Angelino PICONE
Promotion 2010-2013
Département Informatique & Gestion

École Polytechnique de Montréal
2700, chemin de la Tour
Montréal, Québec, Canada

4th year internship

conducted from May 1st to August 31st, 2012

Improving and Extending a Quality Measurement Web Application

Supervisor Polytech'Montpellier
Corinne SEGUIN

Supervisor Ecole Polytechnique de Montréal
Yann-Gaël Guéhéneuc
Full professor, Ph.D. and eng.
Canada Research Chair on Software Patterns and Patterns of Software
Contact: +1 514 340 4711 #7116
www.ptidej.net

Table of Contents

List of Figures

Acknowledgments

I want to warmly thank Yann-Gaël Guéhéneuc, my tutor, for welcoming me within his laboratory. It was a great and rewarding experience that I will not forget. Thank you for your cheerfulness, your advice, your support, and your help.

Many thanks to all the members of the lab, for integrating me so easily into the team and for sharing your experience and your culture during lunch time. Special thanks to Nayantara Duttachoudhury, Adrien Serveaux, Étienne Duclos, and Wei Wu for their friendship and help. And thanks to Sébastien Colladon who keept on helping on his plugin and who always kindly gave me explanations on his work.

Thanks Corinne Seguin, my teacher from Polytech'Montpellier, for your concern about the good development of the internship. It was really nice to meet you there!

Thanks to everyone who enabled this internship and are not thanked enough, like Lysianne Buisson, Hélène Tortosa and Jérémy Vacquie.

Finally, I would like to thank all the people I met there who made this stay really amazing. I cannot wait to come back and enjoy Canada at another season.

Introduction

To complete my 4th year at Polytech'Montpellier, in Computer Science & Management Information Systems, I did an internship abroad.

I chose to do it abroad, on the one hand to improve my English level and become a more fluent speaker, and on the other hand to discover a new culture, a new way of life.

I had the chance to be welcomed at the École Polytechnique de Montréal, among its Computer and Software Engineering department. I was all the more happy because the speaking language in the lab was English, as the students come from all over the World; there was a real multiculturalism.

My main objective during this internship was to improve and complete the implementation of **SQUANER** (Software QUality ANalyzER), a tool for developers that analyses the quality of a project.

Indeed, the scientific literature states that maintenance, which represents, for most software projects, the biggest cost in the life time of the project, could be reduced by improving the quality of the project during its development.[1]

Thus, SQUANER attempts to help the developers to improve the quality of their projects by giving feedback about the written code: it analyses the code of the project, inspects the use of known bad behaviour, and calculates several quality models and faults predictions. However, SQUANER was not usable yet because it could not be deployed on a server and perform analyses. My job was to finish its implementation and improve it by adding some new features.

In the following, I will present in the first place my working environment, and then I will give details about SQUANER and the purpose of my internship, before explaining my work to finish SQUANER's implementation and its improvement.

[1] See Bibliography, references 6, 7 and 9.

Situation

Presentation of the host organization

Figure 1: Pavilion Claudette Mackay-Lassonde

The *Université de Montréal* is a public university founded in 1878. It is one of Canada's leading universities: it is the second in the number of students, the third one in terms of budget allocated to research and it is the largest francophone university in the world.[2]

The *École Polytechnique de Montréal* is affiliated with the Université de Montréal and acts as its engineering faculty. It was founded in 1873 and currently has about 6,420 students.[3]

The Department of Computer Engineering and Software Engineering is located in the pavilion Claudette Mackay-Lassonde. Dating from 2005, it offers a pleasant working environment. The laboratory Ptidej (Pattern Trace Identification, Detection and Enhancement in Java[4]) is located within the Department of Computer Engineering and Software Engineering from the École Polytechnique. It was created in 2003 by Yann-Gaël Guéhéneuc, my tutor.[5]

The objective of this lab is to develop software engineering theories, methods and tools to assess and improve the quality of object-oriented programs[6], by promoting the use of patterns[7] in the code, design, and architecture. The lab members develop a suite of tools to evaluate and improve the quality of object-oriented programs[8], using design patterns.

The laboratory is composed of different members. Yann-Gaël Guéhéneuc, Professor and Research Chair holder leads laboratory. The other members are students in Master of Science (M.Sc), doctoral students (Ph.D.) or interns (B.Sc). There are currently sixteen Ph.D students, two Master students and six interns. Most of these students are co-supervised with colleagues, in particular Giuliano Antoniol, with whom the lab is shared.

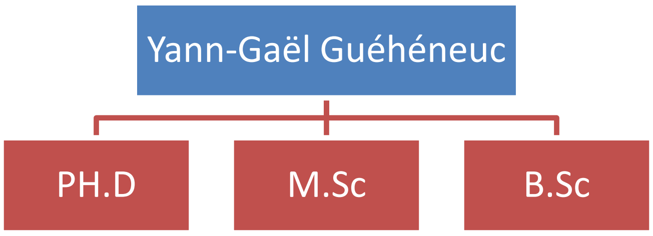


Figure 2: Organization chart of the lab

[2] See Bibliography, reference 11.
[3] See Bibliography, reference 12.
[4] Java is a programming language. Most of the tools developed by the lab are written in Java.
[5] See Bibliography, reference 14.
[6] See Glossary: Object-Oriented Programming.
[7] See Glossary.
[8] See Glossary.

All students are under the direction of the team leader and his colleagues who must not only guide their research, but also manage staff and take care of relations with the outside of the laboratory to obtain the funds needed for research.

Ptidej collaborates in particular with two other laboratories of the *Ecole Polytechnique de Montreal*: SoccerLab (Cost-effective Software Change and Evolution Research Lab) led by Professor Giuliano Antoniol and MCIS (Structural Maintenance of Software and Intelligence) led by Professor Bram Adams.

Problematic of the internship

Despite the large number of quality models and publicly-available quality-assessment tools, such as *PMD*, *Checkstyle,* or *FindBugs*, very few studies have investigated the use of quality models by developers in their daily activities. One reason for this lack of studies is the absence of integrated environments for monitoring the evolution of software quality. This is why **SQUANER** (Software QUality ANalyzER), a framework[9] for monitoring the evolution of the quality of object-oriented systems, has been developed by the lab.

SQUANER connects directly to the SVN[10] repository a project, where the project is stored, extracts the source code, and performs quality evaluations and faults predictions every time a commit[11] is made by a developer. After quality analysis, a feedback is provided to developers with suggestions on how to improve their code.

SQUANER is based on the *Ptidej Tool Suite*, and has been created in 2010 by an intern, Nicolas Haderer. He analysed the state of the art in software quality and in monitoring, and it appeared that there were few existing tools, and those filled a lot less objectives than SQUANER.

Its construction has been continued by another trainee, Samuel Auguste, in 2011. However, when I arrived in the lab, it wasn’t finished yet and not even working, and could not be used. My work was to make sure SQUANER could be used before my leaving.

Objectives of the intern

This internship divides into two main parts: the implementation of SQUANER, and its improvement.

The implementation consists in debugging the Web application, finishing it, and finally deploying it on the servers of the lab so that it can be used by any people. Once this is done, my main objective is completed, and I can devote time to the enhancement of the project.

The second part of the internship was the improvement of SQUANER by adding some features, in particular two new parsers for Java source code and C++ source code.

[9] See Glossary.
[10] See Glossary.
[11] See Glossary.

SQUANER implementation

The first and main objective of my internship was to make sure SQUANER would run before I leave. It was already well advanced but it had to be corrected, adapted and enhanced to run neatly on the server.

Details on SQUANER

A first prototype, called *Grass Monitoring Project* was created in 2008 by Giuliano Antoniol, the team leader of Soccerlab, and Marcus Netler, a professional working on GRASS, a geographical information system, but it was developed to fit a particular case and could not be used to analyze any project.

Then, in 2010, an intern developed *Gutsy*, based on the first prototype. The idea was to enhance the functioning and the quality evaluation of the *Grass Monitoring Project*, which was low-level, and to open the use to a largest public. However, as the prototype was so fitted to a particular project, it was hard to let *Gutsy* work on other projects, and the quality evaluation was still not as good as it could be.

This is why this intern, Nicolas Haderer, decided to begin a new project, **SQUANER**, based on fundamental principles of *Gutsy*, but using the *Ptidej Tool Suite* to perform quality evaluation, and with its own architecture.

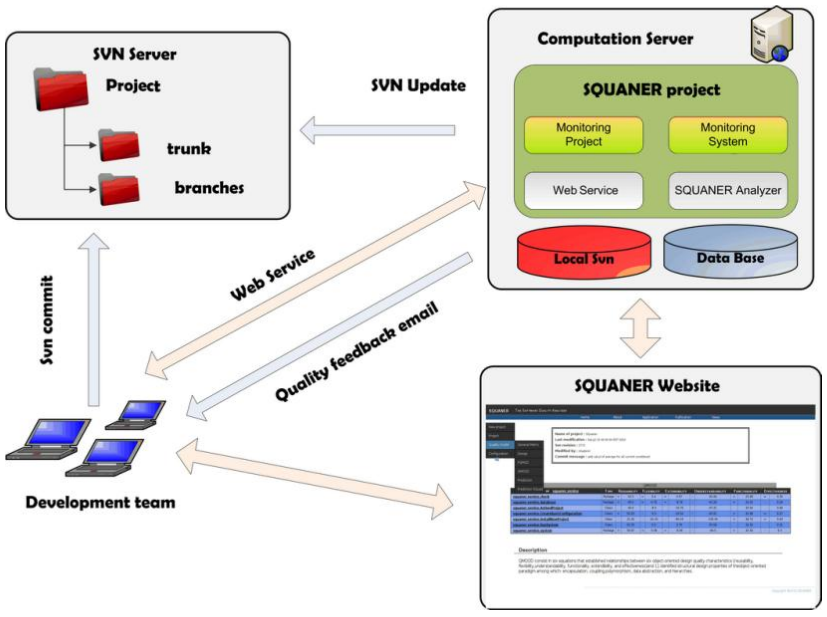


Figure 3: Architecture of SQUANER

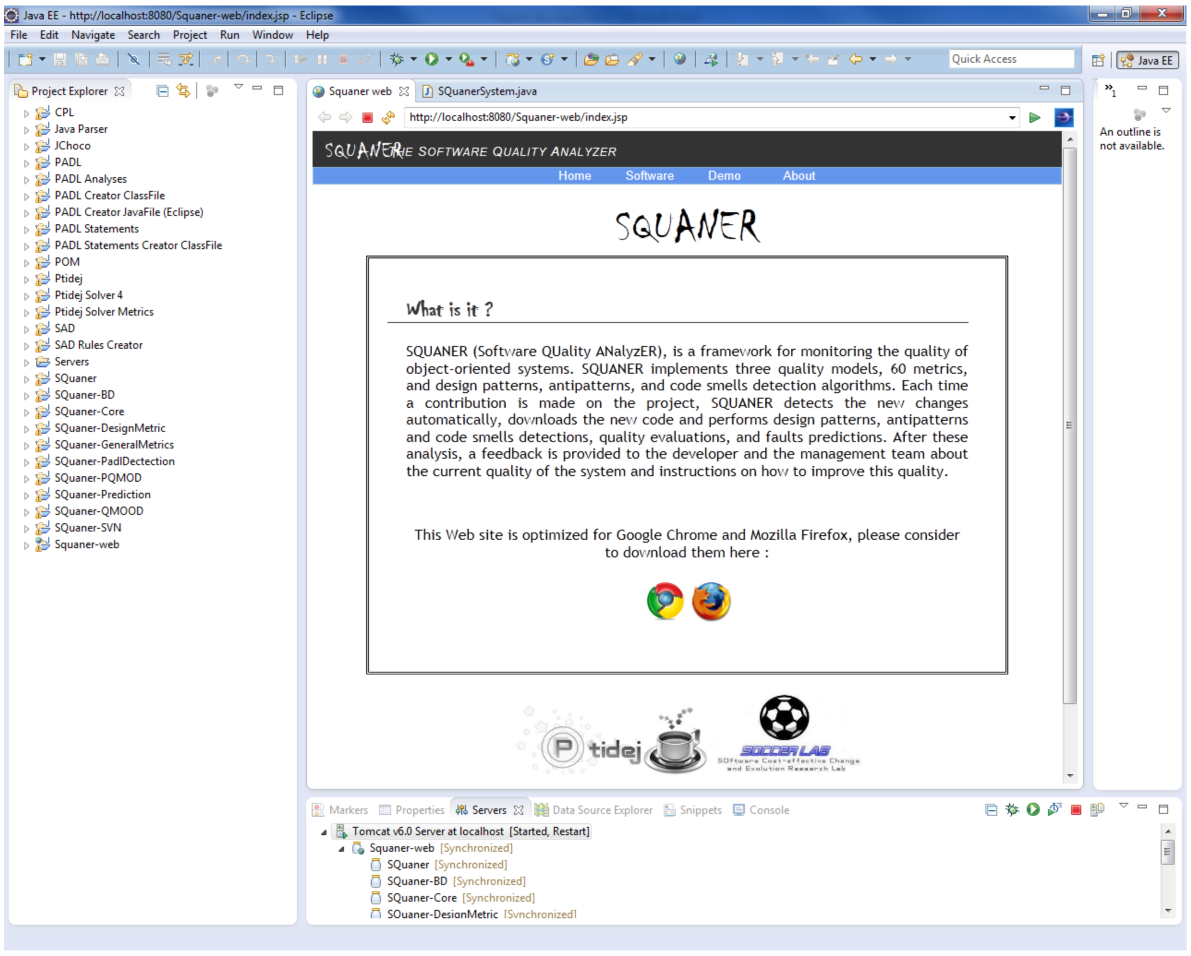
As we can see on Figure 3, SQUANER is a software tool composed of several pieces of program, running on a computation server.

