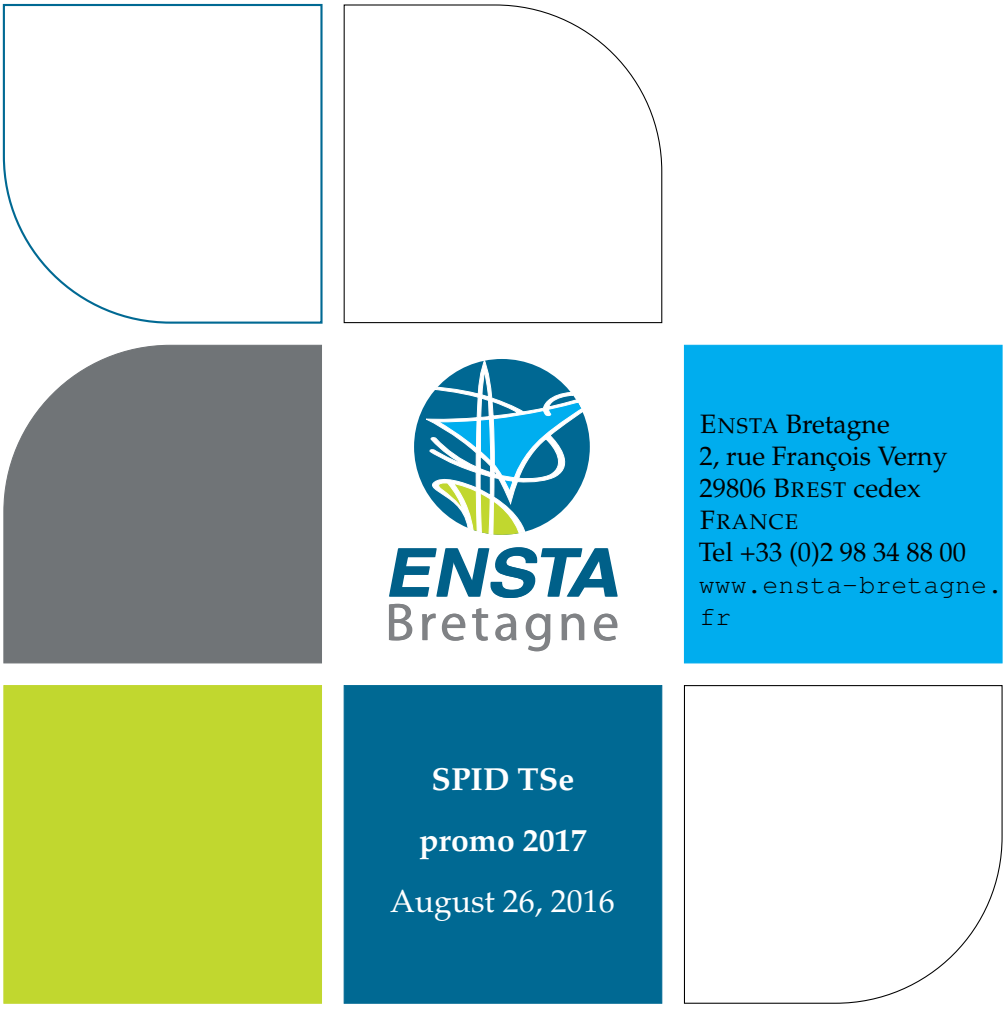

ENSTA Bretagne
2, rue François Verny
29806 BREST cedex
FRANCE
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

SPID TSe

promo 2017

August 26, 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 add a simulator to UML Designer. This plugin has been entirely realize during this internship, but it is based on a simulator developed by Ciprian Teodorov.

[1] under GNU GPL license

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 give 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 laboratory of the ANSYMO department for their help in offering me the resources in running the program, and their welcome.

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 Model and simulate them. The two more user-friendly tools which permit that are: Rhapsody and Papyrus.

Papyrus use Moka to simulate UML Model and it was not well adapted for his lesson, so he choose 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 student said that you won't use this software outside the lesson.

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 aims, the problems and the solutions bring during this internship. In the beginning we will present the project and the tools at the disposal at the beginning of the internship, then we will show our solution, and to finish we are going to underline the objectives after the end of this project.

Part I

Presentation of the context

CHAPTER 1

Presentation of the University

MSDL

Figure 1.1: 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.[1]

My position in the university

My stage chief was Prof. Hans Vangheluwe[1]. But, because he is always busy and not always in the university for professional reason, I was attached to Simon Van Mierlo[2] for this internship. He is a PhD student at the University of Antwerp. He works on the simulator of Statechart SCCD and he create a debugger for this simulator.

