 Eclipse plugin.

Sirius

Sirius is an open-source software project of the Eclipse Foundation. Sirius allows to create graphical modeling workbench. It include EMF[3] and GMF[4]. On the figure A.3, it is possible to see the architecture of Sirius.

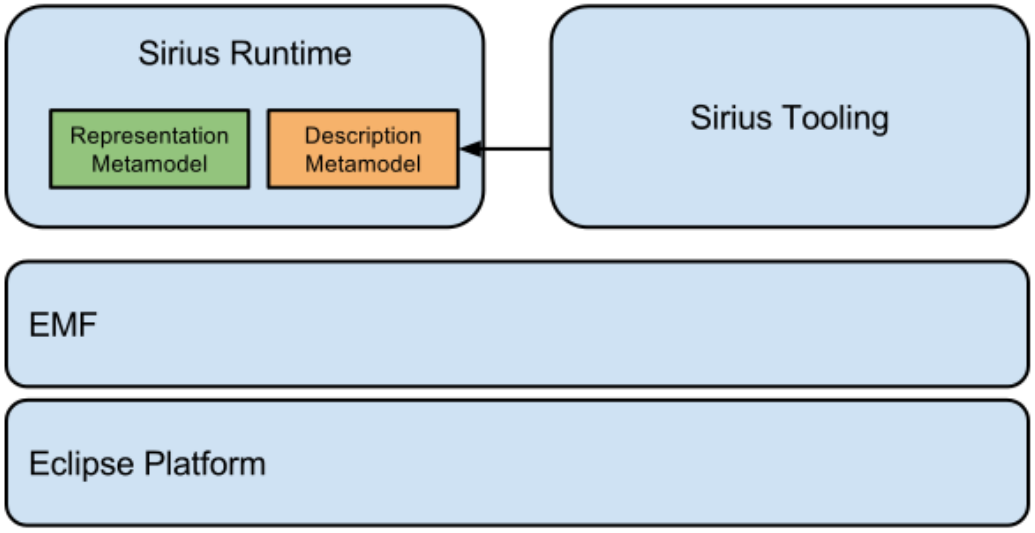


Figure A.3: Sirius architecture[7]

Eclipse

UML Designer is base on Eclipse. The interface is the same as Eclipse. You can notice on figure A.2 that the menu are the same in the both software.

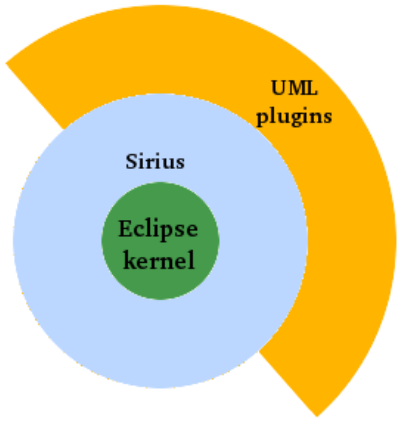


Figure A.4: The UML Designer kernel

[3]Eclipse Modeling Framework
[4]Graphical Modeling Framework

APPENDIX B

Simulator

Description

At the beginning of this project, we had at our disposal the simulator of Mr Teodorov (figure B.1). This simulator have a graphic user interface as you can see on the figure B.1.

It was written in Java and Xtend, and Mr Teodorov used a build automation tool, Gradle.

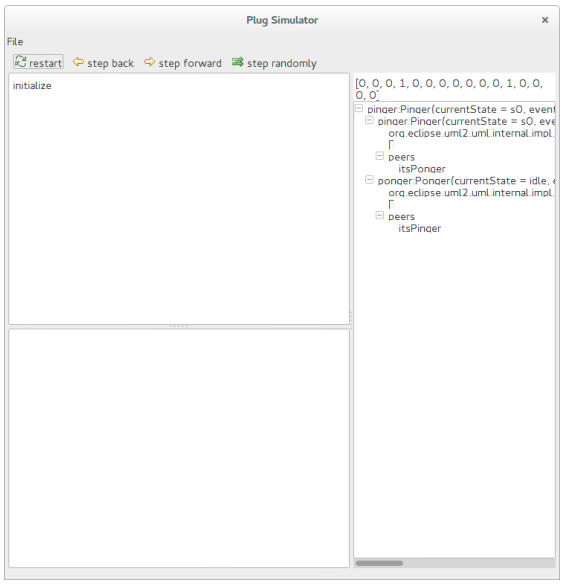


Figure B.1: Mr Teodorov simulator

The simulator is compose on 4 part.

