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

September 2, 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 produce 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 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 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 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 choice for preserve the organization of the project. After, I will show the technical choice given and the result of the technical part. To finish, I will highlight the contribution of this internship for my professional project.

Part I

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

Simon was chosen to be 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.

During my internship I work in the office of Simon and Yentl Van Tendeloo. But I was in relation with all members of MSDL.[2]

Position of MSDL

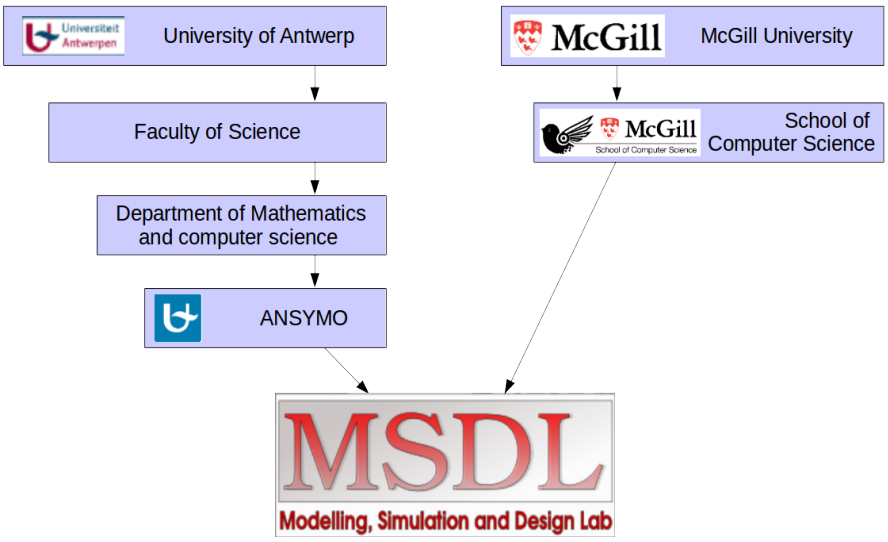


Figure 1.1: Position of MSDL

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

[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`

Montreal, Canada. 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.»[2]

So MSDL is a laboratory of Modeling. It is an important field of computer science because that could permit to create an engineering language understandable by everybody because it is based on graphical diagram. That also permit to create software without programming, just with logical diagram. I had the opportunity during my internship to participated in a presentation of all research subject. The figure 1.3 is a not exhaustive list of all research topics discussed in MSDL. The part the most in connection with my subject was *Simulation* and *Analysis, Validation, Verification, Testing, and Accreditation* for obvious reasons.

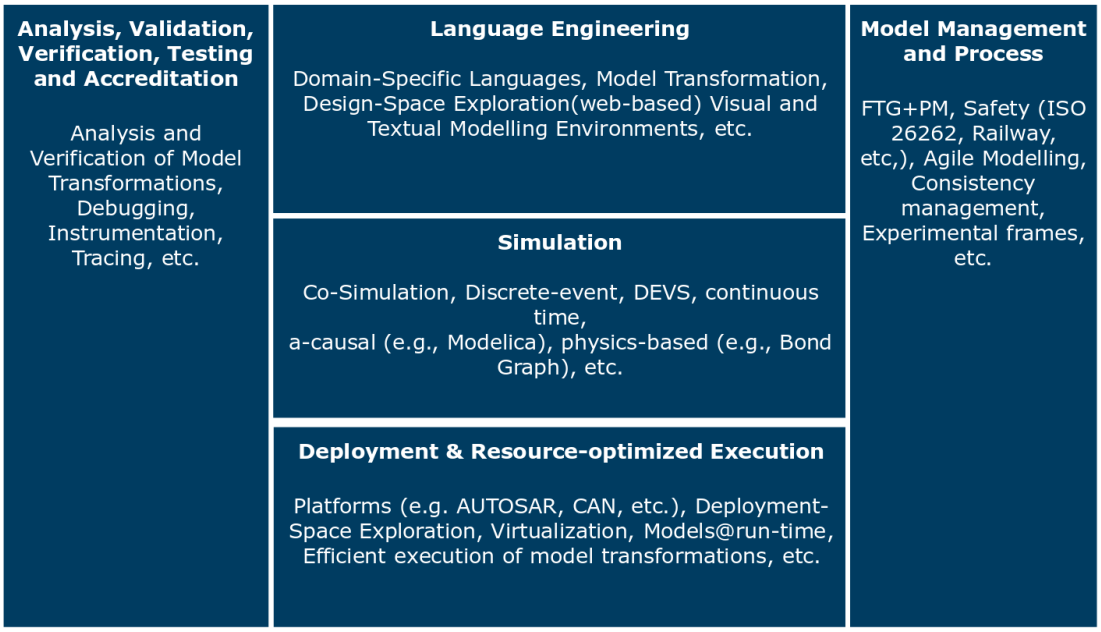


Figure 1.3: Research topics

[3]Only departments which belongs msdl were presented

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 UML Designer as an free alternative and multi-platform of Rhapsody with all their interesting feature. In this way, every student could 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.