Simon has been my tutor because first of all my work was very similar at a debugger interface and because Prof. Vangheluwe expected that I use SCCD as simulator and compare it with the simulator of Mr. Teodorov.

I work during my internship in the office of Simon and Yentl Van Tendeloo[3]

[1]Prof. Hans Vangheluwe: `http://msdl.cs.mcgill.ca/people/hv/`
[2]Simon Van Mierlo: `http://msdl.cs.mcgill.ca/people/simonvm/`
[3]Yentl Van Tendeloo: `http://msdl.cs.mcgill.ca/people/yentl/`

Part II

Presentation of the project

CHAPTER 2

Issues

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

Short term issue

In short term, this plugin should permit at Mr. Champeau and Mr. Teodorov to use during their classroom an free alternative and multi-platform of Rhapsody: UML Designer. In this way, every student can install on their own computer the tool use in classroom.

Advantages of UML Designer compare to Rhapsody: free, open source, work on Windows, Linux, and Apple.

Long term issue

Because this plugin is open source and downloadable on Github, it will be use by everybody. My hope is this plugin will be improve by the community, student, teacher, *etc*... and became a serious alternative to Rhapsody and Papyrus.

Figure 2.1: Rational Rhapsody

Figure 2.2: Papyrus

CHAPTER 3

The plans of the project

Definition of my project

After some interview with Mr. Champeau, I define the table of requirements (tabular 3.1). This table contains all requirements define by the client to respect the issues of this project define before.

FS means service function and *C* means constraints.

Table of requirements		
Number	Type of Designation	Designation
FS1	UI	The plugin need to represent the visualisation in the UML Model
FS2	UML Designer	Find how to integrate code, and document it to future implementation
FS3	Simulator	Give the choice of the simulator. The user need to have the choice to change the simulator if he want.
FS4	UML Designer	The plugin need to be adapted for the Ciprian simulator. To begin, the plugin could looks like the UI of the Ciprian Simulator but on UML Designer
C1	License	Open Source
C2	Compatibility	The plugin need to be multiplatform. Works on Linux, Windows, and Apple.
C3	Documentation	Produce documentation, code readable, modularity, etc. In that way, this project could be improved during another internship.

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 with the outside world that the client expected.

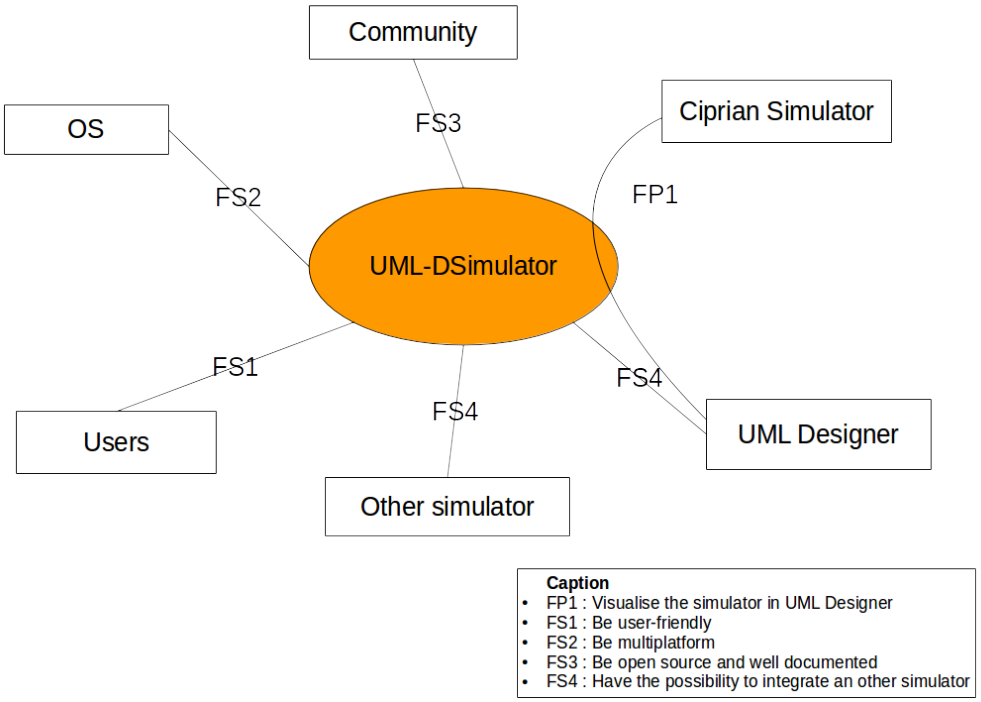

Figure 3.1: Octopus diagram

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 some feature.
2. Understand how work the Ciprian simulator.
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 finish 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 do some improvement after the end of this internship.
6. Try an other simulator and compare its performance with the Ciprian simulator.