- On the top: some buttons to select an action
- On the top-left-corner: The list of the next step
- On the bottom-left-corner: The State Machine associated to the Current State.
- On the right: A visualization of the Statechart

APPENDIX C

Inter-process communication

We are only interesting in communication which permit to exchange data and synchronize them. There is a short presentation of all other communication considered.

Message queue

Advantages	Drawback
Work on most operating systems	The implementation depends on the OS
One part don't need to know the other	
No serialization/deserialization	
Very fast	

The main problem is that the implementation depends on the OS, that is why we don't chose this communication.

Pipe

Advantages	Drawback
	All POSIX systems, Windows
	The implementation depends on the OS
	Windows: Only one-way pipes (Anonymous pipe and Named pipe[2])

This drawback is to penalizing, for the multiplatform constraint.

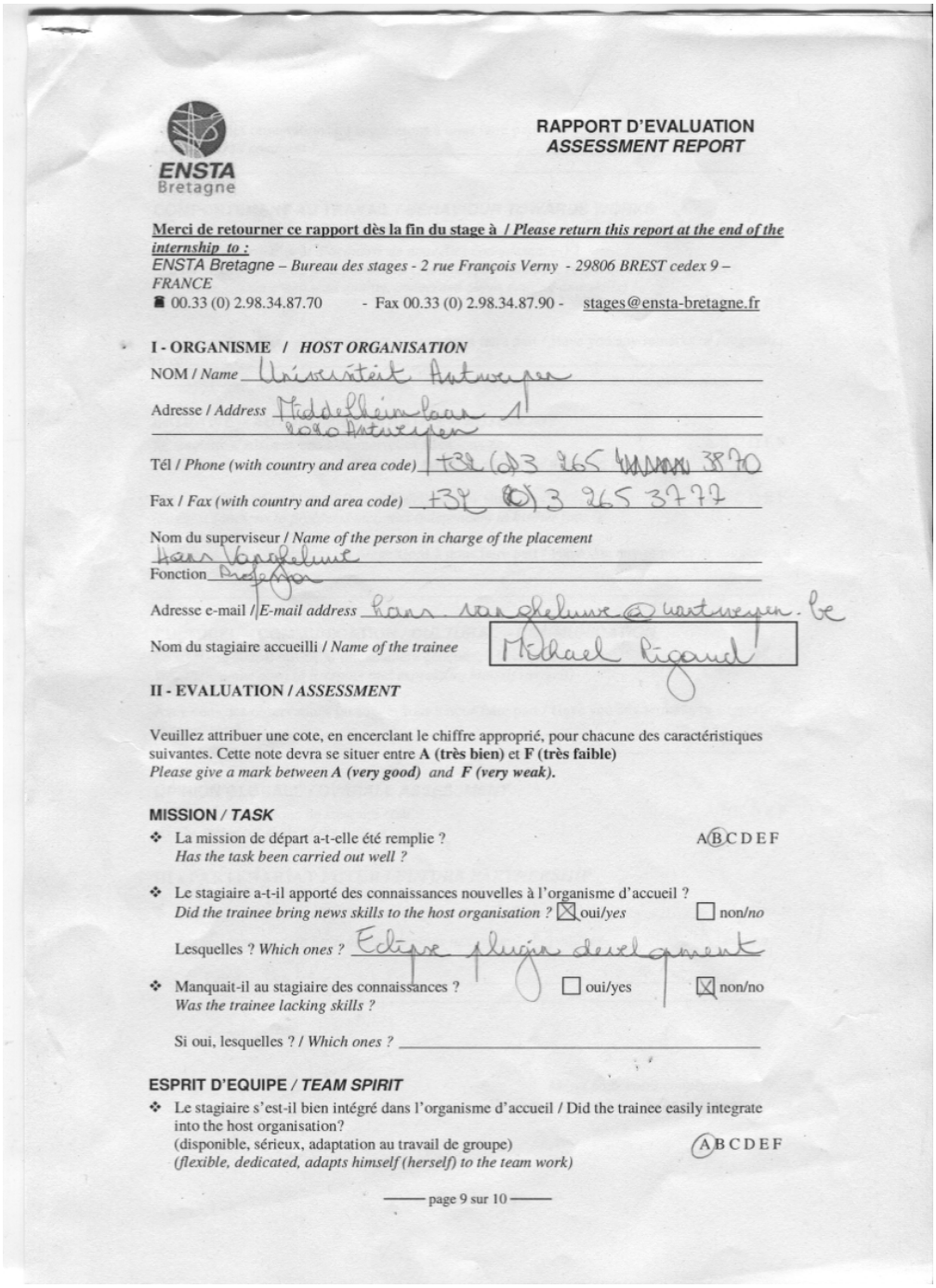
Message parsing

Advantages	Drawback
	Used only in RPC, RMI, and MPI paradigms, Java RMI, CORBA, DDS, MSMQ, MailSlots, QNX, others
	Slow and complex

This drawback is to penalizing, for the multiplatform constraint.

