"Projet Long" Report - Version 1

Zohour Abouakil
Sofia Boutahar
David Courtinot
Xiaowen Ji
Fabien Sauce

Reference : model-checking.final-report
February, 24^{th} 2015

Signatures

Project manager - Zohour Abouakil :
Quality responsible - David Courtinot :
Customers - David Doose - Julien Brunel :

Contents

Acknowledgments

Apart from our personal efforts, the success of any project depends largely on the encouragement and guidelines of many other persons either from the clients or our school professors. We would like to express the deepest appreciation to all our professors at ENSEEIHT who introduced us to Computer Science Engineering and helped us to acquire solid technical skills and widen our knowledge by attending very interesting courses and working on many exciting projects during this three years.

We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. I would like to gratefully acknowledge the enthusiastic supervision of Mr. David Doose and Mr. Julien Brunel, both engineering researchers at ONERA, who had been a source of inspiration and were always guiding us and giving us useful suggestions which helped us in completing the project work. They always did there best to respond promptly and enthusiastically to all our requests, despite their congested daily schedule. I would also like to thank Mr. Jean francois Coiffin, our industrial partner, who helped us to manage the project, raise awareness with problems and methods for this activity and also ensure quality control.

Finally, the reception of ENSEEIHT, which provided us every day with a room to work in all day long.

Introduction

This report deals with the work we did from the end of January, in the context of the "projet long". It is proposed by Mr. David Doose and Mr. Julien Brunel from ONERA, the French Aerospace Lab, and deals with pattern recognition in C++ code. We will describe the project management methods that we used and then we will present the technical aspects of our project.

Our team consists of five ENSEEIHT students (Zohour Abouakil, Fabien Sauce) from the computer science course and (Sofia Boutahar, David Courtinot, Xiaowen Ji) from the imagery and multimedia course, two clients (Mr. David Doose and Mr. Julien Brunel) and an industrial partner from Astrium (Mr. Jean Francois Coiffin)

As third year engineering students in Computer science and applied mathematics, we are interested in groundbreaking technologies. Part of our degree, our final year project has been the right place to get in touch with a lot of new technologies and get in touch with very skilled and professional persons by working on an innovant and ambitious project.

It was the opportunity to discover and set up project management systems that are necessary to respect all the deadlines.

Part I

Project presentation

I.1 Overview

To get our ENSEEIHT engineering diploma, we are required to take part in a project called "Projet long" in teams of five students to work on a common project.The project started on January 19, and will last eight weeks. It ends up with a defense in which we promote our work in front of a jury which evaluates us against different aspects :

- Project management and organization
- Technical accomplishment
- Report and defense presentation
- English evaluation

All over the project, we have to work side by side with the client for whom we have to deliver, at the end of the project, a product that suits their expectations. Furthermore, we are also supervised by Mr. Jean-Francois COIFFIN. He is in charge of helping us through his experience in the project management and organization.

We chose to work on that project because of the originality of the subject, since it is mixing theoretic computer science and technical advanced principles. Moreover, studying model checking and temporal logic to assert properties on a source code was a topic that we studied in ENSEEIHT courses. This project is an opportunity to apply this theory and dive deeper into it.

I.2 Subject

I.2.1 Main idea

The client is waiting for a prototype that allows a search of patterns on a C++ code. The patterns will be expressed in terms of temporal logic properties.

I.2.2 Project description

Embedded systems and robotics are designed to interact with humans. Therefore, a single failure or a malfunction can really be catastrophic. This is why various analyses are undertaken to limit and prevent such problems. Theses analysis aim to study the embedded code and prove that it does what it is supposed to do. The goal of our project is to find out whether the embedded code meets a number of programming rules by

defining authorized and prohibited patterns. Our clients have an existing tool named Coccinelle that is developed at INRIA. This tool detects patterns and also offers the possibility to modify the code. However, this tool only works on C code. The objective of the project is to design a prototype for pattern matching in a C++ code.