Tools at my disposal

It is a short description of UML Designer and the Ciprian Simulator. A further description can be found in the annex.

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 know and documented.

Simulator

Then, Mr Ciprian Teodorov, one of my professor, has developed a simulator for UML Model. This simulator needed to be improved, but it composed a good beginning for this project.

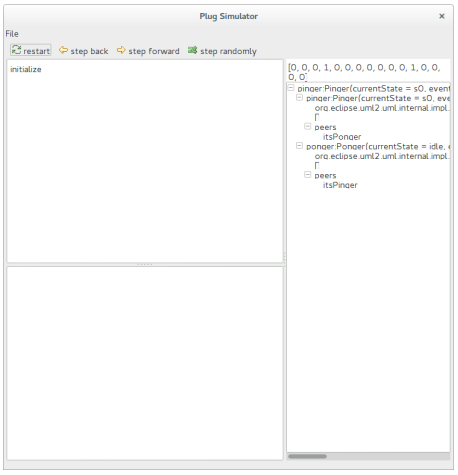

Figure 3.3: Mr Teodorov simulator

Part III

Results of the internship

CHAPTER 4

Technical choice

During this project we made a lot of technical choice. We will explain which choice we take and why.

Type of communication

First of all, we had to find the best way to integrate the simulator in UML Designer. Because we want to keep the possibility to change the simulator we have decided to put it outside. In this way, we had the possibility to change the simulator without changing everything in the plugin.

Moreover, we have to find the best way to realize the communication enter the simulator and the plugin. We decided to chose a socket communication, and send message only formatted as json object.

Socket

This next table explain advantages and drawback of socket. We chose socket instead of other type of communication because it have a better ratio of Advantages/Drawback.

Advantages	**Drawback**
Work with every simulator type (python, java, ...)	Message need to be formatted
	Not very fast

Type of message

Now we have chosen that we will use socket to communicate enter the plugin and the simulator, we have too choose which type of object we will send by this socket.

In the same way, we list the type of message that we 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, we chose to use Json. Moreover, Json are send like String but with a formatted type.

To do that we use a library which permit to manipulate Json object in Java. We found it on Github [5].

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 construct. There is a plugin incorporate in UML Designer and a communication layer for the simulator. The plugin communicate to the communication layer with socket and json. The plugin receive the data send from the simulator, analyze them, display them, and send instruction to the simulator.

The figure 4.1 resume its.