The development team, which implements a new project, can work as a team thanks to a SVN server. This server allows to work remotely, in real time, and to perform versioning. Each time a developer makes a change in the code of the project, he commits these changes to the server. The SQUANER web application detects the change in the server, downloads the new code, and analyzes it using PADL models[12] of the Ptidej Tool Suite. The results of the analysis are stored in a database hosted on the computation server. They are then sent by e-mail to the developers who committed the changes and can also be accessed on-line for historical analysis.

Finally, the SQUANER web application is hosted on a computation server. The developer team can, thanks to the website, add a new project to be analyzed, and consult analysis results.

Working environment

As the Ptidej team developed all the projects in the Eclipse IDE, and since I knew this IDE[13], it seemed not only natural but also indispensable to use it. Thus, this software development environment has been used to write and modify SQUANER.



SQUANER (Software QUality ANalyzER), is a framework for monitoring the quality of object-oriented systems. SQUANER implements three quality models, 60 metrics, and design patterns, antipatterns, and code smells detection algorithms. Each time a contribution is made on the project, SQUANER detects the new changes automatically, downloads the new code and performs design patterns, antipatterns and code smells detections, quality evaluations, and faults predictions. After these analysis, a feedback is provided to the developer and the management team about the current quality of the system and instructions on how to improve this quality.

This Web site is optimized for Google Chrome and Mozilla Firefox, please consider to download them here :

Figure 4: Eclipse, development environment

[12] The PADL meta-model (Pattern and Abstract-level Description Language) is a semi-language independent meta-model conceived by the Ptidej team. It is at the core of the *Ptidej Tool Suite*, and allows representing generically a program, whatever the programming language used, to analyze it.

[13] Integrated Development Environment. See Glossary.

As SQUANER also consists of a *web service*[14], I used Eclipse IDE for Java EE[15] Developers that allowed me to run the website on a local server. Figure 4 shows the environment offered by Eclipse. On the left-hand side, we can notice that SQUANER is indeed divided into several Eclipse projects; it also depends on other projects (such as the *Ptidej Tool Suite*). Then, the central frame shows SQUANER's website launched from Eclipse.

However, SQUANER is meant to be deployed on a server to be used worldwide. More precisely, it has to be deployed on Soccerlab's server[16], in a Tomcat server[17]. Thus, people can use SQUANER through its website that interacts with SQUANER. This justifies the use of Java EE. The programs and resources are transferred to the server via SVN or SFTP[18], and the website (and consequently SQUANER system) is reachable from a simple web browser.

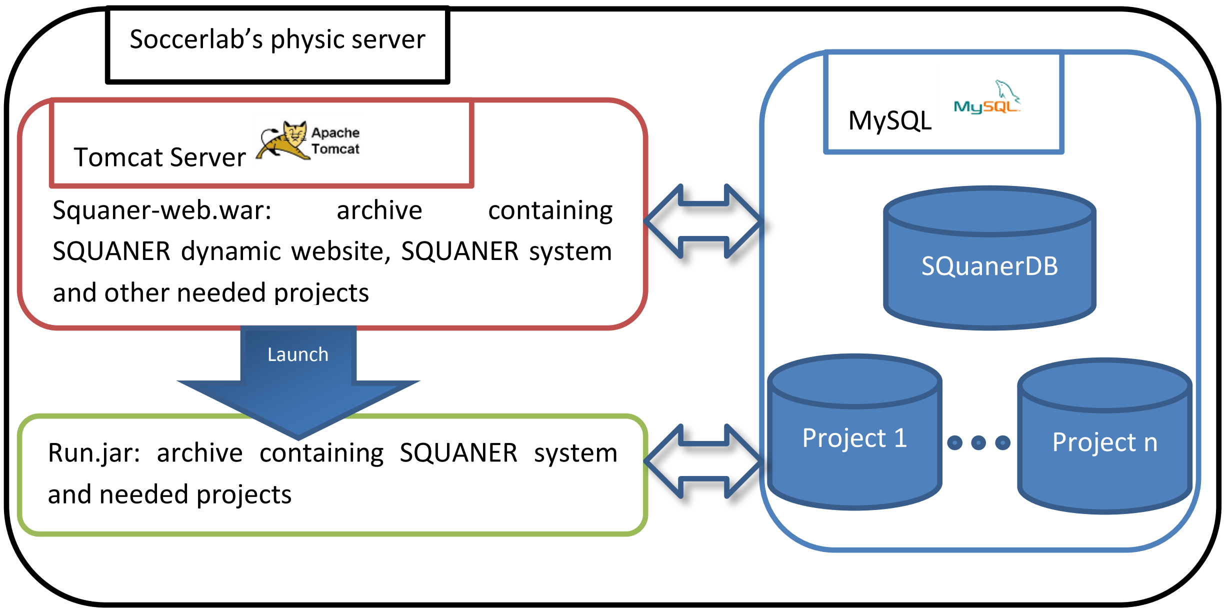

Soccerlab's physic server
Tomcat Server
Apache Tomcat
Squaner-web.war: archive containing SQUANER dynamic website, SQUANER system and other needed projects
Launch
Run.jar: archive containing SQUANER system and needed projects
MySQL
MySQL
SQuanerDB
Project 1
Project n

Figure 5: Composition of the physical server

Finally, data resulting from analyses can be accessed through SQUANER's website after being stored in MySQL[19] databases. One database is used for SQUANER's properties, called SQuanerDB, and there is one database per analyzed project, to store the projects' properties and their analyses.

[14] See Glossary.
[15] A platform provided by Oracle, a software company, used to develop and run notably network applications.
[16] Reminder: Soccerlab is one of the two labs collaborating with the Ptidej lab.
[17] Apache Tomcat is a web container that allows creating dynamically data inside a HTTP server (a web server, notably used to consult web pages).
[18] SSH File Transfert Protocol. See Glossary.
[19] MySQL is the world's most used Relational DataBase Management System (RDBMS). It is a popular choice of database for use in web applications. For instance, it is used by famous websites such as Google, Wikipedia, Facebook or Twitter.

Both SQUANER system and SQUANER website can access and put data into the databases, according to Figure 6.

Action	Application	Database
Read SQUANER's configurable properties	SQUANER **system**	SQuanerDB
Modify some SQUANER's properties	SQUANER **system**	SQuanerDB
Get the list of the projects to analyze	SQUANER **system**	SQuanerDB
Add a new project	SQUANER **system**	SQuanerDB *Creates project's database*
Store a new analysis of a project	SQUANER **system**	Project's database
Add a new user	SQUANER **website**	SQuanerDB
Modify a project's properties	SQUANER **website**	Project's database
Consult the analysis	SQUANER **website**	Project's database

Figure 6: Uses of the databases

Conduct of my work

The main part of my work was to debug the existing code. However, I chose to divide this first part of my internship into several steps; I proposed those to my supervisor and he agreed with my strategy.

Documentation on SQUANER

My first step was to gather information about SQUANER. Indeed, SQUANER was a well-advanced project, and I had to learn its goal, its functioning, the context, and the work progress. This step began before the beginning of the internship: while talking with my tutor by email, I realized the scope of the project and asked him for some documentation so that I would be aware of the situation as soon as I arrived. Thus, before my arrival, I studied the reports from the two previous interns who worked on SQUANER, a publication, and a presentation. During the internship, I could consult this documentation, as well as the source code, each time I had to understand a process, to modify or correct SQUANER.

Handling

On my arrival, I was first presented to the team, I discovered the school and I settled in the lab. There, I downloaded several pieces of information on the lab's wiki[20], notably the code conventions of the lab[21] to be sure I would respect the lab's conventions and make sure that my work would fit into theirs. On the wiki, I also found more information on the work done for SQUANER[22].

[20] The lab's wiki is part of the Ptidej website but is reserved for team members. Let's remind that a wiki is a website which allows users to add, modify or delete content. This allows the team members to add content about their work, and any kind of information to help themselves in the future, or their teammates.
[21] See Bibliography, reference 8.
[22] See Bibliography, references 1 and 2.

At this point, I had enough information on SQUANER to know what SQUANER is, what had been done, and what I had to do.

Then, I downloaded the workspace and installed all the tools I would need: Eclipse JEE to work on SQUANER’s code, Apache Tomcat to run the web service, and MySQL for the databases.

I had to make sure I used the same versions of each application than the ones used on Soccerlab’s server, so that I would not have problems related to version incompatibility during the deployment of the project at the end of the month.

I started taking those tools in hand by opening the workspace in Eclipse, inserting the databases into MySQL thanks to an AMP solution[23] and deploying the last version of SQUANER on the local Tomcat server.

I also started to learn about Java EE, as I did not know this technology used for SQUANER website.

While trying the different tools, I quickly realized that, indeed, SQUANER was not able to run. But before I could reach that conclusion, I had to follow several steps that were not indicated. That is why I decided to write my own documentation on SQUANER for the future contributors of SQUANER.[24]

Work on the code

First, I analyzed the different SQUANER projects, the dependencies, and the written code, to know better the application, to understand the sources of the problems, and to know where I would have to make changes. I tried to use automatic reverse-engineering[25] tools to get the application’s design, but the whole architecture was so complex that I had many issues with the tools that I used; I decided not to waste time on it and kept on reading the code. While reading, I added some annotations and comments to keep track of my remarks and to fill the Javadoc[26] that had not been maintained by the previous interns; this helped me a lot to understand the logic organization of SQUANER and the different processes used.

Then, I could finally tackle the problem of debugging the application. Most of the issues came from dependency issues, missing try/catch statements, and badly configured properties.

The first big issue was that I could not launch the application. Solving the problems previously mentioned helped to launch it. Then I could help the access to the list of the projects to analyze, and fix some other mistakes.

[23] AMP is an acronym for Apache, MySQL, Perl/PHP/Python. It is a set of packages used to run dynamic Web sites or servers. Apache is a web server, MySQL a relational database management system, and Perl, PHP and Python are programming languages.
[24] See Appendix 1.
[25] See Glossary.
[26] Javadoc is a documentation generator: it creates documentation in the form of web pages from special comments inserted in Java source code.

I also had a problem of which I could not find the source but I have been helped on it by my tutor, who knew perfectly all the projects on which SQUANER depends and found a missing case in a switch statement[27] in *Ptidej Tool Suite*.

At this stage, SQUANER could analyze only two types of files: java files (Java source code) and class files (Java bytecode[28]).

Although the ClassFile Analyzer worked correctly, I had problems with the JavaFile Analyzer: the analyzed project was not added to the list, and then the analyses were not stored in the database. The problem was fixed by adding missing code in another Eclipse project, on which SQUANER depends.

Once all the bugs were gone, I could insert some more code to make the application usable. Indeed, the website was not quite functional and prevented the user from performing some tasks. I corrected the web design problems and the information display, I enhanced the navigation, and I added some features such as the accounts system, which allows multiple users with their own private projects, and an administrator panel that allows to launch SQUANER system.

Tests

Before deploying the application on the target server, I wanted to make sure it was functional.