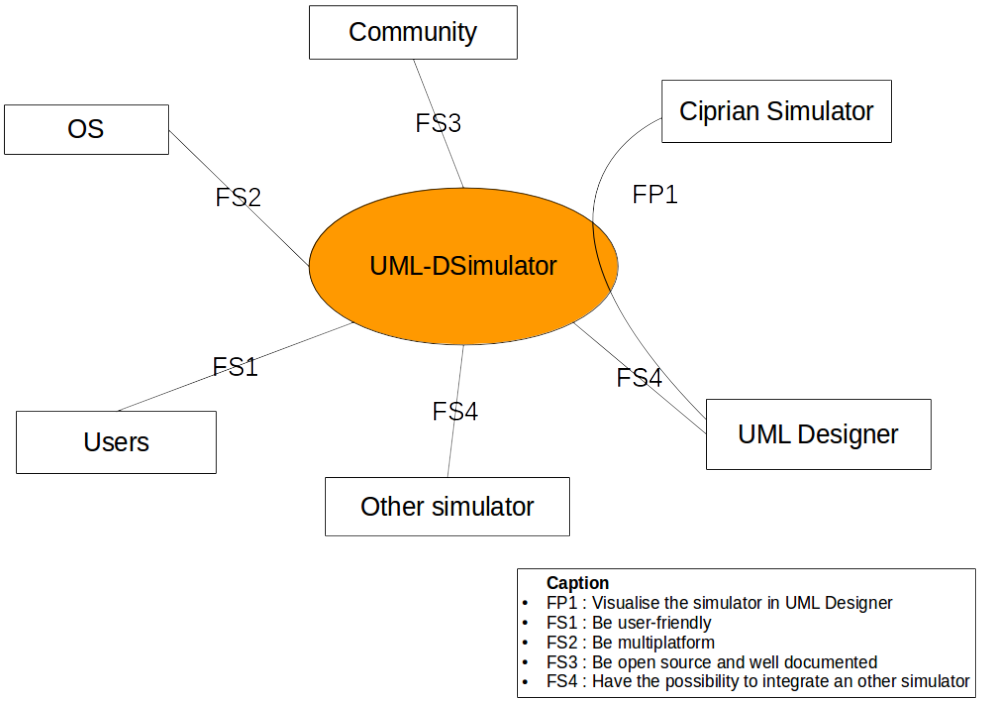


Figure 3.1: Octopus diagram

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.

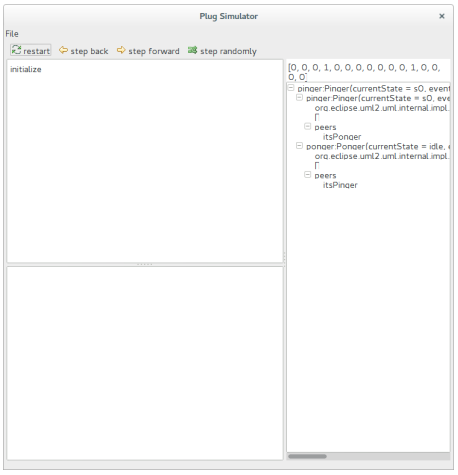


Figure 3.3: Mr Teodorov simulator

CHAPTER 4

Organization

At the beginning of this internship I made some choice in the organization of my work.

Calendar

First of all I made a calendar. This calendar give me a good overview of the time that I am allow 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, that is why it is a trivialized week.

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 every week the work done during this week and the work planned for the next week. I put as much as possible a short video to present my improvement.

I also did a video conference with Mr Champeau and Mr Teodorov the 1th July. This video conference was very useful to know after 1 month if I take the good way and to clear some important point.

Tools use for the project

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

I employ Framaboard. It is 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.

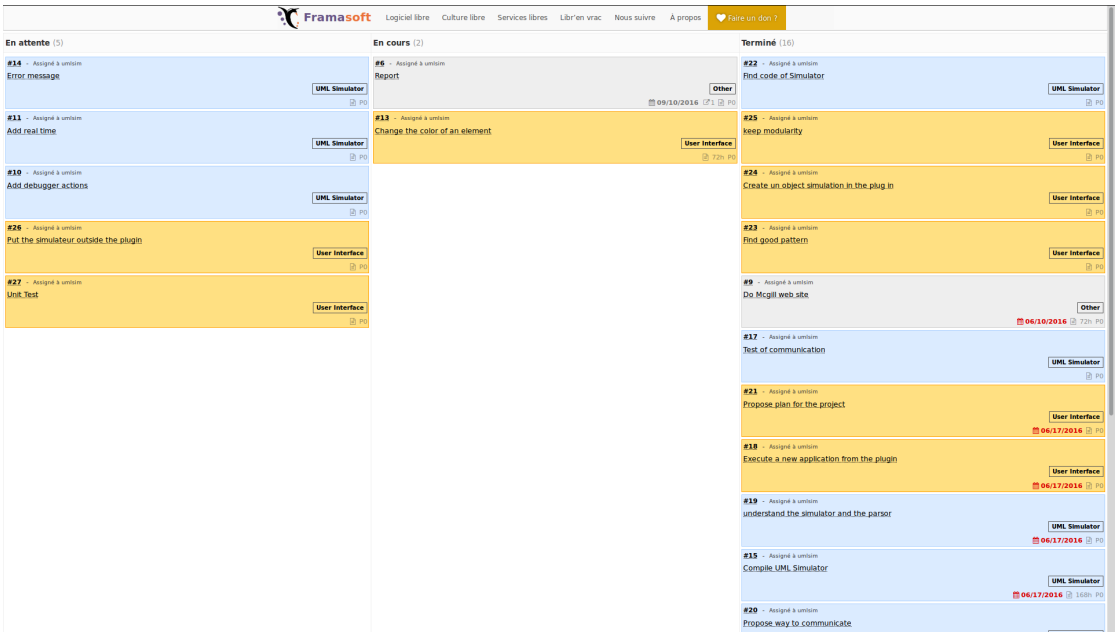


Figure 4.1: Screen shot of the framaboard

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.

