Plan de développement

Zohour Abouakil
Sofia Boutahar
David Courtinot
Xiaowen Ji
Fabien Sauce

Recherche de motifs dans un code C++ à l'aide de la logique temporelle

Sommaire

I Project description and objectives **2**
- I.1 Surroundings of the project . . . 2
- I.2 Project description . . . 2
  - I.2.1 Main idea . . . 2
  - I.2.2 Related technologies . . . 2
  - I.2.3 Project parts . . . 2
  - I.2.4 To conclude . . . 3
- I.3 Final project . . . 3
  - I.3.1 Define priorities . . . 3
  - I.3.2 Deliverable documents . . . 3

II Project organization **4**
- II.1 Role definition . . . 4
- II.2 Development organization . . . 5
  - II.2.1 Use of a software development framework : Scrum . . . 5
  - II.2.2 Team repartition approach . . . 5
- II.3 Tasks organization . . . 5
  - II.3.1 Tasks definition . . . 5
  - II.3.2 Planning . . . 5

III Risk management **8**

IV Code management **9**
- IV.1 Quality management . . . 9
  - IV.1.1 Automated coding style checks . . . 9
- IV.2 Test strategy . . . 10
- IV.3 Configuration management . . . 10

V Appendices **11**

Part I

Project description and objectives

I.1 Surroundings of the project

Le projet long à l'ENSEEIHT Organisation du projet

Le client c est qui ? ? Les noms, leurs fonctions, les motivations du projet

Nos motivations – pas sur

I.2 Project description

I.2.1 Main idea

I.2.2 Related technologies

- Coccinelle
- Clang

I.2.3 Project parts

- Parser
- CTL
- Model checking

I.2.4 To conclude

I.3 Final project

I.3.1 Define priorities

I.3.2 Deliverable documents

Part II

Project organization

II.1 Role definition

Project manager

Quality manager

Test manager

Test manager

Configuration manager

Documentation manager

Chain development

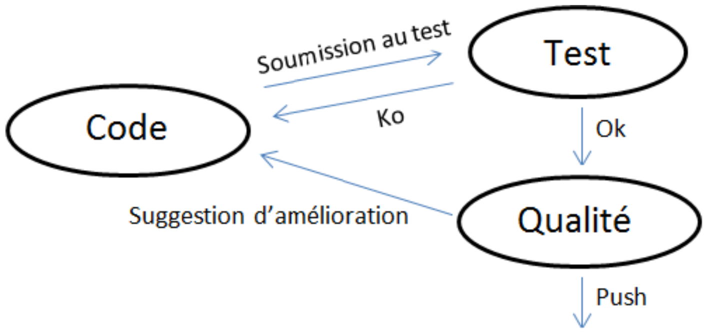


Figure II.1 - Schéma descriptif de la chaîne de développement

II.2 Development organization

To secure our evolution we can use :

II.2.1 Use of a software development framework : Scrum

II.2.2 Team repartition approach

II.3 Tasks organization

II.3.1 Tasks definition

II.3.2 Planning

Tâches à réaliser	21-janv	22-janv	23-janv	26-janv	27-janv	28-janv	29-janv	30-janv	02-févr	03-févr	04-févr	05-févr	06-févr	09-févr	10-févr	11-févr	12-févr	13-févr
Rédaction des spécifications				Z D X	Z D X	Z D X												
Formation :																		
- Scala	Z X F S	Z X F S	Z X F S	Z X F S	Z X F S	Z X F S	Z X F S	Z X F S										
- CTL et model checking	D	D	Z D X F S	Z D X F S	Z D X F S													
Tâches récurrentes :																		
- Design review				Z D X F	Z D X F	Z D X F												
- Code review														Z D X F S				
Itération 1 :																		
- Implémentation Model Checking 1						Z F	Z F	Z F	Z D F	Z D F	Z D F	Z D	Z D	Z D	Z D	Z F		
- Implémentation Parsing AST 1						D X	D X	D X	X	X	X	X	X	X	X	X		
- Implémentation conversion AST to GFC 1									D S	D S	D S	D S	D S	D S	D S	D S		
- Validation séparée :																		
- jeu de tests MC 1																	Z F	Z F
- jeu de tests AST/GFC 1																	D X S	D X S

	16-févr	17-févr	18-févr	19-févr	20-févr	23-févr	24-févr	25-févr	26-févr	27-févr
Tâches récurrentes :										
- Design review	Z D X F S	Z D X F S								
- Code review						Z D X F S				
Itération 2 :										
- Implémentation Model Checking 2										
- Implémentation Parsing AST 2 (VF)										
- Implémentation conversion AST to GFC 2										
- Validation séparée :										
- jeu de tests MC 2										
- jeu de tests AST/GFC 2										
- Validation du produit mergé										