Of course I had to test the functionalities that I debugged while I worked on the code, but I tried to avoid working with a trial-and-error approach to force me into thinking more carefully my solutions and understanding the application. When working with such a project, SQUANER, which depends on many other ones, it is easy to make mistakes that would have consequences on all the Eclipse projects, which are also used by other members of the lab, as the dependencies between those projects are very numerous.

However, I did this time a conditioning test and a usability test. I deployed the application on my local tools and tested the whole process: I acted as a user by adding some projects to be analyzed, committing some changes in the projects, waiting for analyzes and finally consulting them. I also had to act as an administrator by launching SQUANER system and keeping an eye on the log files[29] to detect any problem.

Moreover, I had some teammates of the lab test the application, so that I would be sure not to omit a detail, to get their impression, bug reports and improvement ideas. All the feedbacks I received about the web application were positive.

[27] A switch statement is a type of selection control mechanism. It provides different reactions depending on the value of a variable: the cases. It is the equivalent of the imbrication of several conditional statements, which perform a specified action if a given condition is verified.
[28] Java bytecode, which is obtained by the compilation of source code, is a set of instructions that will be executed by a Java Virtual Machine. It is a binary code which groups all needed resources together.
[29] Those are files created and filled by SQUANER system. They keep track of all the actions of SQUANER (at a specified time, which project is being analyzed and at which step of the analysis). If there are errors encountered, they are recorded in the log files.

Also, with the help of my tutor, I analyzed the relevance of the analysis results. This testing phase had been a success: the application worked correctly and efficiently. I had then to deploy the application on Soccerlab's server, for its use in real conditions.

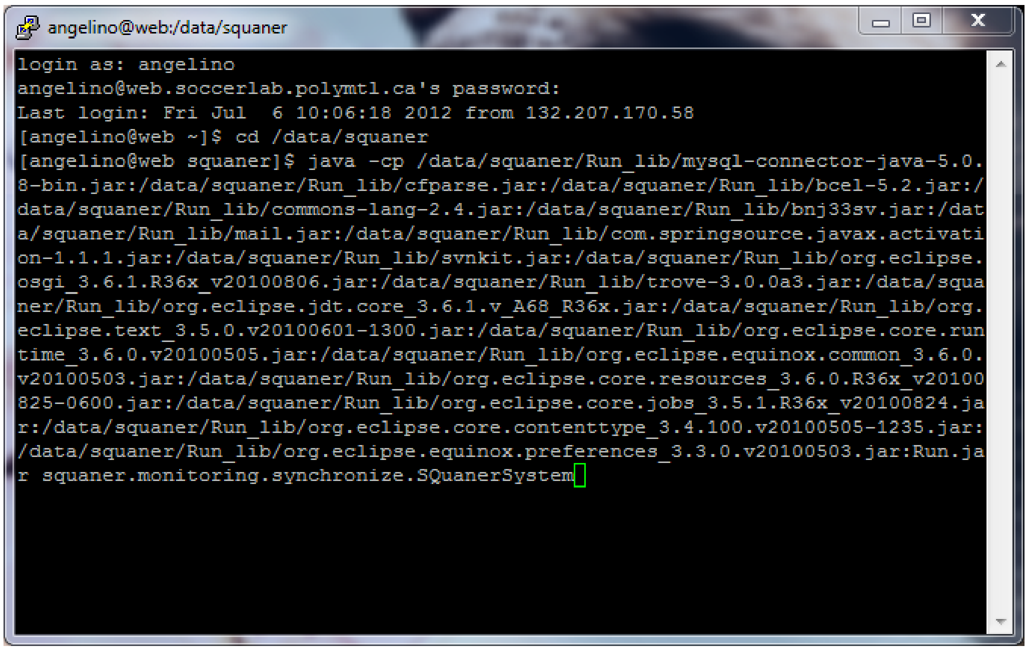
Deployment

To deploy SQUANER on Soccerlab's server, I had to follow 5 steps. First, I had to adapt the environment variables. Those are mostly paths that give SQUANER the location of needed files — those paths are not the same on my local machine and on the target server. Then, I could export SQUANER and put it on the server. Then, I had to create SQUANER database on Soccerlab's MySQL database system. Afterwards, I had to put on the server SQUANER's configuration files, which are used by SQUANER to get parameters and put data. For example, the access to the database is stored into these files. Finally, I had to export the SQUANER web application and put it in the Tomcat server.

Once all was put on the server, I could try the application in real conditions. I encountered three types of problems: class path, permissions, and compatibility.

The most difficult part had not been to solve those problems but to find them, find their cause, understand them.

As we saw on Figure 5, the web application located on Tomcat launches SQUANER system, which is a JAR archive. SQUANER is built on several projects, and has a dense network of dependencies: many external libraries[30] are needed, and I had to find them all, and specify where to find them in the command lines used to launch SQUANER system. The final command can be seen on Figure 7.



```
login as: angelino
angelino@web.soccerlab.polymtl.ca's password:
Last login: Fri Jul  6 10:06:18 2012 from 132.207.170.58
[angelino@web ~]$ cd /data/squaner
[angelino@web squaner]$ java -cp /data/squaner/Run_lib/mysql-connector-java-5.0.
8-bin.jar:/data/squaner/Run_lib/cfparse.jar:/data/squaner/Run_lib/bcel-5.2.jar:/
data/squaner/Run_lib/commons-lang-2.4.jar:/data/squaner/Run_lib/bnj33sv.jar:/dat
a/squaner/Run_lib/mail.jar:/data/squaner/Run_lib/com.springsource.javax.activati
on-1.1.1.jar:/data/squaner/Run_lib/svnkit.jar:/data/squaner/Run_lib/org.eclipse.
osgi_3.6.1.R36x_v20100806.jar:/data/squaner/Run_lib/trove-3.0.0a3.jar:/data/squa
ner/Run_lib/org.eclipse.jdt.core_3.6.1.v_A68_R36x.jar:/data/squaner/Run_lib/org.
eclipse.text_3.5.0.v20100601-1300.jar:/data/squaner/Run_lib/org.eclipse.core.run
time_3.6.0.v20100505.jar:/data/squaner/Run_lib/org.eclipse.equinox.common_3.6.0.
v20100503.jar:/data/squaner/Run_lib/org.eclipse.core.resources_3.6.0.R36x_v20100
825-0600.jar:/data/squaner/Run_lib/org.eclipse.core.jobs_3.5.1.R36x_v20100824.ja
r:/data/squaner/Run_lib/org.eclipse.core.contenttype_3.4.100.v20100505-1235.jar:
/data/squaner/Run_lib/org.eclipse.equinox.preferences_3.3.0.v20100503.jar:Run.ja
r squaner.monitoring.synchronize.SQuanerSystem
```

Figure 7: SQUANER system launch command

[30] See Glossary.

The permission issues appeared because the web service was launched by the user *Tomcat*, and the analyzer was launched by me. As Apache Tomcat and I were not in the same group[31], SQUANER could not write some data, and got stuck in its analysis process. Moreover, I could not add some needed data because of the right restrictions on the server. I contacted the technician in charge of rights on the server, and also M. Giuliano Antoniol, lead professor of the Soccerlab laboratory. They gave me the necessary rights on the server, and put Apache Tomcat and me in the same group. However, this last point was not necessary, as I only needed to make Apache Tomcat launch SQUANER system, as specified in Figure 5.

Although SQUANER system was running, I got no result to consult. I looked at the database to see if there were results stored, but it was empty. The problem was actually coming from the fact that the RDBMS on Soccerlab's server and on my own machine were not exactly the same, and there were case differences between the databases and SQUANER system. One of the two RDBMS was case sensitive, and not the other one. I just needed to modify SQUANER system to make the table names and the field names uniform on both the database and SQUANER system.

Finally, I encountered once again an issue with the JavaFile analyzer: the analyses of Java source code projects were not stored in the database. It appeared that the project of the analyzer, which had been added lately by a student of the lab, was built on a different version of Java from the Java Virtual Machine of Soccerlab's server. Changing the Java version of the project in Eclipse and re-deploying SQUANER solved the problem.

Intermediate assessment

The deployment was finally finished, and the application ran perfectly on the server. Thus, the main objective on my internship was completed rather quickly.

However, SQUANER could be improved in many ways, and it was precisely the purpose of the rest of my internship: use all the time remaining to improve SQUANER.

[31] In computing, the term *Group* refers to a grouping of users of a system. The users of a same group share the same access control to the system. In particular, they can read, write in or execute the same files.

SQUANER improvement

With SQUANER operational, the work left was to enhance it. Even if all was working, SQUANER website could have many improvements. Indeed, I noted some ideas I got while working on it, foremost about ergonomics, functionalities and errors prevention.

However, my tutor decided that these ideas were of a lower priority and asked me instead to begin with the addition of a new analyzer. Indeed, SQUANER could for the moment only analyze two types of projects: java files projects and class files projects, whereas SQUANER was also built to analyze C++ files and C# files[32].

A C++ analyzer, called PADL Creator C++, had already been developed by another intern, Sébastien Colladon, in 2012 just before my arrival. My new work was then to integrate this new analyzer into SQUANER.

Details on PADL Creator C++

As for the JavaFile and ClassFile analyser, the goal of PADL Creator C++ is to create a PADL model that represents a program in C++ in that particular case.

The input of this analyser is a C++ project. It will parse[33] the code and generate a PADL model as output.

To parse the C++ code, many tools have been studied to see which one would be the best. It appeared that the best way to parse the code was to use the IDE *Eclipse CDT*[34]. Indeed, by creating a plugin[35] for Eclipse, we can ask it to parse the code and then get its AST[36]. Thanks to this AST, a PADL model can be created.

Thus, PADL Creator C++ is a plugin for Eclipse. However, the analyzer must be integrated into SQUANER, which is a program launched from a server, without any human interaction. Consequently, SQUANER must launch Eclipse and make it work automatically. To do so, an instance of Eclipse without Graphical User Interface must be used: a Headless Eclipse.

Indeed, SQUANER will launch a Headless Eclipse by executing a command, giving the plugin as application of Eclipse, which allows it to control the Java Virtual Machine. The plugin will create the PADL model, and SQUANER will fetch it to perform quality analysis. This expected process is drawn on Figure 8.

[32] C++ and C# are two other programming languages. Like Java, they are object-oriented languages.
[33] Syntactic analysis. See Glossary.
[34] Version of Eclipse IDE which provides C and C++ Integrated Environment Development.
[35] See Glossary.
[36] Abstract Syntax Tree. See Glossary.

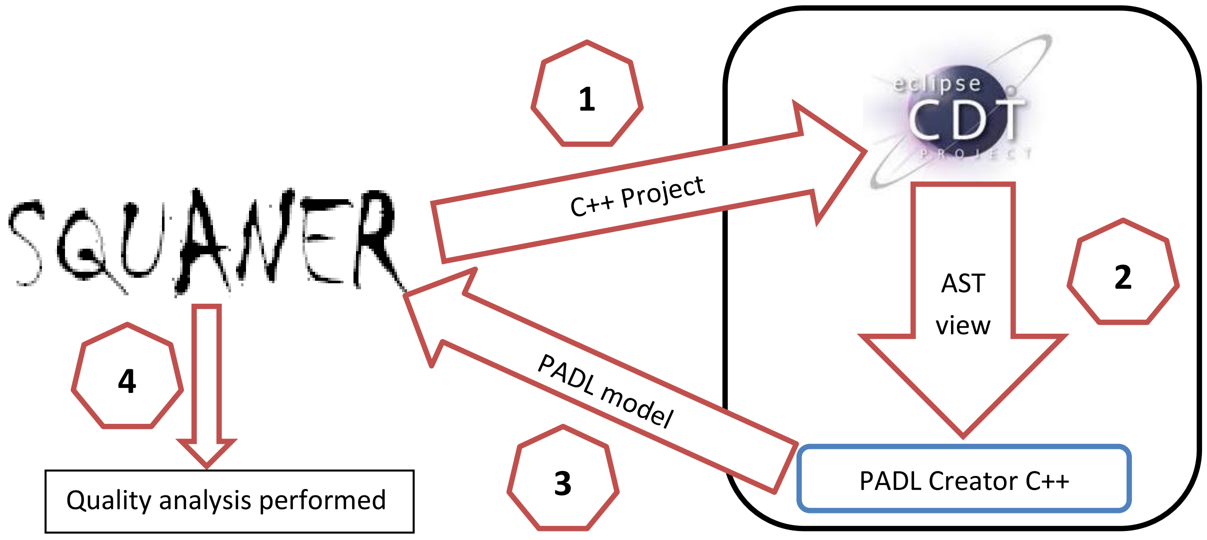


Figure 8: Analysis process of a C++ project

However, Sébastien Colladon only wrote the code that could create the PADL model within a graphical Eclipse. I had to modify this code so that the PADL model could be given to SQUANER. Furthermore, he never managed to deploy its plugin and make it run with a Headless Eclipse. Moreover, the project that I was given did not even compile. Thus, I had to handle these three issues before trying to integrate it into SQUANER.