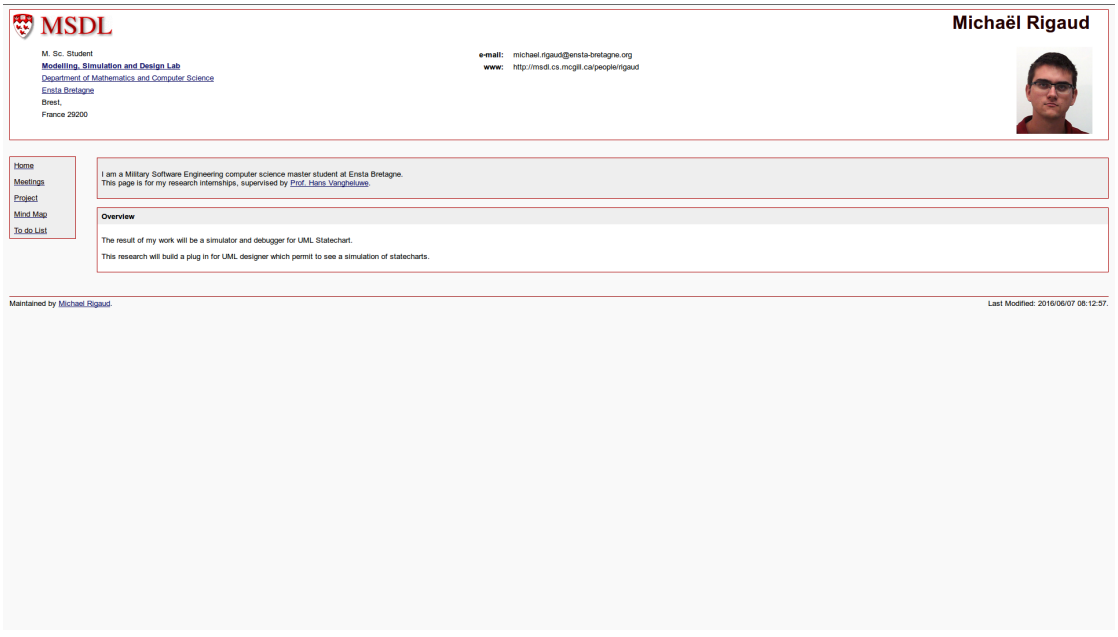


Figure 4.2: MSDL web site

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 breakdown of my computer. The figure 4.3 show my git repository.

[1]Kanban is a method for managing knowledge used by software teams practicing agile software development.[6]

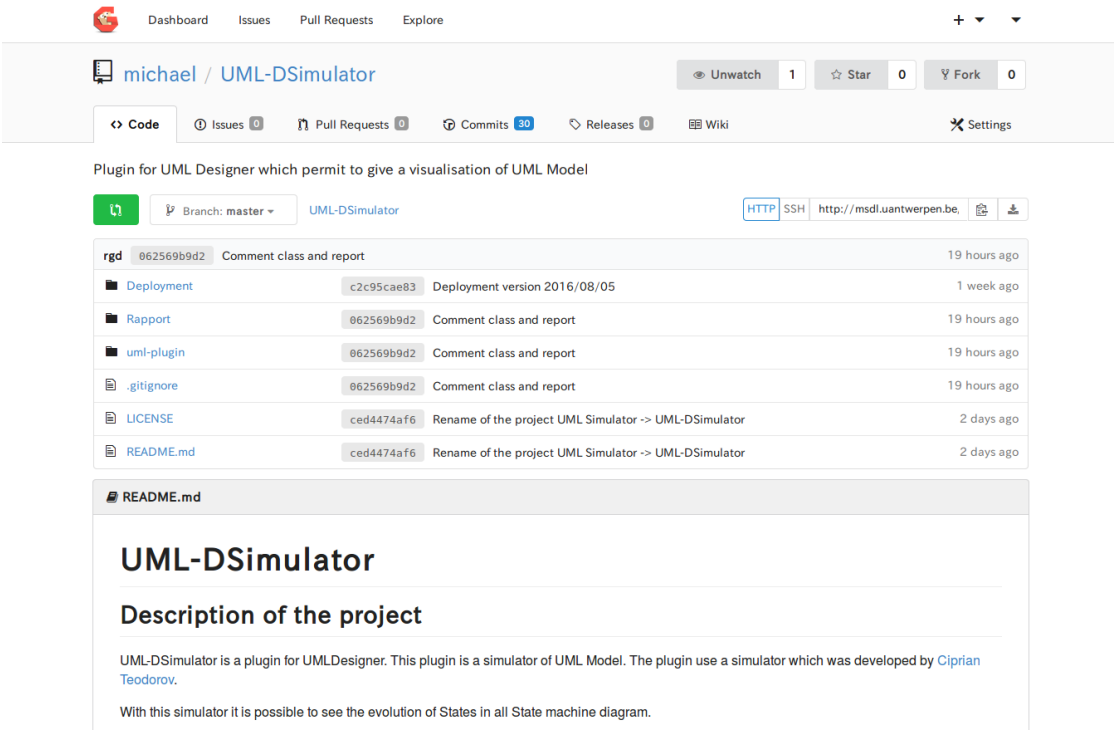


Figure 4.3: git repository

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 have to find the best way to realize the communication enter the simulator and the plugin. After list all type of communication enter process with their advantages and their drawback, I decided to chose a socket communication.

Socket

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

Advantages	**Drawback**
Work with every language (python, java, ...)	Message need to be formatted
Allow communications enter process which don't use the same language	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 exist 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 are send 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 [7].