I.3 Objectives

From a pedagogical point of view, this project was the opportunity to apply a lot of techniques that we saw in our three years of courses in ENSEEIHT and new techniques that we learned while working on it. We took our project management courses as a reference to organize our time and to catch up with the deadlines.

To deliver a good product at the end, we realized that the good coordination in the team, the regular exchange of ideas during meetings and code/design reviews and production of relevant documents are main keys of success.

From the technical view, the clients expect us to design and create a tool written in Scala that would detect patterns in a C++ code using temporal logic expressions. They specified that our project would be a combination of two main parts:

- **C++ parsing and transformation:**
  In this part:
  - We take a C++ source code file as an input
  - We parse the file generated by the clang compiler to get its AST representation
  - The result of the AST parsing is a raw data that has to be structured and used for checking some code properties.
  - Finally, we transform the AST into a graph structure CFG that is traversable by a model checking algorithm

- **Model checking:** The model checker takes a temporal logic expression from the user input that respects the CTL syntax and then it marks all the nodes in the graph that verify this logical expression. We will have to implement an extension of CTL, CTL-V, which allows us to quantify the meta-variables as well.

I.4 Constraints

Part II

Project management

II.1 Team organization

The way a project team is structured can play a major role in how it functions. Team structure will probably be adjusted at each stage to meet the evolving nature of the project. Building a good, effective team is vital. Team structure will influence the way the team behaves. It aims to create a collaborative team, where individuals share knowledge, co-operate, support each other and are motivated to achieve the team's goals.

- **Project manager:** The project manager is primarily concerned about communications with the industrial and the customers. He has a leading role in the organization and planning of the tasks. A project manager is the person responsible for accomplishing the stated project objectives including creating clear and attainable project objectives, building the project requirements, and managing cost, time, scope, and quality. He is often a client representative and has to determine and implement the exact needs of the client, based on knowledge of the firm they are representing.
- **Supervisor:** The supervisor has a global technical view of the project. He supervises the advancement of simultaneous tasks. Otherwise, he can rearrange groups and objectives if an unforeseen occurs. The supervisor has also to participate in coding or documenting an assigned task. Nevertheless, it is not his primary function.The team supervisor can change from one week to another.
- **Quality manager:** The quality manager is in charge of checking that every deliverable documents meets the quality standards. In other words, any produced code will pass under the watchful eye of the quality manager before being validated. He also ensures the quality and consistency of all documents produced by the team.
- **Test manager:** The test manager is responsible of the validation and testing in global environment written by the developers (each developer has its own set of unit tests). He does not only run tests, he also determines whether the tests are complete or not (code coverage).
- **Configuration manager:** The configuration manager should take care of every tool we are going to use, make some choice about which tools are better than other (example : Scalastyle, an Eclipse plugin that we use for automated quality checs). In particular he will handle the installation and the follow up of a version tool as Github for example.

The figure shows the team organization as we decided to do it in our team.