Conduct of my work

I had to go ahead with two tasks at the same time: make PADL Creator C++ work (i.e. get a PADL model from any C++ code) and launch a Headless Eclipse with PADL Creator C++ as application.

Documentation and handling

As during the first part of my internship, I began by reading the documentation about the tools on which I would work. I downloaded the workspace and asked my tutor for some details about the integration of PADL Creator C++ and for the internship report of Sébastien Colladon.

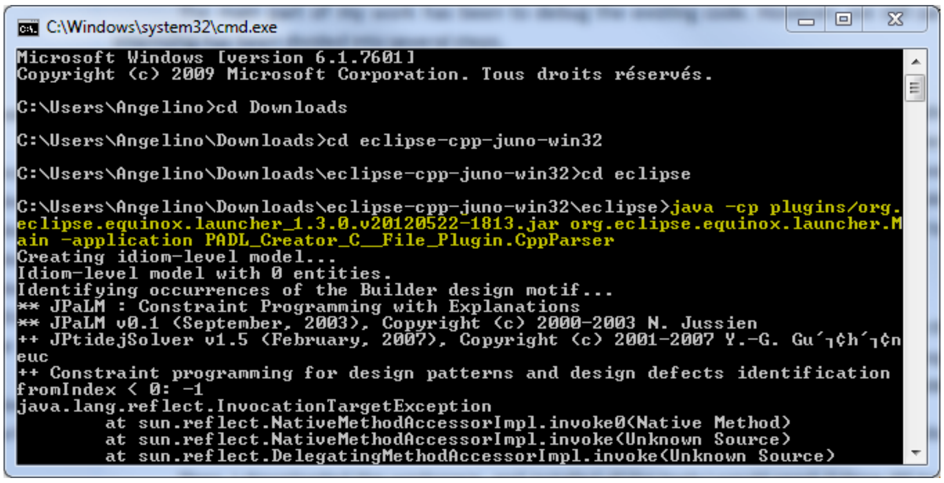
This report spent much time on detailing the PADL meta-model and justifying the choices made (for example, the use of Eclipse CDT for parsing C++ source code), but it cruelly lacked details on the work achieved and on the use of PADL Creator C++. I decided to contact this intern, who kindly answered my questions.

I also got used to the plug-in development environment of Eclipse, since PADL Creator C++ is a plugin.

Launching headless Eclipse

There is very few documentation on Headless Eclipse, and most of it is outdated, based on an older version of Eclipse. Therefore, I read the Eclipse documentation to find a way to launch Eclipse without Graphical User Interface. By mixing the Internet tutorials and the official Eclipse documentation, I managed to identify the command line which could do that. This command can be seen as the highlighted part on Figure 9.

The Headless Eclipse is launched thanks to a Main method[37] contained in the JAR archive provided with Eclipse — org.eclipse.equinox.launcher. Then the plugin is used thanks to the option *application*. The plugin must be called by its ID, which is specified in its Manifest file[38].



```
C:\Windows\system32\cmd.exe
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\Angelino>cd Downloads

C:\Users\Angelino\Downloads>cd eclipse-cpp-juno-win32

C:\Users\Angelino\Downloads\eclipse-cpp-juno-win32>cd eclipse

C:\Users\Angelino\Downloads\eclipse-cpp-juno-win32\eclipse>java -cp plugins/org.
eclipse.equinox.launcher_1.3.0.v20120522-1813.jar org.eclipse.equinox.launcher.M
ain -application PADL_Creator_C__File_Plugin.CppParser
Creating idiom-level model...
Idiom-level model with 0 entities.
Identifying occurrences of the Builder design motif...
** JPaLM : Constraint Programming with Explanations
** JPaLM v0.1 (September, 2003), Copyright (c) 2000-2003 N. Jussien
++ JPtidejSolver v1.5 (February, 2007), Copyright (c) 2001-2007 Y.-G. Gu´┐¢h´┐¢n
euc
++ Constraint programming for design patterns and design defects identification
fromIndex < 0: -1
java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
```

Figure 9: Headless Eclipse launch command

However, the correspondence between the ID and the plugin could not be done. After some research, I discovered it may be coming from a dependency issue.

Dependency relationships are very strong in this workspace, and it had been very difficult to find the source of the problem. To help me, I made a dependencies graph of the plugin from the Manifest files (Figure 10). Thus, I could easily visualise the links between the projects, and investigate them.

During this investigation, I found three sources of problem. First, the plugin PADL Creator C++ is dependent on several projects, some of which are not plugins. However, to be in the plugin registry — which allows to make the correspondence between the ID and the application — the plugin must only depend on other plugins; not on regular project, or external library. Therefore I converted all the projects into plugins. It was not quite difficult: I just created new plugin projects and moved the java source code into the news projects.

Then thanks to my dependencies graph, I easily traced all the links and I realized that a library was missing. I looked for it on the Internet, downloaded it, and added it to the Eclipse plugins.

Finally, an error written in the log file stated that a class[39] had no definition. I located the library that should have contained the definition, but it was empty. Apparently, the automatic exportation of all the needed plugin has failed, so I exported manually the project in a JAR archive and I launched Headless Eclipse by specifying it to look for the class definition in the freshly exported library, but it did not work either. Then, I simply renamed the exported library so that it replaced the original plugin, and it worked.

[37] See Glossary.
[38] See Glossary.
[39] See Glossary: Object-Oriented Programming.

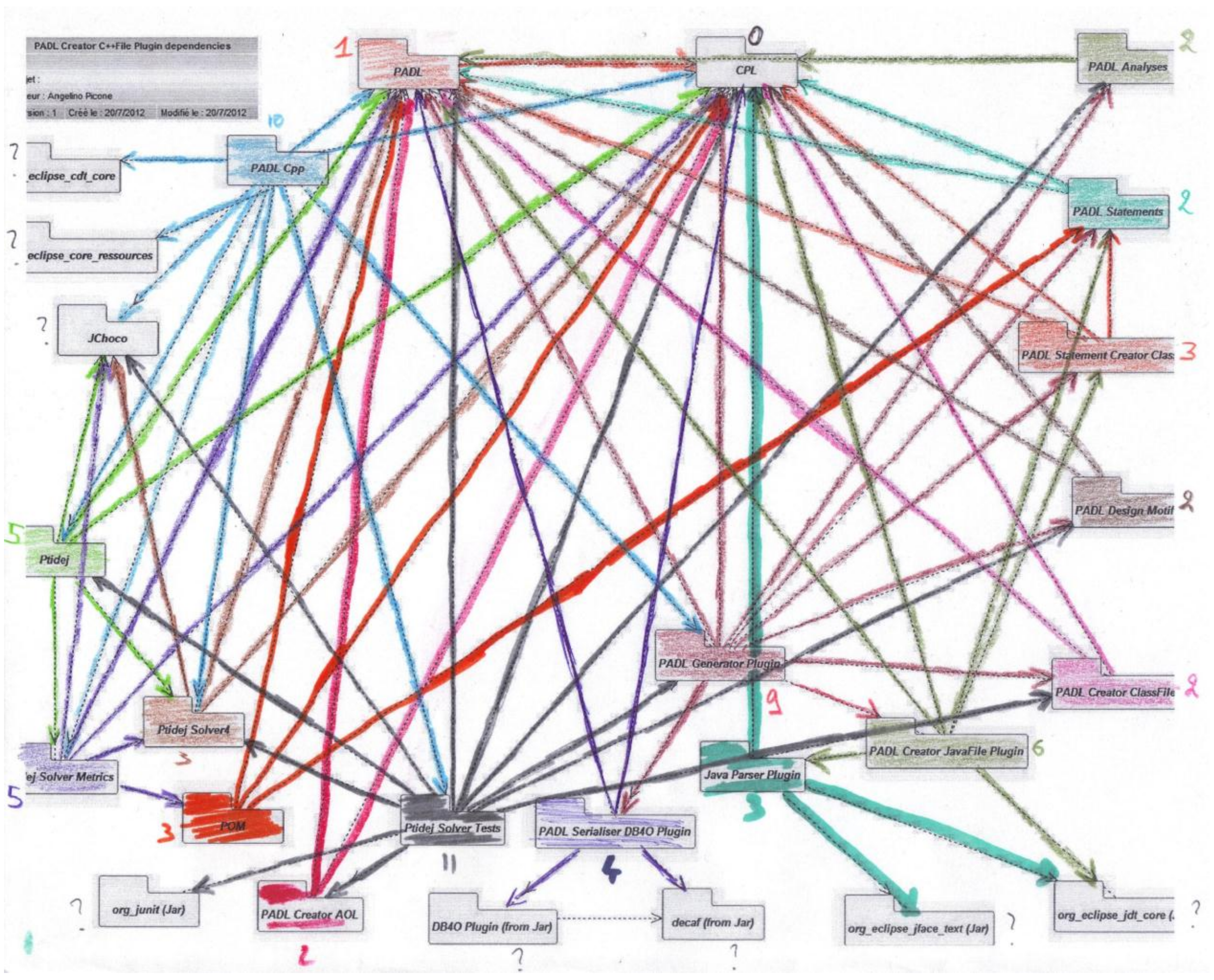


Figure 10: Dependencies graph of PADL Creator C++'s workspace

As a result, I finally had Eclipse work headless with the plugin. Nevertheless, the plugin was still not functional. This work on headless Eclipse just allowed me to make the plugin run automatically, without graphical interface, but PADL Creator C++ was still not doing its job.

By chatting with my teammates, it appeared that this work would be interesting for them. Indeed, the idea to let Eclipse perform automatic tasks may be very helpful for the lab's works; this is why I decided to publish the method on the lab's wiki. This way, all my teammates and even the whole World would access this information when needed.[40]

Integrating PADL Creator C++

In the meantime, I tried to debug the plugin. In view of the difficult situation, I have been helped by the intern who developed the plugin and a student of the lab.

I have been stuck for a while with an execution error. The error came from a project on which the plugin depended, but finding the cause and solving it was very difficult, because of the lack of documentation about the plugin. The problem actually came from the path of the C++ project to analyze; a workspace was automatically created during execution above the workspace of the plugin

[40] See Appendix 2.

in hierarchy, and I needed to put there the C++ files with some specific metadata files in a specific folder.

Moreover, there were some missing elements in the code that caused errors, and I had to add them.

Once all the errors were solved, I tested the plugin; it worked, but the results were not conclusive. Indeed, no occurrence of design motifs is identified. I had to find a way to correct the analysis process, but in spite of all my research and my attempts, no solution was found.

As I knew another intern or another student, or even my tutor, would try to finish this work, I updated the documentation I started about SQUANER. Thus the next collaborator would not lose as much time as I did in handling and would know exactly what has to be done, if I don't manage to finish it before the end of my internship.[41]

Assessment of this second part

The assessment of this second part of the internship is quite mixed. The work on headless Eclipse has been a success and I advanced the debugging of the plugin, but of course without the plugin working, the project is not complete.

[41] See Appendix 1.

Critical analysis

My work during this internship has been mostly to debug code. Thus, a thorough project management was not really possible. I had no analysis to do, no conception (and there was not any), no choices to justify… I just had to make things work without modifying the expected behavior.

As a result, I could not establish a relevant detailed work schedule, but I divided my work into several steps. For the first part of the internship, there were documentation, handling, debug, tests and deployment. Nevertheless, when I was given my first objective, I estimated to one month the time to work, and the product was deployed only 5 weeks later.

As for the second part, I tried to follow the same pattern: I got informed on the tools, I got used to them, and then I worked on the code to make the plugin work before testing and deploying it. Moreover, before working on the code, I did some prior research on Headless Eclipse, which were needed for the analysis process. However, I could not predict that I would have such difficulties with one execution error, and, finally, with the accuracy of the analysis performed by the plugin.

Nevertheless, my work contributed to the lab's work and fitted in line with the series of intern contributions.