APPENDIX **D**

Assessment report



ENSTA Bretagne

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport dès la fin du stage à ***/ Please return this report at the end of the internship to :***
ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 - Fax 00.33 (0) 2.98.34.87.90 - stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Universiteit Antwerpen

Adresse / *Address* Middelheimlaan 1 2020 Antwerpen

Tél / *Phone (with country and area code)* +32 (0)3 265 [illegible] 3870

Fax / *Fax (with country and area code)* +32 (0)3 265 3777

Nom du superviseur / *Name of the person in charge of the placement*
Hans Vangheluwe

Fonction Professor

Adresse e-mail / *E-mail address* hans.vangheluwe@uantwerpen.be

Nom du stagiaire accueilli / *Name of the trainee* Mikael Rigaud

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une cote, en encerclant le chiffre approprié, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please give a mark between **A (very good)** and **F (very weak)**.*

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? *Has the task been carried out well ?* (A) B C D E F

- Le stagiaire a-t-il apporté des connaissances nouvelles à l'organisme d'accueil ? *Did the trainee bring news skills to the host organisation ?* ☒ oui/*yes* ☐ non/*no*

  Lesquelles ? *Which ones ?* Eclipse plugin development

- Manquait-il au stagiaire des connaissances ? *Was the trainee lacking skills ?* ☐ oui/yes ☒ non/no

  Si oui, lesquelles ? / *Which ones ?* _____

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil / Did the trainee easily integrate into the host organisation?
  (disponible, sérieux, adaptation au travail de groupe)
  (flexible, dedicated, adapts himself (herself) to the team work) (A) B C D E F

page 9 sur 10

Avez vous des observations ou suggestions à nous faire part / *Do you have any remarks or suggestions to comment ?*_____

_____

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORKS*

Le comportement du stagiaire était-il conforme à vos attentes *(Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances)* ?
Did the trainee come up to expectations (*Punctual, methodical , responsive to management instructions, concerned with quality, concerned about gaining new skills)* ?

(A) B C D E F

Avez vous des observations ou suggestions à nous faire part / Have you any remarks or suggestions to comment ?_____

_____

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'adaptait vite à de nouvelles situations ? A B C D E F
(Proposer des solutions aux problèmes rencontrés, autonome dans son travail,..)

Did the trainee adapt himself (herself) well to new situations ? A (B) C D E F
(Suggest solutions to problems met, was independent in his/her job…)

Avez vous des observations ou suggestions à nous faire part / Have you any remarks or suggestions to comment ?_____

_____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication A (B) C D E F
Was the trainee open to listening and expressing himself (herself)

Avez vous des observations ou suggestions à nous faire part / Have you any remarks or suggestions to comment ?_____

_____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : A (B) C D E F
The technical skills of the trainee were :

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ? depending on the topic
Are you prepared to host another trainee next year ? ☒ oui/*yes* ☐ non/*no*

Fait à Antwerpen ,le 7/9/2016 In
_____ ,on _____

Signature

Merci pour votre coopération
We thank you very much for your cooperation

List of Figures

List of Tables

Bibliography

[1] apache. xmi2scxml, 2009. https://github.com/apache/commons-scxml/tree/master/extras.

[2] Microsoft corporation. About pipes. https://msdn.microsoft.com/en-us/library/windows/desktop/aa365137(v=vs.85).aspx.

[3] Joeri Exelmans. Configurable semantics in the sccd statechart. Master's thesis, University of Antwerpen, 2014.

[4] Glenn De JONGHE. A visual modelling environment for statechart and class diagrams in unity. Master's thesis, University of Antwerpen, 2015.

[5] MSDL. Mdsl web site. http://msdl.cs.mcgill.ca/.

[6] Obeo. Contribute developer guide.

[7] Eclipse Obeo. Sirius documentation. https://www.eclipse.org/sirius/.

[8] OMG. Object management group. http://www.omg.org/.

[9] Dan Radigan. Kanban. https://www.atlassian.com/agile/kanban.

[10] stleary. Json-java. https://github.com/stleary/JSON-java.

[11] Wikipédia. Observer pattern. https://en.wikipedia.org/wiki/Observer_pattern.