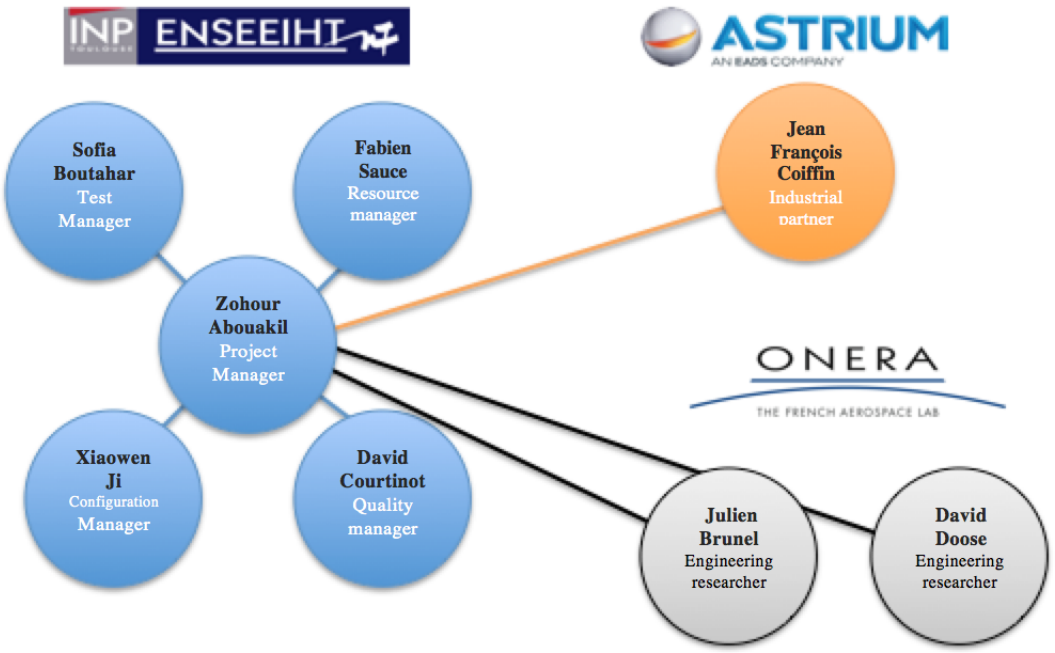

Figure II.1 - The team organization of our project

Although everybody was involved in the same manner in searching , coding and testing stages, we gave everyone a role in order to improve team organization. The project manager (Zohour Abouakil) has several responsabilities related to the project management. She is in charge of the communication with the industrial coordinator, Mr. Jean Francois Coiffin and the scheduling of the different appointments . Fabien Sauce is in charge of the specifications and the documentation. Xiaowen is in charge of the version tool Git and takes care of all the software we use in the Project. Sofia Boutahar is responsible for the unitary tests and the regressions tests when there is a new version of the code or a part that is fully developped.

In addition to being the project manager, Zohour establishes a contact with the client to see if the team is going in the right direction or needs additional information.

II.2 Project objectives in terms of management and organization

- Managing coordination of the partners and working groups engaged in project work.
- Writing a detailed project planning including:
  - Developing and maintaining a detailed project plan.
  - Managing project deliverables in line with the project plan.
  - Recording and managing project issues and escalating where necessary.
  - Resolving issues and try to prevent them.
- Managing project scope and change control and escalating issues where necessary.
- Monitoring project progress and performance.
- Managing project evaluation and dissemination activities.

- Final approval of the design specification.
- Working closely with clients to ensure the project meets business needs.
- Definition and management of the testing program.

II.3 Deliverable documents

II.3.1 Delivrable documents expected by ENSEEIHT and the industrial supervisor

- Report in PDF format
- Development plan in PDF format
- Presentation supports

II.3.2 Delivrable documents expected by the client

- Documented source code in Scala language
- Test strategy
- Architecture design document

II.4 Development plan

II.4.1 Development organization

We used the Scrum method, which is widely used, and recognized for its effectiveness. At first, we defined a product backlog containing all desired functionalities in the final product. In fact, this report is also a part of the product backlog. Next, we divided the project into three sprints (which means iterations). A sprint backlog is defined for each sprint, including all we need to realize at the end of an iteration. Each sprint lasts two weeks and lies in improve the software incrementally, so that it is close to product backlog. At the end of each sprint, we organised a meeting, in order to review the progress and propose improvements or modifications of planning, but in the process of a sprint, we cannot modify the sprint backlog. To finish, each day starts with a scrum meeting, on the meeting, each team member present his objective of the day and his actual difficulties.

II.4.2 Team organization approach

We will use an approach inspired by the XP (EXTREME PROGRAMMING) method which is a practice of pair programming. Considering the amount of code that we will have to write, we find it unnecessary that the five team members work separately, and we consider as excellent to work in pairs, in order to prevent errors and bias of the program structure, so that we can save times in testing and debugging. As a consequence, four of us will work in pairs and the last one works individually or supervises us. The groups repartition may change as the tasks are completed.