I never lost sight of my goals, and was only stopped by an insolvable issue. However, I made sure this work would be continued by documenting it. To my mind, SQUANER is a very useful tool which has an interesting future and I am proud to be a part of it. Indeed, I still delivered a fully usable product.

Moreover, some interns may not be very conscientious; working on interns' work has been very tedious. There was no track of a carefully thought-out work, no conception, no documentation. However, the way I worked, reflective and organized, helped me to progress anyway through such an amount of code.

Finally, the whole project included a technology I never learned at school or by but I managed to use it properly in a very reasonable time, by reading theory and analyzing the code.

Conclusion

SQUANER is a very interesting project, foremost about the need it wants to fill. I never considered quality issues this way, and my curiosity about this field has been largely fulfilled while reading the documentation before the beginning of my internship, which reminded me of some parts of the Object-Oriented Engineering class that I had this year, when M. Stratulat warned the classroom about the importance of code quality.

So far, it has been the biggest project on which I have ever worked. It was a great challenge that I was pleased to take. I never had the opportunity to work on project already well advanced. Thus, before working, I had to study carefully the whole project and the work achieved.

I am happy to have successfully completed the main objective of my internship and, as I am always enthusiastic to learn a new technology, the discovering of Java EE was a very positive point to me.

Of course I am disappointed not to have finished yet the addition of the C++ analyzer, but I like considering that I still did advance the project, contributed with my tutorial about headless Eclipse and helped the work of the future contributor to SQUANER.

Beyond the professional experience gained, I discovered the research world, which is very interesting. My tutor, as a professor, gave all the interns, as well as the students of the lab for which a reminder is always welcomed, a lecture about research. Furthermore, the lab organized every Friday afternoon a seminar, in which some students of the lab explained their research topic and their work progress. It was the occasion for the lab members to consider the interest of different works and to share ideas and solutions, and for me to know more about being a M.Sc. or Ph.D. student. I could get answers to many questions that I had about it and assure my professional project.

Moreover, this internship abroad has been the occasion to travel and visit several places in North America, discover another culture, new people, different wildlife, new landscapes… I met many people from all over the World, all very kind and interesting, and I can also confirm that Quebecois are very, very nice people.

Bibliography

This bibliography gathers resources that have been used to write this internship report or to help me during the internship.

They are sorted first by category, then by the initials of the (main) author. The categories are sorted by alphabetical order.

Documentation

1. **S. Auguste**. Situation Review of Project Manager, Gutsy and Squaner. 2011.
2. **S. Auguste**. Squaner changes report. 2011.

Memoirs

3. **N. Haderer**. Squaner System QUality AnalyzER. 2010.
4. **S. Auguste**. Nettoyage, amélioration et mise en ligne de SQUANER. 2011.
5. **S. Colladon.** Rapport de PFE. 2012.

Periodicals

6. **B.P. Lientz** and **E.B. Swanson**. Problems in application software maintenance. *Communications of the ACM*, vol. 24, no. 11, p.769, 1981.
7. **L. Erlikh**. Leveraging legacy system dollars for e-business. *IT Professional*, vol. 2, issue 3, p. 17-23, 2000.

Presentations

8. **Y-G. Guéhéneuc**. Programming conventions for Java. 2009.

Publications

9. **M. Lehman**. Laws of software evolution revisted. *Software Process Technology*, vol.5, p. 108-124, 1996.

Websites

10. **The Eclipse Fundation.** www.eclipse.org.
11. **Ecole Polytechnique de Montréal**. www.polymtl.ca.
12. **Ptidej Team**. www.ptidej.net.
13. **Site du Zéro.** www.siteduzero.com.
14. **Université de Montréal**. www.umontreal.ca.
15. **The Wikipedia Fundation**. en.wikipedia.org.

Glossary

Abstract Syntax Tree: tree representation of the syntactic structure of a program. It is obtained by analysing the source code of the program. It is composed of nodes which represent different kinds of statements.

Commit: using SVN (see below), it means the user sends its changes from the working copy to the repository.

Design pattern: in computer science, a semi-formal way of documenting a solution to a recurring design problem (i.e. find a good and possible way to make a program do what it is implemented for). It is based on the idea that if you encounter a problem while coding, maybe someone has encountered the same issue but found a solution that would be scalable and flexible.

Framework: universal and reusable software platform used to develop applications, products and solutions. It provides generic functionalities so that a certain behavior of the application can be assured.

IDE: acronym for *Integrated Development Environment*. It is a software application conceived for developers to make development easier. It is composed of a source code editor — to write the code —, build automation tools to create the final program, and a debugger to detect errors in the code.

Library: in computer science, collection of resources used by programs. In particular, they contain code and data that provide services to programs.

Main method: in a program written with an object-oriented language, the *main* method is a method — it contains instructions — called *main* and that is executed first: it is given command-line arguments as parameters, and is the entry point to the program.

Manifest file: specific file contained in jar archives which contains data about the archive. For instance, it can specify the Main class — which contains the Main method and is the entry point of an executable JAR archive, a specific environment or required bundles — which are, among others, packages provided together, such as the *Ptidej Tool Suite*.

Object-Oriented Programming: one of the four main fundamental styles of programming, using data structures, which consist of data fields and methods, to design applications and computer programs. Those structures are called *objects*.

Plugin: package that brings new functionalities to a software application. This software application is usually larger and can be completed by the addition of several plugins.

Reverse engineering: process of analyzing a system to understand its internal functioning and its fabrication method.

SFTP: SSH File Transfer Protocol, a network protocol used to access, transfer and manage files. It is an extension of SSH (Secure Shell, a network protocol for secure data communication and remote command execution) that provides secure file transfer capability.

SVN: abbreviation for Subversion, a piece of software for managing versions. It is based on the principle of centralized and unique deposit.

Syntactic analysis: or, less formally, parsing, is in programming the process of analyzing a text file (source code, for example) made of a sequence of tokens (key-words of a programming language, for example) to determine its grammatical structure with respect to a given grammar.

Web service: program that allows communication and data exchange between applications.

Appendix

SQUANER

HOW TO WORK ON SQUANER v3

Angelino

6/5/2012

Last review: 8/22/12

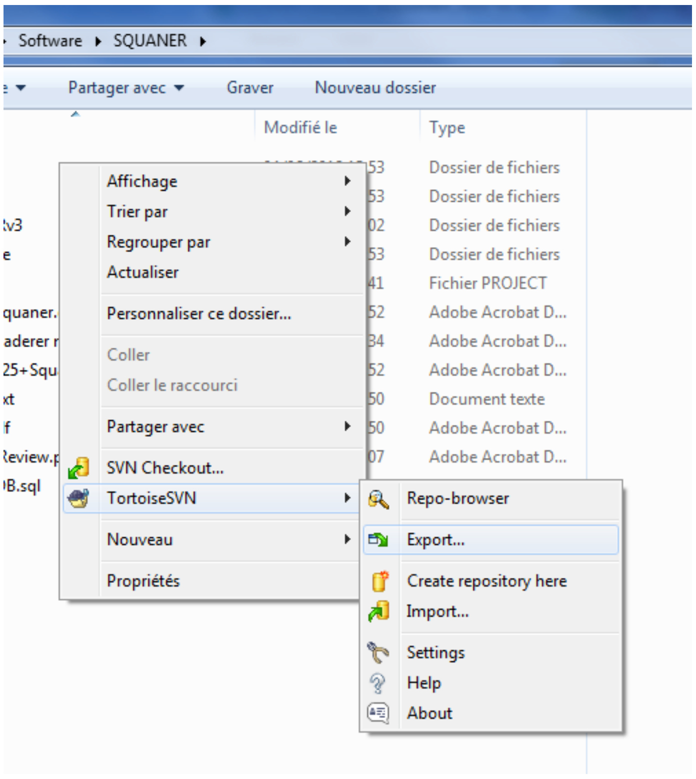
Work Locally

Download the project with TortoiseSVN

First of all, you need to install TortoiseSVN: http://tortoisesvn.net/downloads.html

Then, create a directory on your local machine, where you want to work on Squaner.

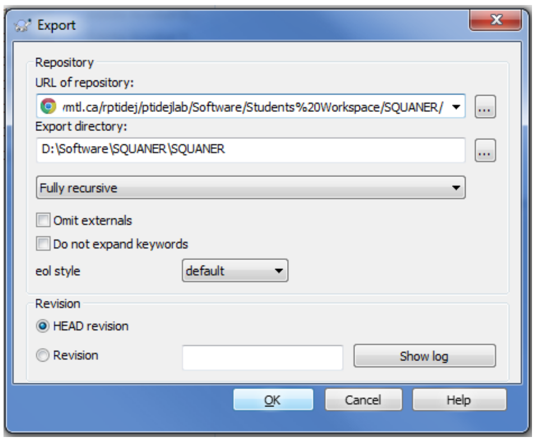
Once in this folder, show the contextual menu (right-click) > TortoiseSVN > Export.



In the *URL of repository* field, paste the following link:

https://web.soccerlab.polymtl.ca/rptidej/ptidejlab/Software/Students%20Workspace/SQUANER/v3/

and press *OK*.


Export

Repository

URL of repository: mtl.ca/rptidej/ptidejlab/Software/Students%20Workspace/SQUANER/

Export directory: D:\Software\SQUANER\SQUANER

Fully recursive

☐ Omit externals

☐ Do not expand keywords

eol style: default

Revision

● HEAD revision

○ Revision

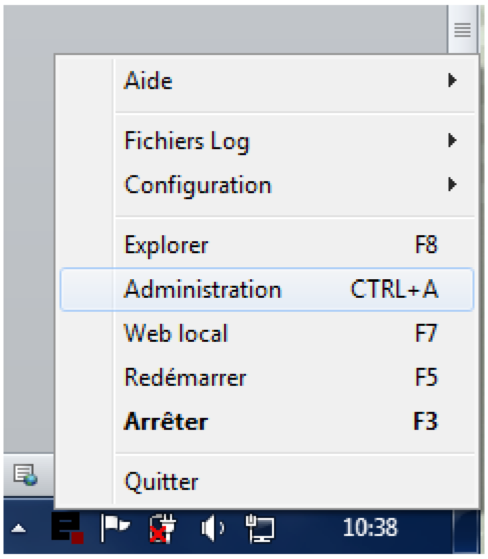
Create a database

You can use a LAMP (if you are under Linux) or a WAMP (if you are under Windows) server to run a local database.

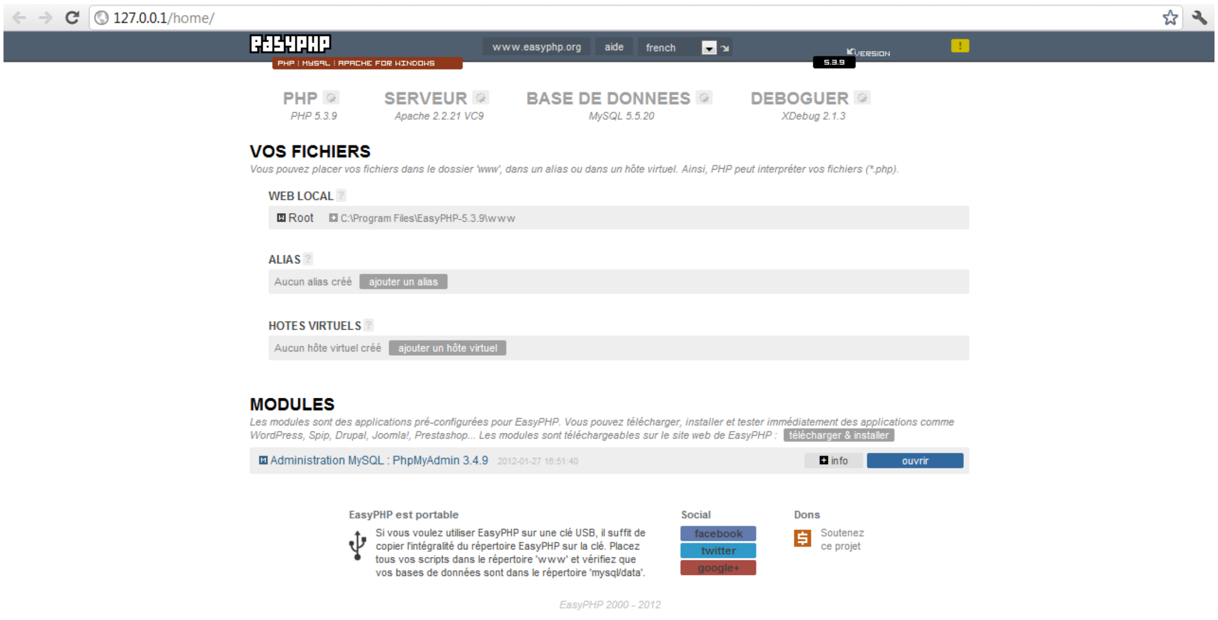
For this example, I will use easyPHP for Windows, but the process remains the same for all platforms.