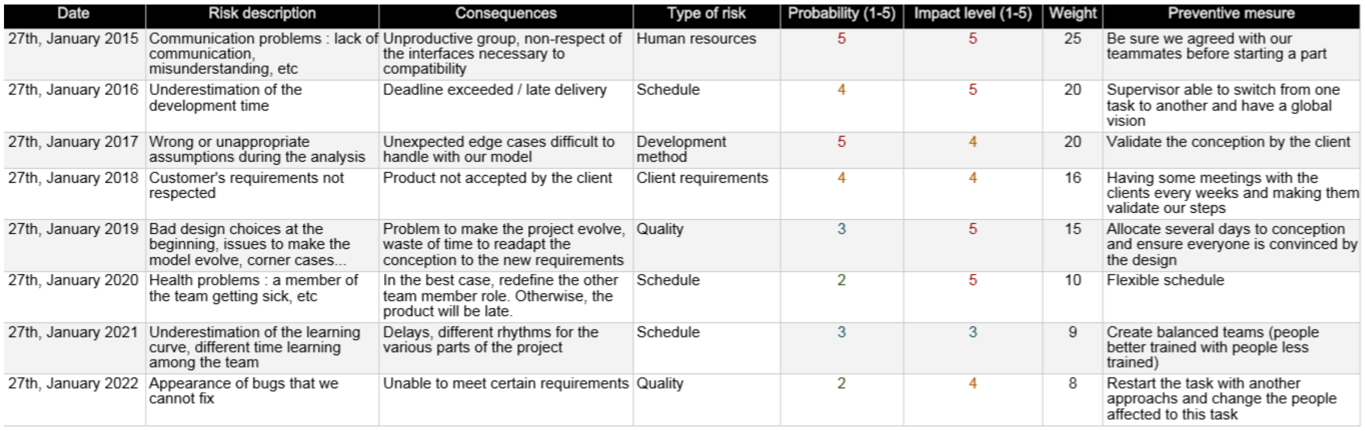
Légende :

Ressource	Rôle
Zohour Abouakil	Chef de projet
David Courtinot	Responsable qualité
Xiaowen Ji	Responsable de la gestion de configuration
Fabien Sauce	Responsable de la documentation
Sofia Boutahar	Responsable des tests

		02-mars	03-mars	04-mars	05-mars	06-mars		09-mars	10-mars	11-mars	12-mars	13-mars
Tâches récurrentes :	▨						▨					🟥
- Design review	▨	Z D X F S	Z D X F S				▨					🟥
- Code review	▨						▨	Z D X F S				🟥
	▨						▨					🟥
Itération 3 :	▨ 🟧	🟧	🟧	🟧	🟧	🟧	▨ 🟧	🟧	🟧	🟧	🟧	🟥
- Implémentation Model Checking 3 (VF)	▨						▨					🟥
- Implémentation conversion AST to GFC 3 (VF)	▨						▨					🟥
- Validation séparée :	▨						▨					🟥
- jeu de tests MC 3	▨						▨					🟥
- jeu de tests AST/GFC 3	▨						▨					🟥
- Validation du produit mergé	▨						▨					🟥

Part III

Risk management



Date	Risk description	Consequences	Type of risk	Probability (1-5)	Impact level (1-5)	Weight	Preventive mesure
27th, January 2015	Communication problems : lack of communication, misunderstanding, etc	Unproductive group, non-respect of the interfaces necessary to compatibility	Human resources	5	5	25	Be sure we agreed with our teammates before starting a part
27th, January 2016	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
27th, January 2017	Wrong or unappropriate assumptions during the analysis	Unexpected edge cases difficult to handle with our model	Development method	5	4	20	Validate the conception by the client
27th, January 2018	Customer's requirements not respected	Product not accepted by the client	Client requirements	4	4	16	Having some meetings with the clients every weeks and making them validate our steps
27th, January 2019	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
27th, January 2020	Health problems : a member of the team getting sick, etc	In the best case, redefine the other team member role. Otherwise, the product will be late.	Schedule	2	5	10	Flexible schedule
27th, January 2021	Underestimation of the learning curve, different time learning among the team	Delays, different rhythms for the various parts of the project	Schedule	3	3	9	Create balanced teams (people better trained with people less trained)
27th, January 2022	Appearance of bugs that we cannot fix	Unable to meet certain requirements	Quality	2	4	8	Restart the task with another approachs and change the people affected to this task

Figure III.1 - Analyse des risques

Part IV

Code management

IV.1 Quality management

IV.1.1 Automated coding style checks

For ensuring that our coding rules are respected and evaluate the quality of our sources, we have used a tool called *Scalastyle* that enables, using an easy-to-use xml configuration file, to check some properties on a Scala code. Combined with a specific pulgin, this can be use 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
MagicNumberChecker	Checks for use of magic numbers instead of constants (safer)	ignore = -1, 0, 1

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

IV.2 Test strategy

IV.3 Configuration management

Part V

Appendices