II.4.3 Tasks organization

Task definition

The sprint backlog is a list of tasks that are identified by the members of the project and that has to be completed during the Scrum sprint. During our meetings, we try to estimate how many hours and development

efforts needed to complete a task.
Sprint 1 backlog :

- AST parsing of procedure C++ code
- CFG conversion from parsed AST
- Model checking with simple properties

Sprint 2 backlog :

- AST parsing of object oriented C++ code
- CFG conversion from parsed AST
- Model checking with simple criteria

Sprint 3 backlog :

- Improved CFG conversion from parsed AST
- Model checking with complex criteria

Task planning

Our sprint backlog was maintained as a spreadsheet. During the Scrum sprint, each team member is requested to keep the sprint backlog updated.

II.5 Risk management

II.5.1 Risk management strategy

The first step in project risk management is to identify the risks that are present in the project. Also, some risks have a higher impact than others. Therefore, we spend our time on the risks that can cause the biggest losses and gains.

II.5.2 Risk analysis

Below a table that summarizes the major risks that we could face in our project and how we planned to prevent them.

Date	Risk description	Consequences	Type of risk	Probability (1 to 5)	Impact level (1-5)	Weight	Preventive measure
January 27th	Communication problems: lack of communication, misunderstandings, etc.	Unproductive group, non-respect of the interfaces necessary to compatibility	Human resources	5	5	25	Be sure we agreed with our colleagues before starting a part
January 27th	Underestimation of the development time	Deadline exceeded / late delivery	Schedule	4	5	20	Supervisor able to switch from one task to another and have a global vision
January 27th	Customer's requirements not respected	Product not accepted by the client	Clients requirements	4	4	16	Validate the conception by the client
January 27th	Bad design choices at the beginning, issues to make the model evolve, corner cases.	Problem to make the project evolve, waste of time to readapt the conception to the new requirements	Quality	3	5	15	Allocate several days to conception and ensure everyone is convinced by the design
January 27th	Health problems: a member of the team getting sick	In the best case, redefine the other team member role. Otherwise, the product will be late	Schedule	2	5	10	Flexible schedule
January 27th	Underestimation of the learning curve, different time learning among the team	Delay, different rhythms for the various parts of the project	Schedule	3	3	9	Create balanced teams (people more experienced with less experienced)
January 27th	Appearance of recalcitrant bugs	Unable to meet certain requirements	Quality	2	4	8	Use of the scrum method, incremental test
January 27th	Wrong or inappropriate assumptions during the analysis	Unexpected edge cases difficult to handle with our model	Development method	5	4	20	Validate the conception by the client

Figure II.2 - The risk analysis for our project

II.6 Resource management system

II.6.1 Versioning tool

We use Git to manage our project especially the versioning of the code and the documents that we develop. Our choice was made because Git is a free and open source distributed version control system that handles any software project in a very efficient way. It also supports rapid branching and merging, and includes specific tools for visualizing and navigating through the development history. All the delivrable documents are managed on a git repository, including documentation and reports. Anyone is allowed to commit at anytime, however any push must have been authorized by the quality responsible after the code has been thoroughly tested against a set of tests by the test responsible.

II.6.2 Communication between team members

We used Google Drive to share all the documents between the team members. Our drive was organized by folders: one with all the papers and the interesting documents that we found or we received from our clients,

one that contains all the documents that are related to the project management, one with the notes that we take during the meetings either with the clients or the industrial coordinator, and the last one that contains all the useful links to have deeper knowledge about the project's subject or the technologies used.