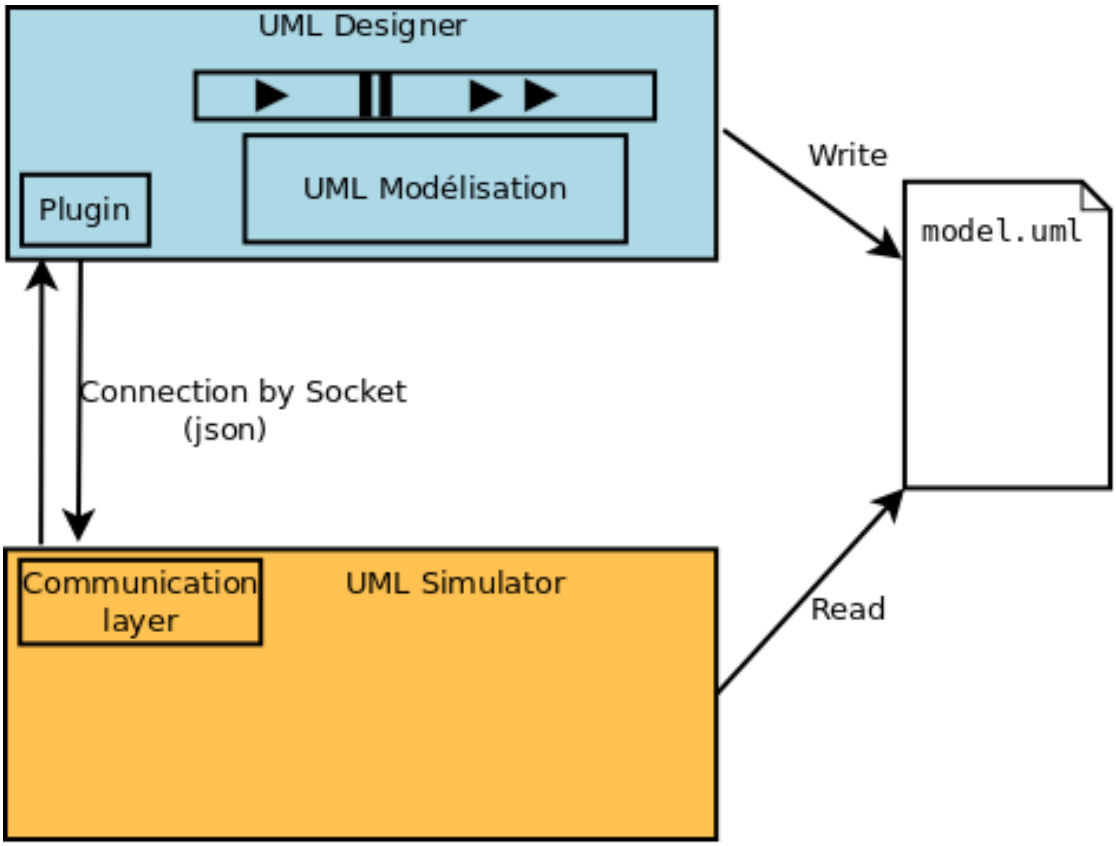

Figure 4.1: overview of the project

CHAPTER 5

Results

CHAPTER 6

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 to don't tests them. However, I have tested all other functions used in the plugin.

To do this unit tests, I used junit and a eclipse feature EclEmma which permit to see the coverage of code during unit tests. On the figure 6.1, 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[5], so I didn't write unit tests for this package.

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 6.1: Coverage view of my project

Integration tests

I also did some integration tests to verify that I respect one of my constraint, be installable in all platform (Windows, Linux, Apple).

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.

[1] It is the usual value of acceptable coverage

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 6.3), and a Kali virtual machine based on Debian (figure 6.2).