Download easyPHP: http://www.easyphp.org/download.php

Once installed, right-click on the easyPHP icon in the tray bar, then click *Administration*.



This will open your browser. On the displayed page (127.0.0.1/home), select *Open* next to *MySQL* and *PHPMyAdmin*. This will display another page: your database.



PHP — PHP 5.3.9
SERVEUR — Apache 2.2.21 VC9
BASE DE DONNEES — MySQL 5.5.20
DEBOGUER — XDebug 2.1.3

VOS FICHIERS

Vous pouvez placer vos fichiers dans le dossier 'www', dans un alias ou dans un hôte virtuel. Ainsi, PHP peut interpréter vos fichiers (.php).*

WEB LOCAL

Root C:\Program Files\EasyPHP-5.3.9\www

ALIAS

Aucun alias créé — ajouter un alias

HOTES VIRTUELS

Aucun hôte virtuel créé — ajouter un hôte virtuel

MODULES

Les modules sont des applications pré-configurées pour EasyPHP. Vous pouvez télécharger, installer et tester immédiatement des applications comme WordPress, Spip, Drupal, Joomla!, Prestashop... Les modules sont téléchargeables sur le site web de EasyPHP : télécharger & installer

Administration MySQL : PhpMyAdmin 3.4.9 2012-01-27 16:51:40 — info — ouvrir

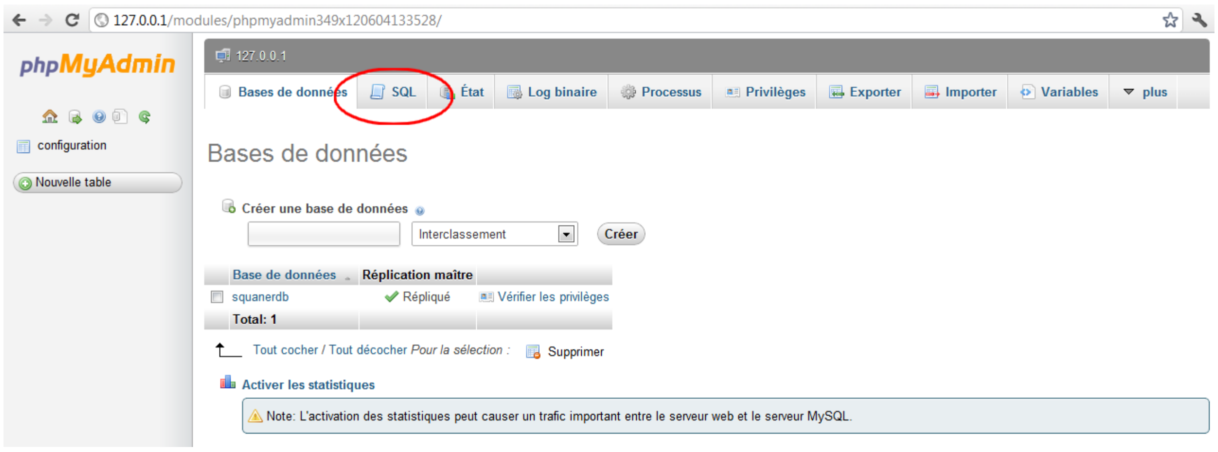
EasyPHP est portable

Si vous voulez utiliser EasyPHP sur une clé USB, il suffit de copier l'intégralité du répertoire EasyPHP sur la clé. Placez tous vos scripts dans le répertoire 'www' et vérifiez que vos bases de données sont dans le répertoire 'mysql/data'.

Social: facebook, twitter, google+

Dons: Soutenez ce projet

EasyPHP 2000 - 2012



Bases de données

Créer une base de données

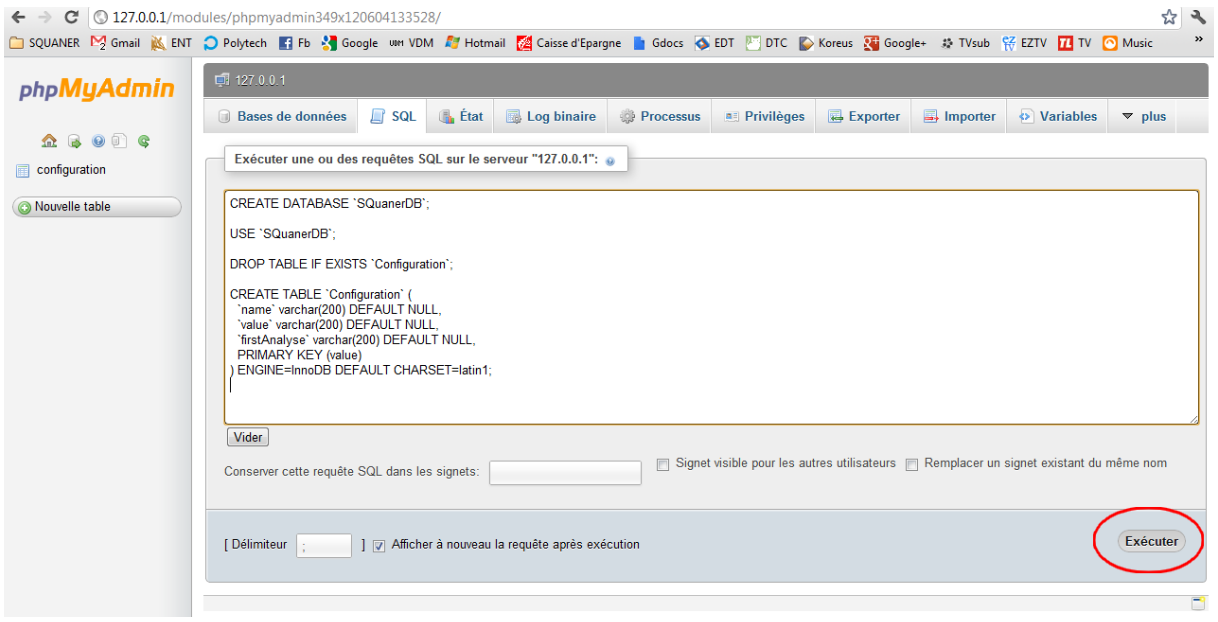
Interclassement | Créer

Base de données	Réplication maître	
squanerdb	✓ Répliqué	Vérifier les privilèges
Total: 1		

Tout cocher / Tout décocher *Pour la sélection :*

Note: L'activation des statistiques peut causer un trafic important entre le serveur web et le serveur MySQL.

Now click on SQL and enter the following code, then click on *Execute*.



Exécuter une ou des requêtes SQL sur le serveur "127.0.0.1":

```
CREATE DATABASE `SQuanerDB`;

USE `SQuanerDB`;

DROP TABLE IF EXISTS `Configuration`;

CREATE TABLE `Configuration` (
  `name` varchar(200) DEFAULT NULL,
  `value` varchar(200) DEFAULT NULL,
  `firstAnalyse` varchar(200) DEFAULT NULL,
  PRIMARY KEY (value)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Conserver cette requête SQL dans les signets: ☐ Signet visible pour les autres utilisateurs ☐ Remplacer un signet existant du même nom

[Délimiteur ;] ☑ Afficher à nouveau la requête après exécution

Important note: You will find the latest SQL request on the following link.

https://web.soccerlab.polymtl.ca/rptidej/ptidejlab/Software/Students%20Workspace/SQUANER/v3/squaner/squanerdb.sql

Adapt the project to your environment

The downloaded project is configured for the Soccerlab server. Let's change some settings!

Open Eclipse IDE (http://www.eclipse.org/downloads/). You'd better use Eclipse JEE which is specific to web-oriented applications.

Select the downloaded workspace as Eclipse's workspace.

You should see on the left several projects. Open SQuaner project, in the sources open the package squaner.configuration and finally the file Enum.java. You must modify the *HOME* variable, typing the absolute path leading to the downloaded squaner folder on your computer.

Now, open the Squaner-web project, the package run and the file RunSquaner.java. Modify the *javapath* variable, typing the absolute path leading to your JRE (Java Runtime Environment) on your computer.

In a file explorer, go to the downloaded *squaner* folder. In the *configuration* folder modify the file *Configuration.ini*. If you did not change anything while installing easyPHP (or whatever server you used), the file should be modified as following:

```
################################################
#*** Data Base Properties
################################################

db.address = localhost
db.port = 3306
db.user = root
db.password =
```

If you still have errors, you may need to add references to several needed projects in the Java Build Path (right-click on the project > Properties > Java Build Path > Projects).

Tomcat server

Finally, you will use a tomcat server to test the project.

Download Tomcat server 6.0.24 (It is very important to use the same version as Soccerlab to avoid compatibility issues while deploying on Soccerlab's server):

http://archive.apache.org/dist/tomcat/tomcat-6/v6.0.24/bin/

Also, you should install the Tomcat plugin for Eclipse.

http://www.eclipsetotale.com/tomcatPlugin.html#A3

Download the archive, and then decompress it in the *plugin* folder of Eclipse. Restart Eclipse.

If you have errors on Squaner-web project, right-click on it > Tomcat > Check *Is a Tomcat project*.

Export the Squaner project

In Eclipse IDE, right-click on the Squaner project > Export > JAR file.

Select the destination folder, and specify squaner.monitoring.synchronized.SQuanerSystem as main class. Call the jar archive Run.jar. Don't forget to check all needed projects in dependencies.

Now, create the WAR archive, which will be used by the TomcatServer to display the web-service.

In Eclipse IDE, right-click on the Squaner-web project > Export > WAR file.

Call the war archive Squaner-web.rar. Don't forget to check all needed projects in dependencies.

Paste the exported Squaner-web.war in the web-apps directory of the Tomcat Server (usually where you installed the server). When you will launch the TomcatServer, it will create a folder named Squaner-web, and you will be able to run the project opening a web browser and typing 127.0.0.1:8080/Squaner-web in the address bar.

Deploy SQUANER on Soccerlab's server

Adapt SQUANER to the target environment

In squaner.configuration.Enum you must modify the *HOME* variable, typing the absolute path leading to the squaner *home* folder on the server: /data/squaner.

Now, open the Squaner-web project, the package run and the file RunSquaner.java. Modify the *javapath* variable, typing the absolute path leading to the JRE (Java Runtime Environment) of the server: /usr/lib/jvm/java/bin/java.

In a file explorer, go to the downloaded *squaner* folder, in the *configuration* folder modify the file *Configuration.ini*. The file should be modified as following:

```
##############################################
#*** Data Base Properties
##############################################

db.address = localhost
db.port = 3306
db.user = root
db.password =
```

Export SQUANER

In Eclipse IDE, right-click on the Squaner project > Export > JAR file.

Select the destination folder, and specify squaner.monitoring.synchronized.SQuanerSystem as main class. Call the jar archive Run.jar. Don't forget to check all needed projects in dependencies.

Thanks to TortoiseSVN or FileZilla Client (establishing a SFTP connexion to web.soccerlab.polymtl.ca and using your SVN identifiers), put the archive Run.jar in /data/squaner/Run.jar

However, saving a copy of the current working version of SQUANER before overwriting it would be an excellent idea… (No, it's an order! ☺)

Export SQUANER-web

Now, create the WAR archive, which will be used by the TomcatServer to display the web-service.

In Eclipse IDE, right-click on the Squaner-web project > Export > WAR file.

Call the war archive Squaner-web.rar. Don't forget to check all needed projects in dependencies.

Thanks to TortoiseSVN or FileZilla Client (establishing a SFTP connexion to web.soccerlab.polymtl.ca and using your SVN identifiers), delete the directory /var/lib/tomcat6/webapps/Squaner-web and the archive /var/lib/tomcat6/webapps/Squaner-web.war, and put the new archive Squaner-web.rar in /var/lib/tomcat6/webapps/Squaner-web.war

However, saving a copy of the current working version of Squaner-web before overwriting it would be, again, an excellent idea… (No, it's still order! ☺)

The Tomcat server should automatically (it takes about 5 seconds) create the folder Squaner-web. If not, ask the technician in charge.

Export the configuration files

Don't forget to put on the server the configuration files: put the content of the squaner folder in /data/squaner/, after making a copy of the working environment.

Export the database

The database is located on https://web.soccerlab.polymtl.ca/phpmyadmin/index.php.

You will find the access identifiers in squaner.software.HistoricalDatabase lines 436 and 437.

If you need to clean the database, delete all databases named after the values of the field *value* in the table *Configuration* in the database *SQuanerDB*. Then you can clear the content of the three tables of *SQuanerDB*.

If needed, you can create the database *SQuanerDB* thanks to the following file:

https://web.soccerlab.polymtl.ca/rptidej/ptidejlab/Software/Students%20Workspace/SQUANER/v3/squaner/squanerdb.sql

Use SQUANER

To use Squaner, open a web browser and go to the following URL:

http://web.soccerlab.polymtl.ca:8080/Squaner-web/

In the Software tab, you can create your account and add a new project to analyze.

You can launch Squaner analyzer following this link http://web.soccerlab.polymtl.ca:8080/Squaner-web/run.jsp, or using Putty: connect to web.soccerlab.polymtl.ca with your SVN identifiers, and use the two following commands :

- cd /data/squaner
- java -cp /data/squaner/Run_lib/mysql-connector-java-5.0.8-bin.jar:/data/squaner/Run_lib/cfparse.jar:/data/squaner/Run_lib/bcel-5.2.jar:/data/squaner/Run_lib/commons-lang-2.4.jar:/data/squaner/Run_lib/bnj33sv.jar:/data/squaner/Run_lib/mail.jar:/data/squaner/Run_lib/com.springsource.javax.activation-1.1.1.jar:/data/squaner/Run_lib/svnkit.jar:/data/squaner/Run_lib/org.eclipse.osgi_3.6.1.R36x_v20100806.jar:/data/squaner/Run_lib/trove-3.0.0a3.jar:/data/squaner/Run_lib/org.eclipse.jdt.core_3.6.1.v_A68_R36x.jar:/data/squaner/Run_lib/org.eclipse.text_3.5.0.v20100601-1300.jar:/data/squaner/Run_lib/org.eclipse.core.runtime_3.6.0.v20100505.jar:/data/squaner/Run_lib/org.eclipse.equinox.common_3.6.0.v20100503.jar:/data/squaner/Run_lib/org.eclipse.core.resources_3.6.0.R36x_v20100825-0600.jar:/data/squaner/Run_lib/org.eclipse.core.jobs_3.5.1.R36x_v20100824.jar:/data/squaner/Run_lib/org.eclipse.core.contenttype_3.4.100.v20100505-1235.jar:/data/squaner/Run_lib/org.eclipse.equinox.preferences_3.3.0.v20100503.jar:Run.jar squaner.monitoring.synchronize.SQuanerSystem

If you launch Squaner *via* the web-service, don't forget to press the Stop button when you are done, or the thread will have to be stopped by command-line: use ps –u tomcat to get the PID, and then kill -9 *PID* to kill it. You may not have the rights to kill a process; then ask your tutor to do so, warning him not to kill the *java* process. If this process is killed, a reboot of the tomcat server will be necessary.

Finally, if you launch Squaner *via* the web-service, you will find the log file in /data/squaner/log.txt

Work on PADL Creator C++

By the time this document is being written, SQUANER only analyzes java projects (java files and class files), and is getting a new feature to analyze C++ projects too: PADL Creator C++.

First, download the current version of SQUANER using PADL Creator C++.

Concept

The idea is to launch Eclipse from SQUANER when a C++ project is about to be analyzed. It will be launched from squaner.parser.padlCreator.CPlusPlusAnalyser, which is used to create an analyzer for a C++ project about to be analyzed.

SQUANER needs to clean the default workspace of the headless Eclipse and put the cpp files of the project to analyze in the project *C++ Project to Analyse*. Then this Eclipse instance is launched headless with PADL Creator C++ as application: the plugin will make Eclipse create an AST view of the C++ and create a PADL model from the AST view. Finally, the plugin must serialize the PADL model and SQUANER has to de-serialize it to set the model.

Prerequisite

You may read documentation on Headless Eclipse on Ptidej's wiki:

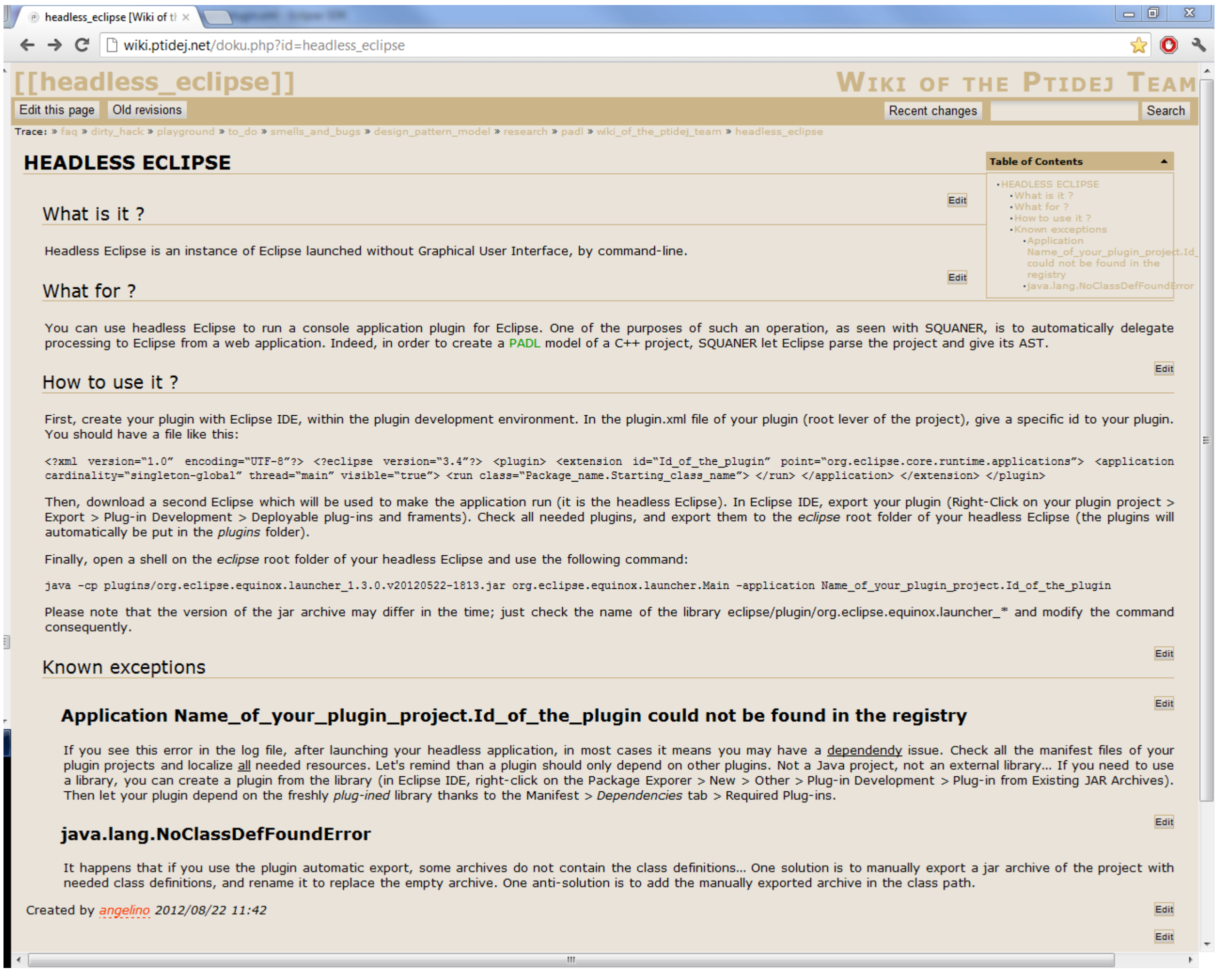
http://wiki.ptidej.net/doku.php?id=headless_eclipse

Tasks

1. Completely debug the plugin PADL Creator C++ so that a PADL model is printed in the console
2. Modify the plugin so that the PADL model is serialized
3. Modify SQUANER so that headless Eclipse is launched with the plugin PADL Creator C++ as application, SQUANER de-serializes the PADL model and put it in *this.codeLevelmodel*.

Documentation on Headless Eclipse

Ptidej's wiki


headless_eclipse [Wiki of th
wiki.ptidej.net/doku.php?id=headless_eclipse
[[headless_eclipse]]
Wiki of the Ptidej Team
Edit this page
Old revisions
Recent changes
Search
Trace: » faq » dirty_hack » playground » to_do » smells_and_bugs » design_pattern_model » research » padl » wiki_of_the_ptidej_team » headless_eclipse
Table of Contents
HEADLESS ECLIPSE
What is it ?
What for ?
How to use it ?
Known exceptions
Application Name_of_your_plugin_project.Id_ could not be found in the registry
java.lang.NoClassDefFoundError
Edit

HEADLESS ECLIPSE

What is it ?

Headless Eclipse is an instance of Eclipse launched without Graphical User Interface, by command-line.

What for ?

You can use headless Eclipse to run a console application plugin for Eclipse. One of the purposes of such an operation, as seen with SQUANER, is to automatically delegate processing to Eclipse from a web application. Indeed, in order to create a PADL model of a C++ project, SQUANER let Eclipse parse the project and give its AST.

How to use it ?

First, create your plugin with Eclipse IDE, within the plugin development environment. In the plugin.xml file of your plugin (root lever of the project), give a specific id to your plugin. You should have a file like this:

```
<?xml version="1.0" encoding="UTF-8"?> <?eclipse version="3.4"?> <plugin> <extension id="Id_of_the_plugin" point="org.eclipse.core.runtime.applications"> <application
cardinality="singleton-global" thread="main" visible="true"> <run class="Package_name.Starting_class_name"> </run> </application> </extension> </plugin>
```

Then, download a second Eclipse which will be used to make the application run (it is the headless Eclipse). In Eclipse IDE, export your plugin (Right-Click on your plugin project > Export > Plug-in Development > Deployable plug-ins and framents). Check all needed plugins, and export them to the *eclipse* root folder of your headless Eclipse (the plugins will automatically be put in the *plugins* folder).

Finally, open a shell on the *eclipse* root folder of your headless Eclipse and use the following command:

```
java -cp plugins/org.eclipse.equinox.launcher_1.3.0.v20120522-1813.jar org.eclipse.equinox.launcher.Main -application Name_of_your_plugin_project.Id_of_the_plugin
```

Please note that the version of the jar archive may differ in the time; just check the name of the library eclipse/plugin/org.eclipse.equinox.launcher_* and modify the command consequently.

Known exceptions

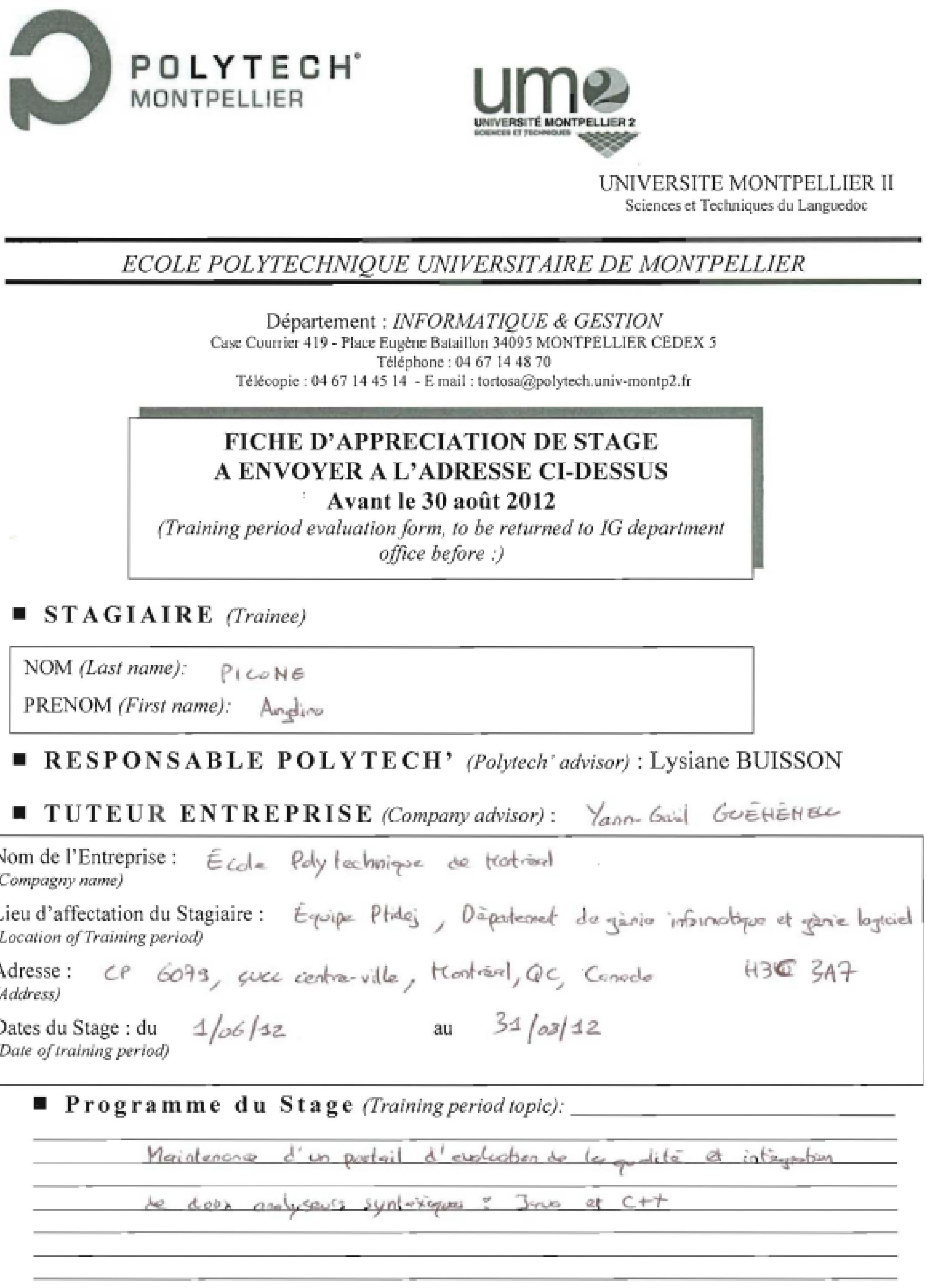
Application Name_of_your_plugin_project.Id_of_the_plugin could not be found in the registry

If you see this error in the log file, after launching your headless application, in most cases it means you may have a dependendy issue. Check all the manifest files of your plugin projects and localize all needed resources. Let's remind than a plugin should only depend on other plugins. Not a Java project, not an external library... If you need to use a library, you can create a plugin from the library (in Eclipse IDE, right-click on the Package Exporer > New > Other > Plug-in Development > Plug-in from Existing JAR Archives). Then let your plugin depend on the freshly *plug-ined* library thanks to the Manifest > *Dependencies* tab > Required Plug-ins.

java.lang.NoClassDefFoundError

It happens that if you use the plugin automatic export, some archives do not contain the class definitions... One solution is to manually export a jar archive of the project with needed class definitions, and rename it to replace the empty archive. One anti-solution is to add the manually exported archive in the class path.

Created by *angelino 2012/08/22 11:42*



UNIVERSITE MONTPELLIER II
Sciences et Techniques du Languedoc

ECOLE POLYTECHNIQUE UNIVERSITAIRE DE MONTPELLIER

Département : *INFORMATIQUE & GESTION*
Case Courrier 419 - Place Eugène Bataillon 34095 MONTPELLIER CEDEX 5
Téléphone : 04 67 14 48 70
Télécopie : 04 67 14 45 14 - E mail : tortosa@polytech.univ-montp2.fr

FICHE D'APPRECIATION DE STAGE
A ENVOYER A L'ADRESSE CI-DESSUS
Avant le 30 août 2012
(Training period evaluation form, to be returned to IG department office before :)

■ **STAGIAIRE** *(Trainee)*

NOM *(Last name)*: PICONE
PRENOM *(First name)*: Angelino

■ **RESPONSABLE POLYTECH'** *(Polytech' advisor)* : Lysiane BUISSON

■ **TUTEUR ENTREPRISE** *(Company advisor)* : Yann-Gaël GUÉHÉNEUC

Nom de l'Entreprise : *(Compagny name)* École Polytechnique de Montréal

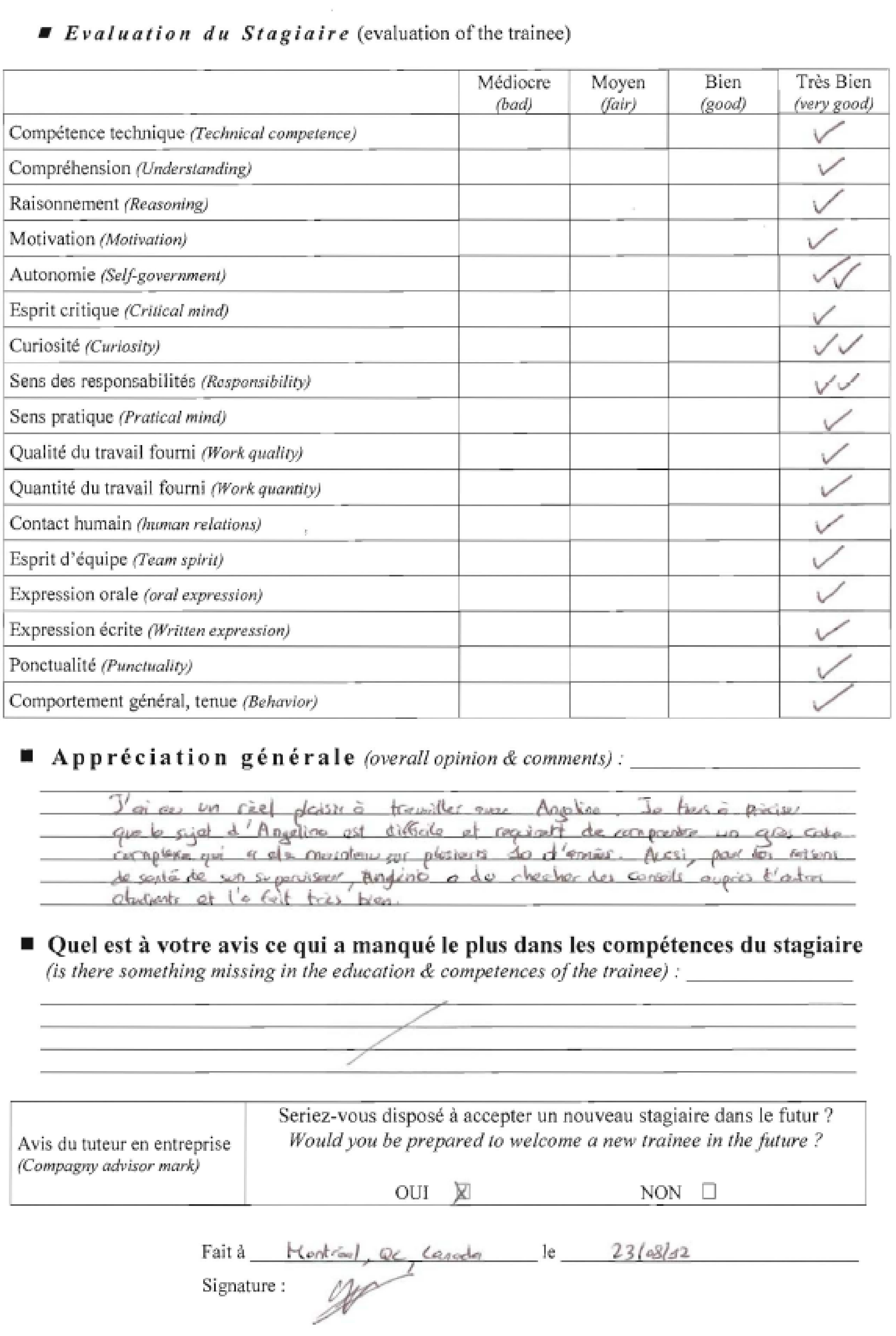
Lieu d'affectation du Stagiaire : *(Location of Training period)* Équipe Ptidej, Département de génie informatique et génie logiciel

Adresse : *(Address)* CP 6079, succ centre-ville, Montréal, QC, Canada H3C 3A7

Dates du Stage : du *(Date of training period)* 1/06/12 au 31/08/12

■ **Programme du Stage** *(Training period topic)*:

Maintenance d'un portail d'évaluation de la qualité et intégration de deux analyseurs syntaxiques : Java et C++



■ **Evaluation du Stagiaire** (evaluation of the trainee)

	Médiocre *(bad)*	Moyen *(fair)*	Bien *(good)*	Très Bien *(very good)*
Compétence technique *(Technical competence)*				✓
Compréhension *(Understanding)*				✓
Raisonnement *(Reasoning)*				✓
Motivation *(Motivation)*				✓
Autonomie *(Self-government)*				✓✓
Esprit critique *(Critical mind)*				✓
Curiosité *(Curiosity)*				✓✓
Sens des responsabilités *(Responsibility)*				✓✓
Sens pratique *(Pratical mind)*				✓
Qualité du travail fourni *(Work quality)*				✓
Quantité du travail fourni *(Work quantity)*				✓
Contact humain *(human relations)*				✓
Esprit d'équipe *(Team spirit)*				✓
Expression orale *(oral expression)*				✓
Expression écrite *(Written expression)*				✓
Ponctualité *(Punctuality)*				✓
Comportement général, tenue *(Behavior)*				✓

■ **Appréciation générale** *(overall opinion & comments)* :

J'ai eu un réel plaisir à travailler avec Angelino. Je tiens à préciser que le sujet d'Angelino est difficile et requiert de comprendre un gros code complexe qui a été construit sur plusieurs dizaines d'années. Aussi, par des raisons de santé de son superviseur, Angelino a dû chercher des conseils auprès d'autres étudiants et l'a fait très bien.

■ **Quel est à votre avis ce qui a manqué le plus dans les compétences du stagiaire**
(is there something missing in the education & competences of the trainee) :

Avis du tuteur en entreprise *(Compagny advisor mark)*	Seriez-vous disposé à accepter un nouveau stagiaire dans le futur ? *Would you be prepared to welcome a new trainee in the future ?* [✓ OUI] NON ☐

Fait à Montréal, QC, Canada le 23/08/12

Signature : [illegible]

Résumé

Ce rapport relate le stage de 4ème année d'Angelino PICONE, élève ingénieur à Polytech'Montpellier en Informatique et Gestion. Ce stage s'est déroulé de juin à août 2012 à l'Ecole Polytechnique de Montréal, au sein du département de Génie Informatique. Dans le laboratoire Ptidej, dirigé par Yann-Gaël Guéhéneuc, le stagiaire a participé à la mise en place et à l'amélioration de SQUANER, une application web de mesure de la qualité.

Des études ont montré que couramment la maintenance représente plus de la moitié du coût total du développement d'un logiciel. Dans un contexte de réduction des coûts liés à la durée de vie d'un projet, SQUANER aide à améliorer la qualité de ce projet durant son élaboration et amène à réduire significativement le temps lié à sa maintenance.

Le rapport présente les entités engagées, les enjeux et les objectifs du stage. Enfin le travail effectué est présenté en deux parties puis est analysé. Chaque partie présente le contexte et détaille les étapes suivies.

Mots-clés

Qualité, Application Web, Analyse, Maintenance, Développement, Java EE, Headless Eclipse, Apache Tomcat.

Abstract

This report deals with the 4th year internship of Angelino PICONE, engineer student at Polytech'Montpellier in Computer Science and Management Information Systems. This internship took place from June to August 2012 at the Ecole Polytechnique de Montréal, within the Department of Computer Engineering and Software Engineering. In the laboratory Ptidej, leaded by Yann-Gaël Guéhéneuc, the intern took part in the settlement and the improvement of SQUANER, a quality measurement web application.

Studies have shown that maintenance usually represents more than half the total cost of software development. In the context of reducing costs related to the life time of a project, SQUANER helps to improve the quality of the project during its development and leads to significantly reduce the time associated with maintenance.

The report presents the entities involved, the issues and objectives of the internship. Finally the work is presented in two parts and then analyzed. Each section presents the context and details the steps followed.

Key words

Quality, Web application, Analysis, Maintenance, Development, Java EE, Headless Eclipse, Apache Tomcat.