Our json object are constructed like this:

plugin → simulator

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

simulator → 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 5.1 resume its. However, by default the simulator and the communication layer are in started by the plugin. So by default the simulator and the communication

layer are in UML Designer but it is possible to put them outside because after the start of their thread they communicate only by socket.

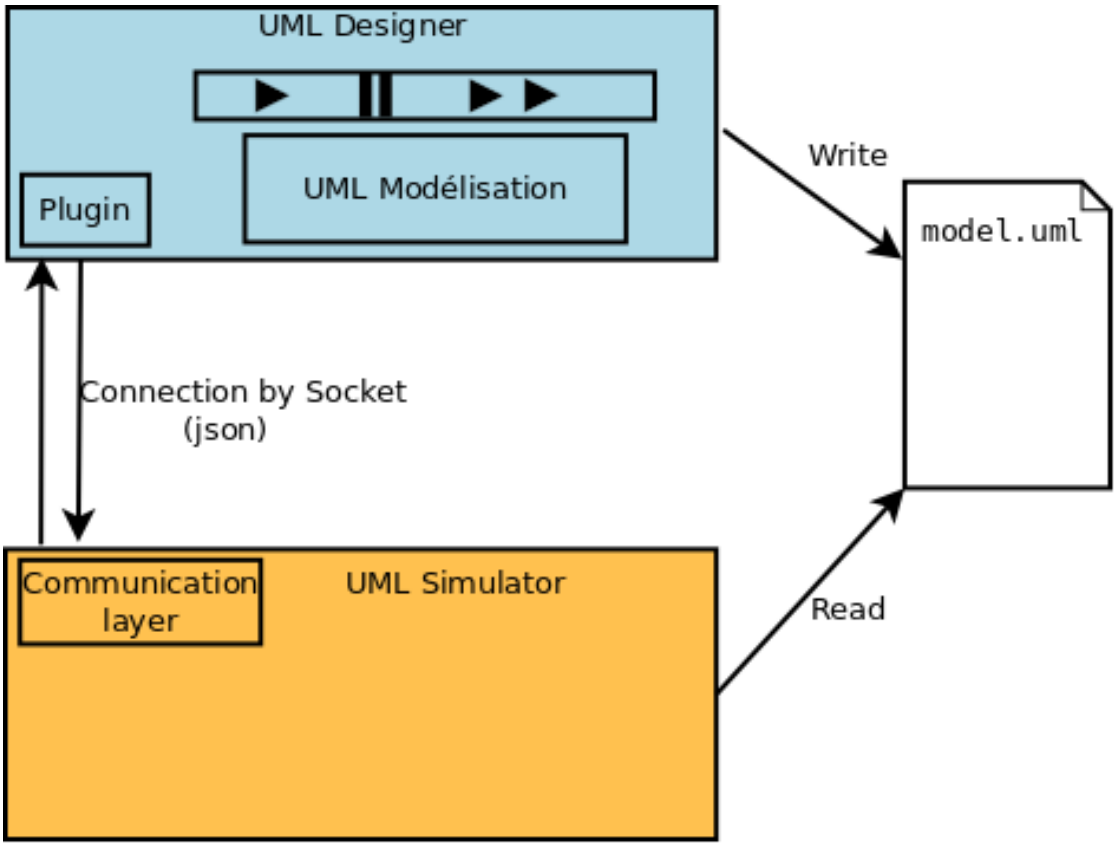


Figure 5.1: overview of the project

CHAPTER 6

The simulator

First of all, I worked on the simulator to know specificities of the simulator and realize the layer of communication.

Specificity of the UML Model

The Ciprian simulator simulate 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.

To work, the simulator need the *model.uml* file. Moreover, this file need to contain some specifics features. 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, as you can see on the figure 6.1.

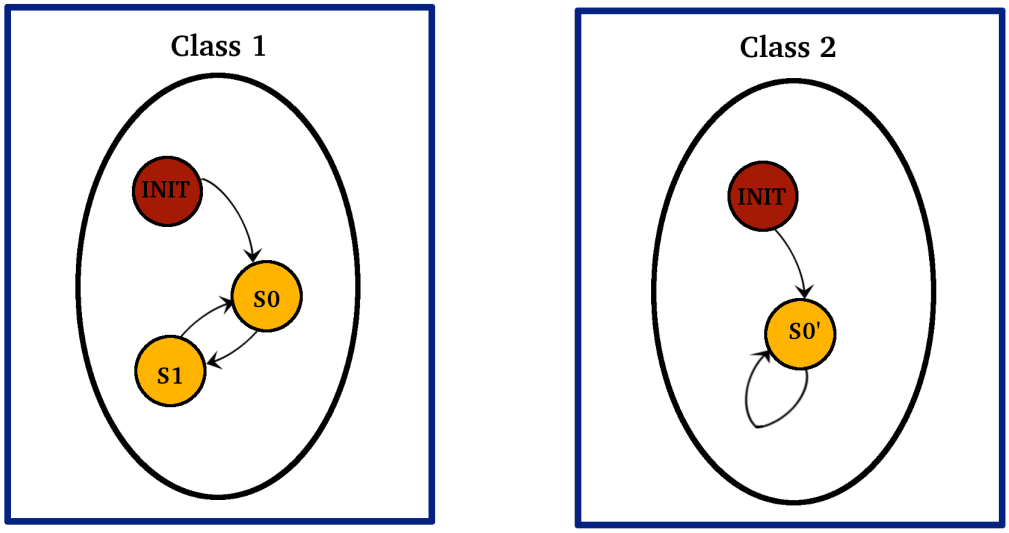


Figure 6.1: representation of the most important elements of the simulator

Additions to Teodorov Simulator

Afterwards, I made some research on the code of the simulator. I realize that the simulator was not develop to communicate and notify any changing. So I change 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.» [8]

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

Communication

The communication Layer was written in Java. In the chapter before, we have decided to put the simulator outside. However, during the implementation we find that is was not very simple to always launch a simulator as a separate software. So I put the simulator and its layer in a new thread inside the plugin. However, the main class of the plugin initiate 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. Even if the plugin start the simulator thread, we can consider that the simulator is outside because there is no exchange enter this thread without socket.