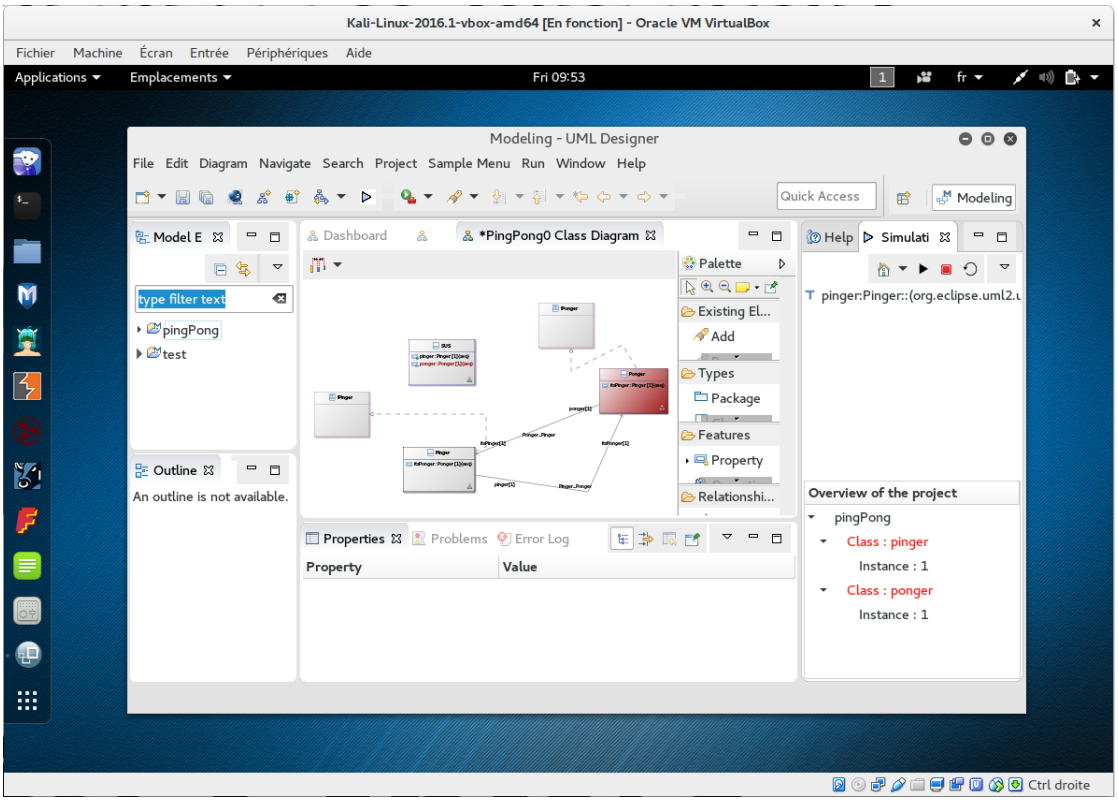

Figure 6.2: Screenshot of the kali virtual machine

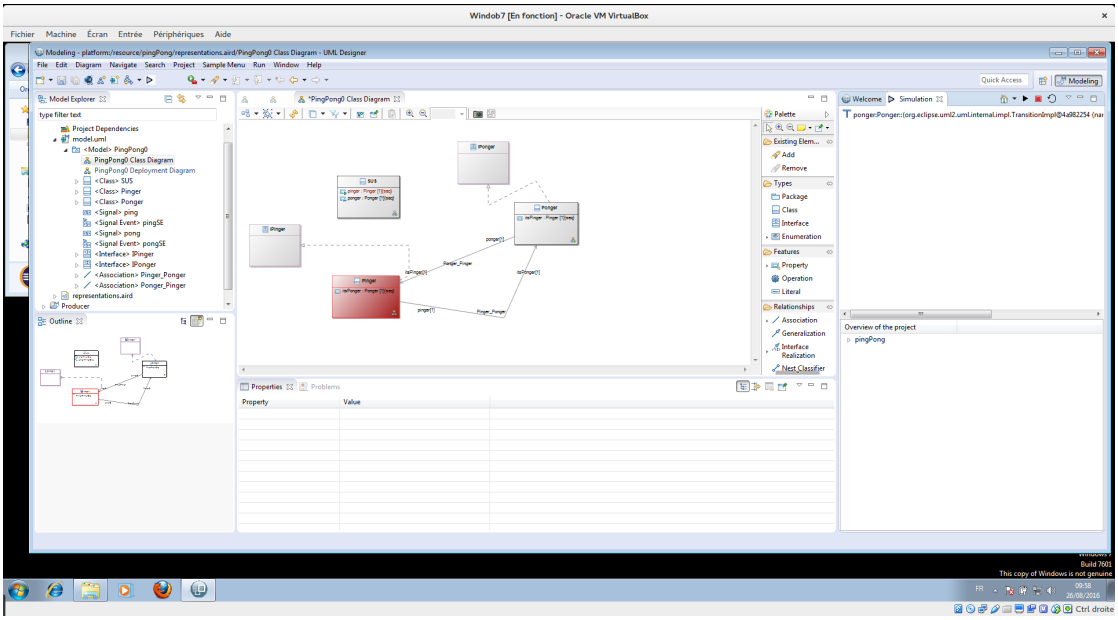

Figure 6.3: screenshot of the Windows virtual machine

Part IV

Contribution of this internship for my professional project

CHAPTER 7

Contribution

Conclusion

Annex

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[4]

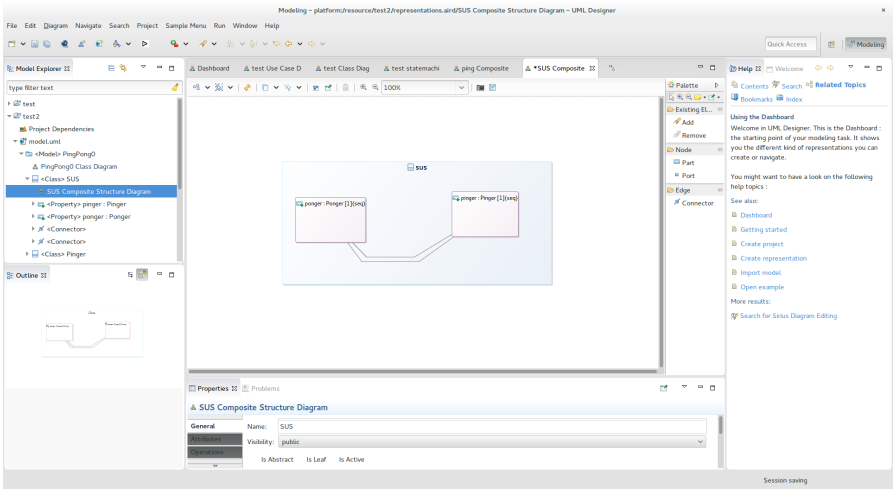

Figure A.2: Screen shot of UML Designer

- 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.