Part III

Code and documentation management

III.1 Quality checking

For ensuring that our coding rules are respected and evaluating the quality of our sources, we used a tool called Scalastyle that enables , using an easy-to-use xml configuration file, to examine the scala code and indicates potential problems with it. Combined with a specific pulgin, this can be used to generate warnings or errors in the IDE the developer is using. Our settings are the following :

Rule	Description	Value
FileLengthChecker	Check the number of lines in a file	1500
FileLineLengthChecker	Check the number of characters in a line	140
FileTabChecker	Check that there are no tabs in a file	enabled
ClassNamesChecker	Check that class names match a regular expression	^[A-Z][A-a-z]*$
ClassTypeParameterChecker	Checks that type parameter to a class matches a regular expression	^[A-Z_]$
FileTabChecker	Check that there are no tabs in a file	enabled
CyclomaticComplexityChecker	Checks that the cyclomatic complexity of a method does exceed a value	12
EmptyClassChecker	If a class/trait has no members, the braces are unnecessary	enabled
EqualsHashCodeChecker	Check that if a class implements either equals or hashCode, it should implement the other	enabled
MethodLengthChecker	Checks that methods do not exceed a maximum length	50
MethodNamesChecker .	Check that method names match a regular expression	^[a-z][A-Za-z0-9]*(_=)?$
MultipleStringLiteralsChecker	Checks that a string literal does not appear multiple times	allowed = 2
NotImplementedErrorUsage	Checks that the code does not have ??? operators	enabled
NullChecker	Check that null is not used	enabled
NumberOfMethodsInTypeChecker	Check that a class/trait/object does not have too many methods	maxMethods = 30

NumberOfTypesChecker	Checks that there are not too many types declared in a file	maxTypes = 20
ObjectNamesChecker	Check that object names match a regular expression	^[A-Z][A-Za-z]*$
ParameterNumberChecker	Maximum number of parameters for a method	maxParameters = 5
RedundantIfChecker	Checks that if expressions are not redundant, ie easily replaced by a variant of the condition	enabled
ScalaDocChecker	Checks that the ScalaDoc on documentable members is well-formed	enabled

This project meets a need from our clients. Therefore our codes will probably be used, studied and modified. That is why we have set up this quality approach. All our codes should be understandable and well commented. To achieve that, our programs were constantly reviewed by our clients as we gave them the link to our Git repository.

III.2 Verification and validation process

III.2.1 Verification process

Verification is the process in which we determine whether the right solution is being developed. Once a functionality of our product is completed, we analyze the results in order to verify that the requirements have been satisfied. In addition, the verification is an ongoing process: at the each completion of each milestone, we performed a verification analysis to make sure we are still on track.

As we have chosen the Extreme Programming model for the programming aspect of the project, we considered that a code would have passed the quality test if at least the two members of a pair have checked it. This is up to the quality manager to ensure this has been done, otherwise he should do it himself. This is specific to the code quality checks and does not apply to the rest of the delivrable documents.

III.2.2 Validation process

Validation is the process in which we make sure that the solution that we came up with is constructed correctly and according to the requirements and the specifications. In this process, we performed various testing procedures on the code and tried to remove the defects as soon as possible. Once the found defects have being fixed, the process repeats itself.

Part IV

Technical aspects

IV.1 Context

IV.1.1 Motivations

Embedded systems and robotics are designed to interact with humans sometimes. Therefore, a single failure or a malfunction can really be catastrophic. This is why various analyses are undertaken to limit and prevent such problems. Theses analysis aim to study the embedded code and prove that it does what it is supposed to do. The goal of our project is to find out whether the embedded code meets a number of programming rules by defining authorized and prohibited patterns. Our clients have an existing tool named Coccinelle that is developed at INRIA. This tool detects patterns and also offers the possibility to modify the code. However, this tool only works on C code.

IV.1.2 Objectives

The model checking, which consists in asserting properties on a model thanks to graph search algorithms (for example), is one of those fields that can be applied to this matter. In this project, we are trying to build a model checker working on C++ code which takes the source code as an input and is transformed a few times in various abstract representations to end with a graph model that we are able to send to a model checker.