Moreover, I did some test where the thread is initiate by an other software, and that's work. So for the rest of this report we consider that the simulator is outside the plugin because the comportment is the same, but the default simulator is in fact initiate by the plugin.

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.

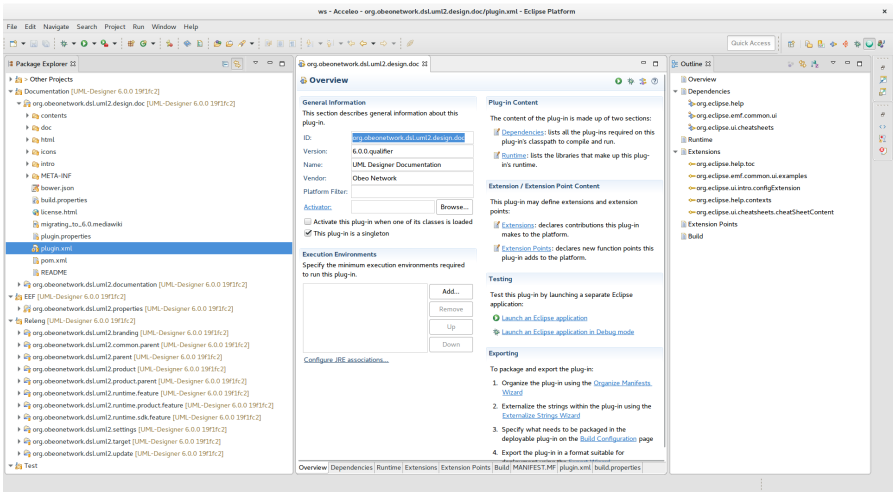


Figure 7.1: Eclipse environment

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.

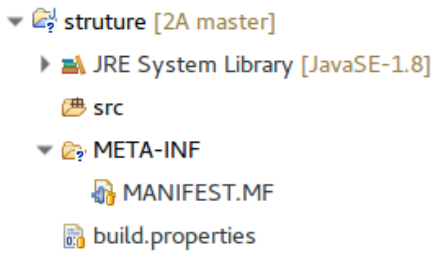


Figure 7.2: Structure of an eclipse plugin

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

General presentation

During the development of the plugin, I tried to separate the 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 the 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 receive.

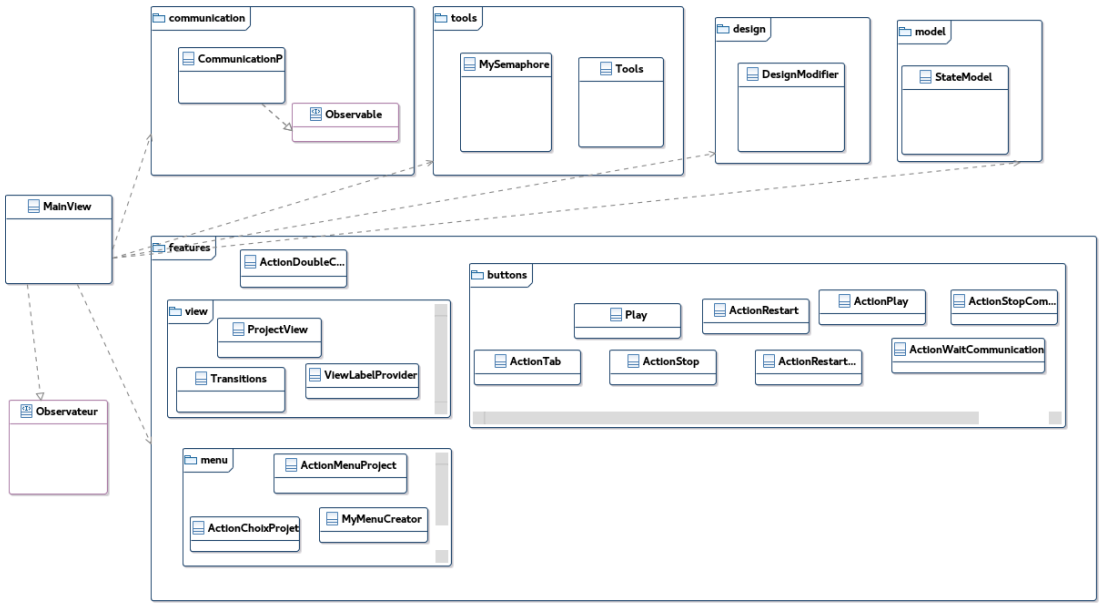


Figure 7.3: plugin class diagram

Functionality implemented

During this project I had time to implements 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 it is a tree of the project

view. The tree has at the top the name of the project simulated, then the name of all class, end to finish the name of all instances.