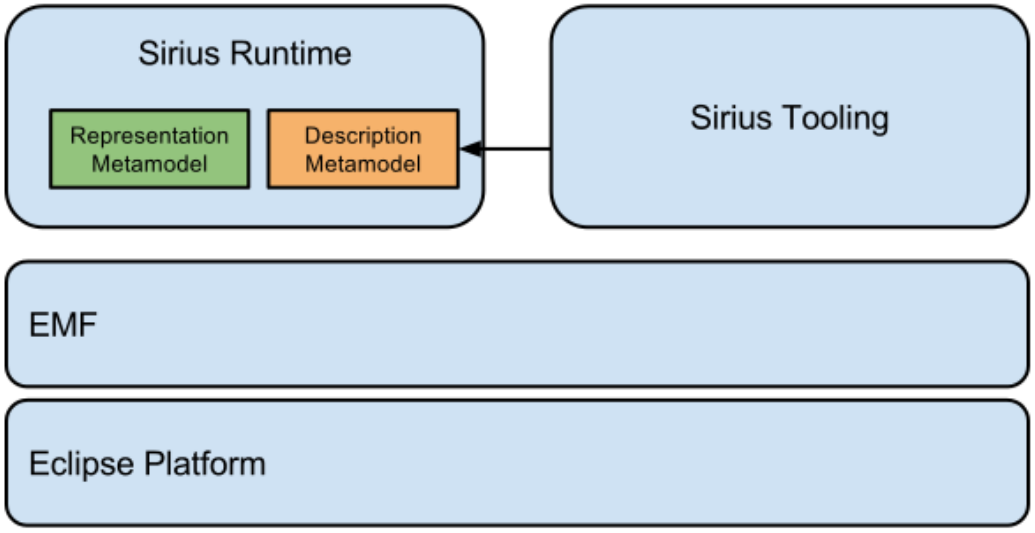

Figure A.3: Sirius architecture[3]

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.

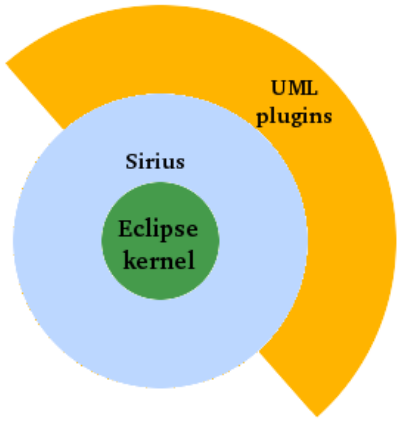

Figure A.4: The UML Designer kernel

[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.

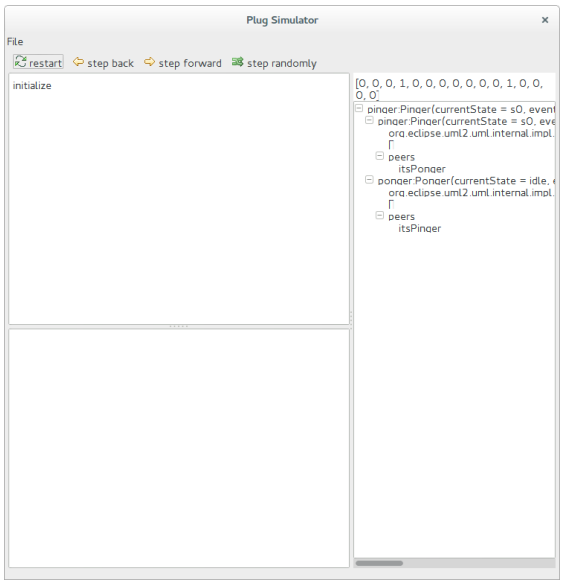

Figure B.1: Mr Teodorov simulator

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

Specificity of the uml file

This simulator simulate a uml file. The uml file need to have a particular architecture.

UML Designer to save the uml project use 2 files. The first is named “model.uml” and the second is named “representation.aird”.

To work, the simulator need the *model.uml* file. Moreover, this file need to contain some specifics feature. It need a class **SUS** which contain the declaration of all other

classes and all other classes need to have a State Machine diagram associated. You can see on the figure B.2, that all classes need to have their own State Machine diagrams.