IV.2 Definitions

IV.2.1 The AST - Abstract Syntax Tree

The AST is an abstract (and low-level) representation of the abstract syntactic structure of the source code. It is a tree data-structure which describes the code in a purely syntactic point of view. Each node of the tree denotes a construct occurring in the source code. The syntax is "abstract" because it's not representing every detail appearing in the real syntax. However, the AST can contain additional data as the position of an element in the source code node. You can see below a simple C/C++ code and its AST representation. The AST is provided by the Clang API, which performs the first step of our transformation chain.

```
int main() {
    int a = 5;
    if (a > 6) {
        int c = 5;
        c *= 5;
    }
    int b = 17;
}
```

```
TranslationUnitDecl 0x1028254d0 <<invalid sloc>>
|-TypedefDecl 0x102825a10 <<invalid sloc>> __int128_t '__int128'
|-TypedefDecl 0x102825a70 <<invalid sloc>> __uint128_t 'unsigned __int128'
|-TypedefDecl 0x102825e30 <<invalid sloc>> __builtin_va_list '__va_list_tag [1]'
`-FunctionDecl 0x102825ed0 <../../ModelChecker/unitary_tests/ast/if/if.cpp:1:1, line:8:1> main 'int (void)'
  `-CompoundStmt 0x10286df40 <line:1:12, line:8:1>
    |-DeclStmt 0x102826058 <line:2:5, col:14>
    | `-VarDecl 0x102825fe0 <col:5, col:13> a 'int'
    |   `-IntegerLiteral 0x102826038 <col:13> 'int' 5
    |-IfStmt 0x10286de70 <line:3:5, line:6:5>
    | |-<<<NULL>>>
    | |-BinaryOperator 0x1028260d0 <line:3:9, col:13> '_Bool' '>'
    | | |-ImplicitCastExpr 0x1028260b8 <col:9> 'int' <LValueToRValue>
    | | | `-DeclRefExpr 0x102826070 <col:9> 'int' lvalue Var 0x102825fe0 'a' 'int'
    | | `-IntegerLiteral 0x102826098 <col:13> 'int' 6
    | |-CompoundStmt 0x10286de48 <col:16, line:6:5>
    | | |-DeclStmt 0x102826188 <line:4:9, col:18>
    | | | `-VarDecl 0x102826110 <col:9, col:17> c 'int'
    | | |   `-IntegerLiteral 0x102826168 <col:17> 'int' 5
    | | `-CompoundAssignOperator 0x10286de10 <line:5:9, col:14> 'int' lvalue '*=' ComputeLHSTy='int' ComputeResultTy='int'
    | |   |-DeclRefExpr 0x1028261a0 <col:9> 'int' lvalue Var 0x102826110 'c' 'int'
    | |   `-IntegerLiteral 0x1028261c8 <col:14> 'int' 5
    | `-<<<NULL>>>
    `-DeclStmt 0x10286df28 <line:7:5, col:15>
      `-VarDecl 0x10286deb0 <col:5, col:13> b 'int'
        `-IntegerLiteral 0x10286df08 <col:13> 'int' 17
```

Figure IV.1 - The AST of the C++ code generated by the clang API

The indentation in the AST is the depth of a node. Nodes that have the same level of indentation are brothers. Each node can have children that are separated by a new line and a |_. We should notice that the last child of a node starts it's representative line by `_ . For example the TranslationUnitDecl has four children : the first three childs are TypedefDef and the last one is a FunctionDecl. The toplevel declaration in a translation unit is always the translation unit declaration. In this example, our first user written declaration is the declaration of the function "main" that contains a declaration of the variable a and then an If statement. The IfStmt (if statement) has a condition and a body. The condition is described in this example by the binary operator > that has two childs which are the operands a and 6. The body of the IfStmt is described by the compound statement which is a combination of two or more simple statements. In this case, it's a combination of a declStmt (declaration statement) in which the declaration and initialization of the variable c to 5 is described and a compoundAssignOperator in which the assignment operation is described. In the end, there is a DeclStmt that describes the declaration and initialization of the variable b to 17.