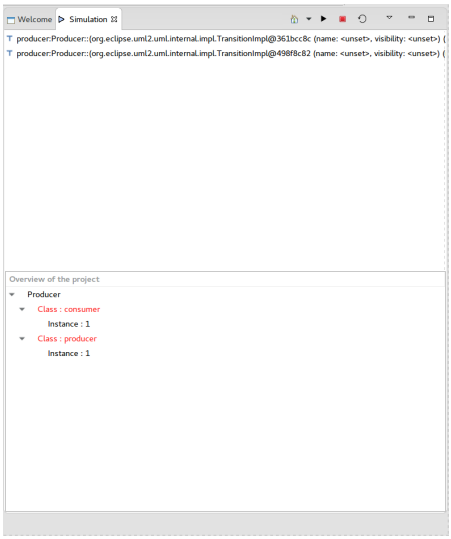


Figure 7.4: result in UML Designer

If the user click on a transition this transition is selected as the next transition, and if he click on an instance of a class it is chosen as the visible instance of the model (useful only if there is multiple instance for a same class).

On the top, it is possible to see that there is many button. These buttons create some action:

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

To finish, 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.

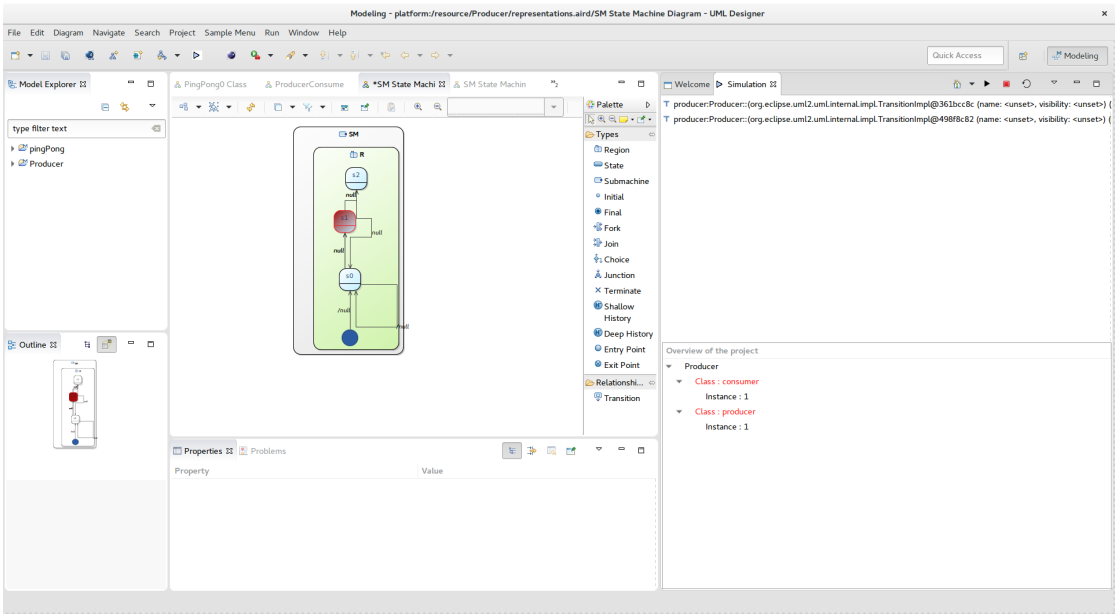


Figure 7.5: result of a simulation

CHAPTER 8

SCCD

In this chapter, I will talk about SCCD. In the MSDL laboratory, they use SCCD to simulate Statechart. After a presentation of my project in July, they suggested me to try their simulator and compare it to the Ciprian simulator. There is more explanation about SCCD tools in annex page ???.

Transformer

As we see before, the project has been written to have a ability to change the simulator. However, the model need 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 have 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.

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 prove of contest.

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.