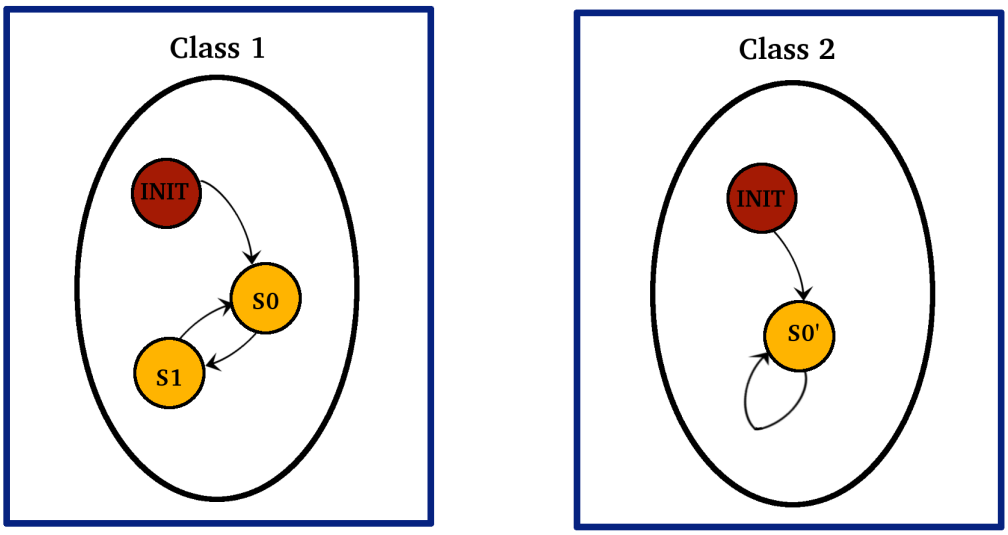

Figure B.2: representation of the most important elements of the simulator

APPENDIX C

Organisation of the work

Calendar

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				-

During the week 10, the University was closed, that is why it is a trivialized week.

Tools use for the project

The Framaboard application:

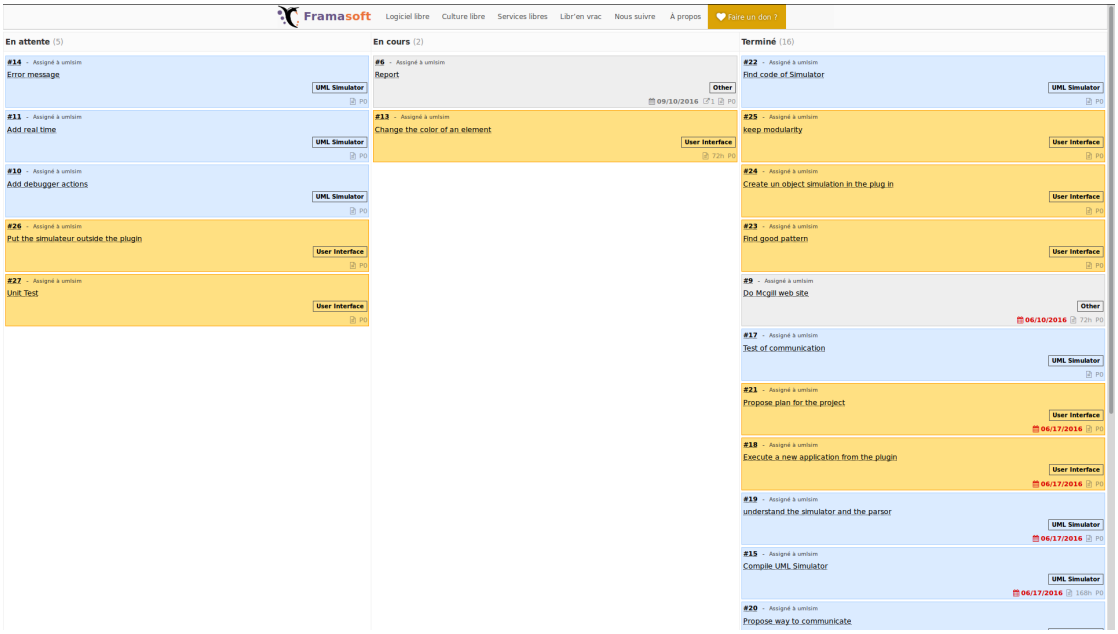

Figure C.1: Screen shot of the framaboard

The web site of MSDL researcher:

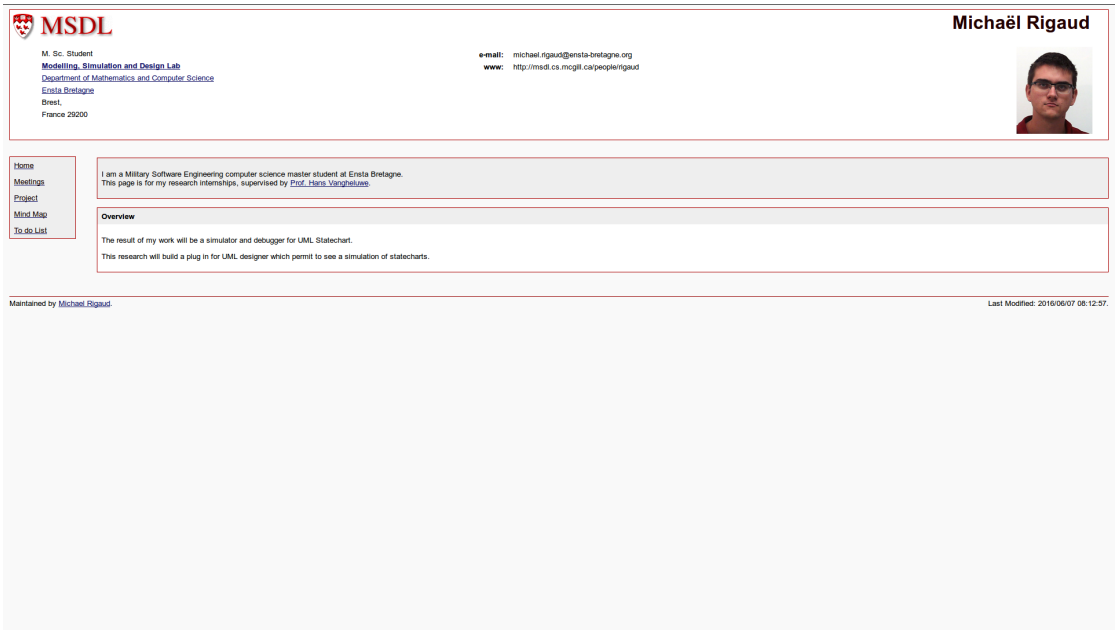

Figure C.2: MSDL web site

The MSDL git repository

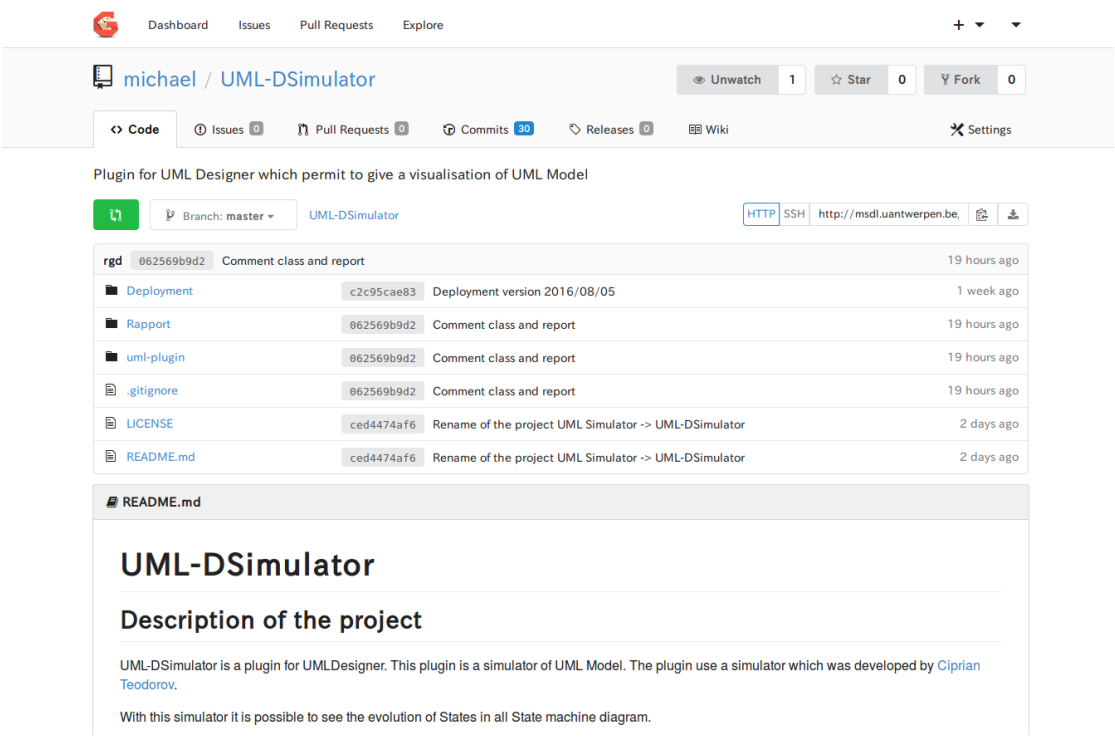

Figure C.3: git repository

List of Figures

Bibliography

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

[2] Obeo. Contribute developer guide.

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

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

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