You can see on the figure 8.1 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.

```
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 8.1: pingpong simulation on SCCD

CHAPTER 9

Tests

Unit tests

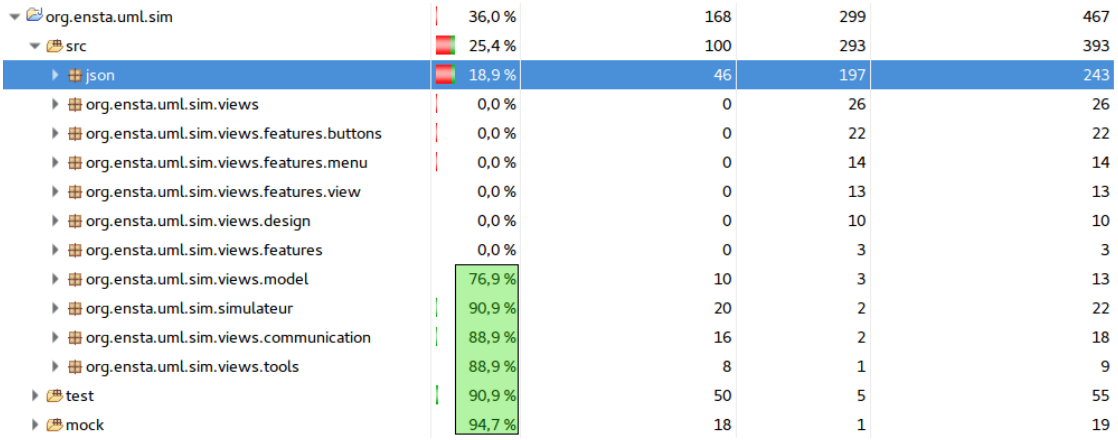
During this project I did some unit tests to preserve the code during the development. But, I had a lot of difficulties to tests user interface features, so I chose to don't tests them. However, I have tested all other functions used in the plugin.

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 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[7], 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 9.1: 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 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 virtual machine on the internet.

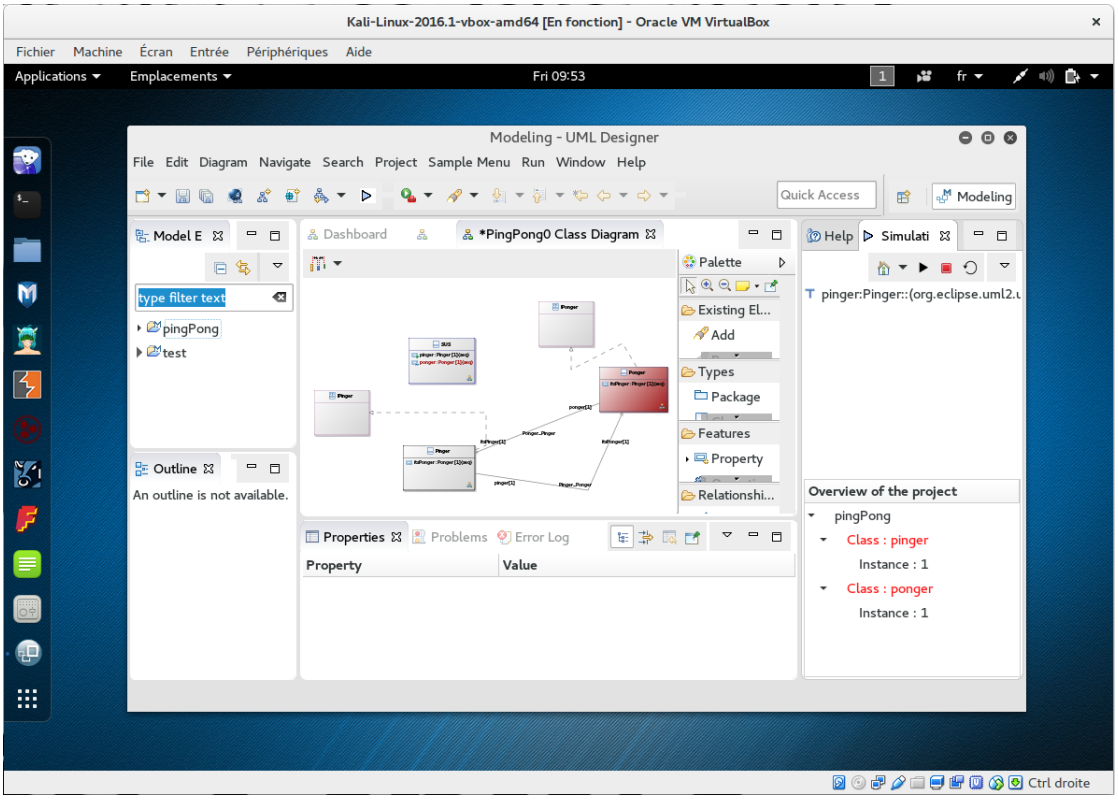


Figure 9.2: Screenshot of the kali virtual machine

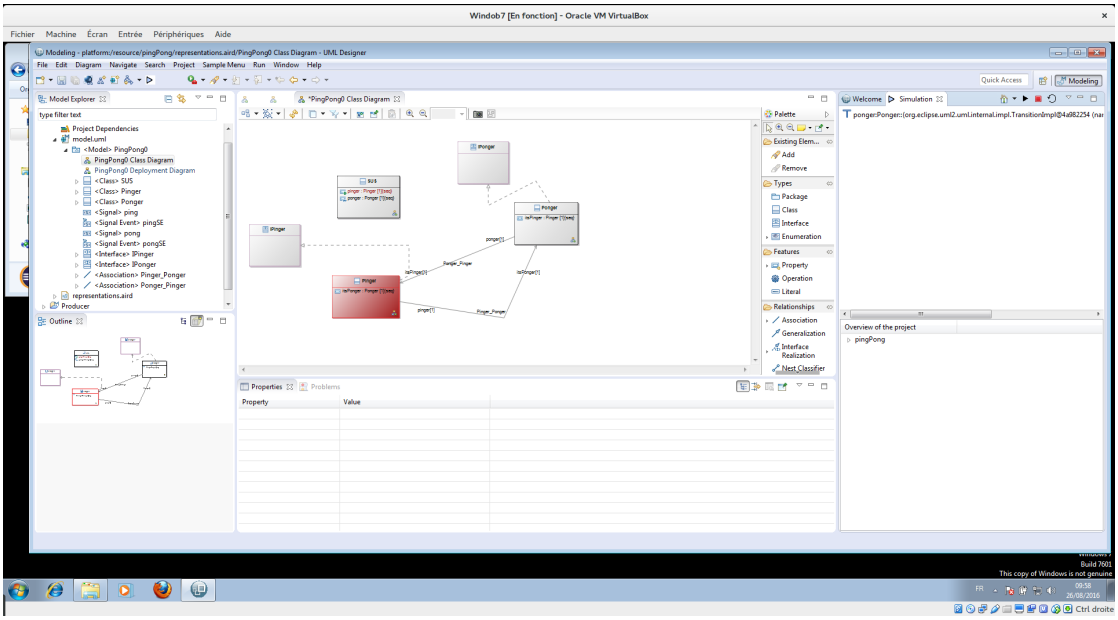


Figure 9.3: screenshot of the Windows virtual machine

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 this type of 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, my work has allowed me to better understand Statechart. 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.

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

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[5]

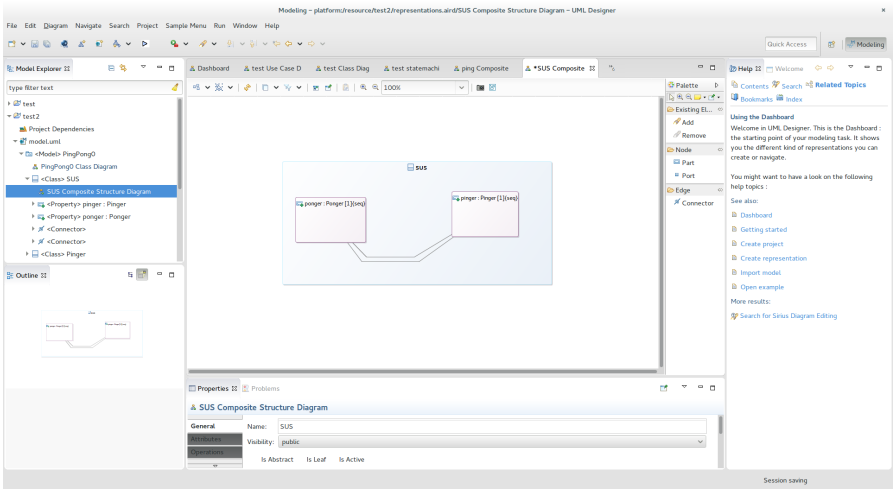


Figure A.2: Screen shot of UML Designer

- 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.

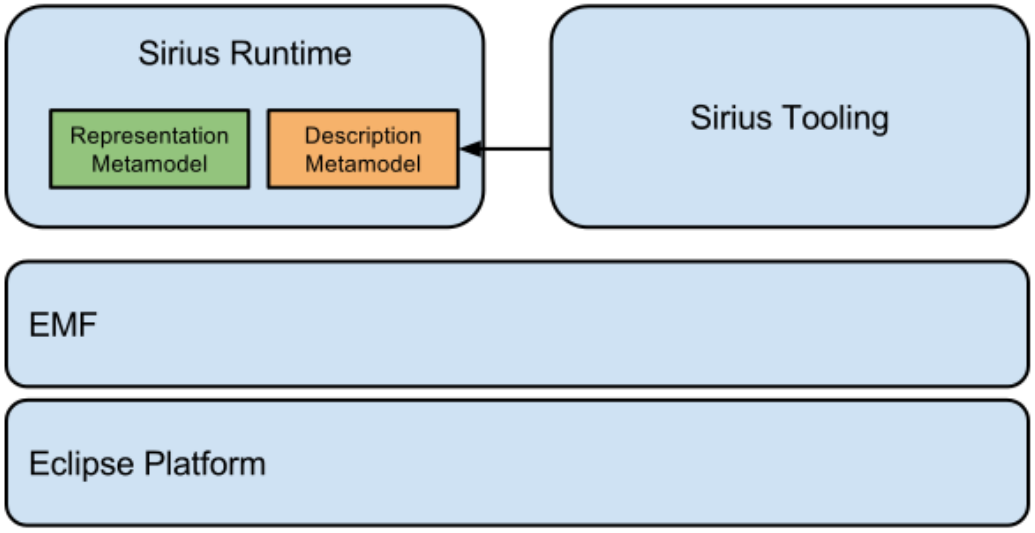


Figure A.3: Sirius architecture[4]

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.

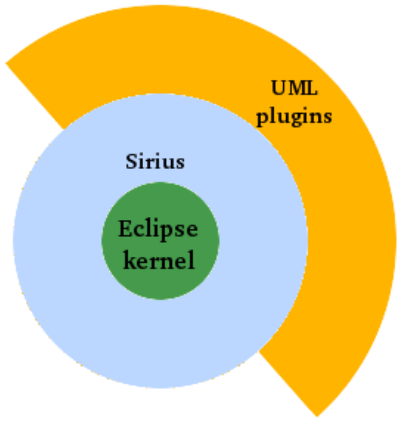


Figure A.4: The UML Designer kernel

[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.

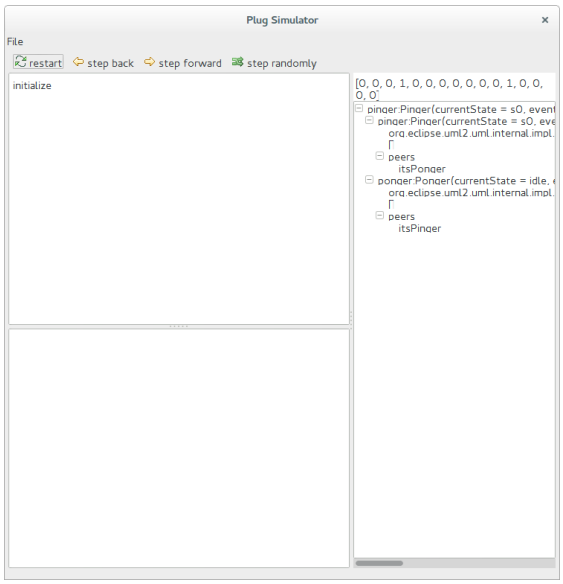


Figure B.1: Mr Teodorov simulator

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

APPENDIX C

Communication enter process

File

Advantages	Drawback
Problem when two software want to change the same file at the same moment	Communication asynchronous

Named pipe

Advantages	Drawback
It is possible to use the Simulator outside the graphical modeling tool	Only available on POSIX systems, Windows, AmigaOS 2.0+

Socket

Details on the chapter 5.

Signal

Advantages	Drawback
	not usually used to transfer data

Shared Memory

Advantages	Drawback
It is possible to use the Simulator outside the graphical modeling tool	

List of Figures

Bibliography

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

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

[3] Obeo. Contribute developer guide.

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

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

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

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

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