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

Partie I

Description du projet et objectifs

I.1 Attentes du client

Le client attend une conception d'un prototype permettant la recherche de motifs dans un code C++ afin d'assurer certaines propriétés sur le code vérifiabes par sa syntaxe. Pour cela, différentes analyses seront mises en œuvre pour éviter les dysfonctionnements. Une partie de ces analyses consiste à étudier le code embarqué et à prouver que le code fait bien ce qu'il est censé faire. D'autres analyses consistent à montrer que le code embarqué respecte un certain nombre de règles de programmation. L'analyse consiste donc à rechercher ces motifs dans le code source.

Nous utiliserons pour ce faire la représentation interne du code de Clang afin de construire des graphes que nous pourrons utilise rà l'aide de la logique temporelle. Ce travail se décompose donc en deux parties : le model checking (algorithmes de recherche etc) et les étapes de transformation du code vers sa représentation en graphe de flot de contrôle (GFC). Ces deux parties doivent fonctionner aussi bien de façon indépendante que l'une avec l'autre. La chaîne de transformations et de traitements est présentée dans la figure ci-dessous :

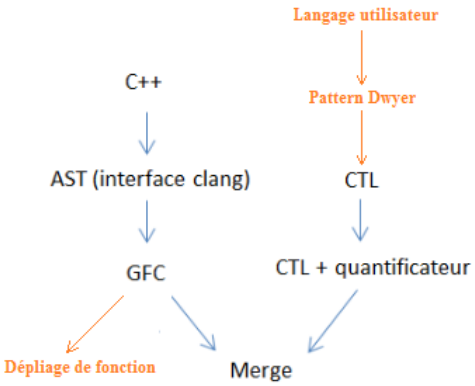


Figure I.1 - Chaîne de transformations et de traitements

I.2 Livrables et priorités

Les livrables attendu (et donc prioritaires) sont les suivants :

- implémentation d'un parser pour l'AST produit par Clang
- une conversion AST vers un modèle Scala de représentation du code en termes de graphe de fulx de contrôle
- de façon indépendante des deux items précédents, des algorithmes d'analyse de propriétés de la logique temporelle sur des graphes de flux de contrôle quelconques
- l'ajout à l'item précédent des quantificateurs tels que "il existe"

Les extensions suivantes pourront être ajoutées :

- dépliage des appels de fonctions sur une profondeur donnée
- création d'un langage utilisateur servant d'interface avec le système

Partie II

Principes d'organisation

II.1 Définition des rôles

Chef de projet

Le chef de projet s'occupe essentiellement des échanges et communications avec l'industriel et les clients. Il a un rôle prépondérant dans l'organisation statique des tâches.

Superviseur

Le superviseur a une vue globale du projet sur le plan technique. Il s'assure de l'avancement des tâches simultanées et peut réorganiser les effectifs et les objectifs de façon dynamique dans le cas d'un imprévu. Le superviseur peut participer à l'écriture du code ou de la documentation mais il ne s'agit pas de sa fonction première. Pour finir, il peut varier d'une semaine sur l'autre.

Responsable qualité

Le responsable qualité définit des règles de bonne programmation et s'assure qu'elles sont vérifiées par les développeurs. Tout code produit passera sous l'oeil attentif du responsable qualité avant d'être validé. Il assure également la qualité et la cohérence de tous les documents produits par l'équipe. Il est le seul à pouvoir pousser du contenu sur le dépôt Github, ou à autoriser un member un pousser.

Responsable de la validation et des tests

Le responsable de la validation est en charge des tests de validation en environnement global écrits par les programmeurs (chaque programmeur a son propre jeu de tests unitaires). Il ne se contente pas de les exécuter, il détermine également si ces tests sont suffisamment exhaustifs ou non.

II.2 Chaîne de développement

II.2.1 Gestion de la qualité

Avant d'être soumis à la validation, tout code produit passe par le responsable qualité. Celui-ci émet, si besoin, des recommandations d'amélioration au programmeur. Le cas échéant, le programmeur doit fournir une nouvelle version tenant compte des remarques ou défendre ses choix s'il ne souhaite pas modifier son code. Ce va-et-vient se poursuit jusqu'à un consensus entre le programmeur et le responsable qualité.

II.2.2 Stratégie de tests

Une fois la qualité du code vérifiée, il est transmis au validateur qui lance les tests, vérifie leur exhaustivité et les complète si besoin. Il n'est pas responsable du débugage du code, tout code échouant aux tests est renvoyé à la personne les ayant développé avec un log détaillant les tests en échec.

II.2.3 Gestion de configuration

Une fois les deux étapes précédentes achevées, le validateur effectue ou autorise le dépôt des sources concernées sur le gestionnaire de version. Le schéma global de cette organisation est donc le suivant :

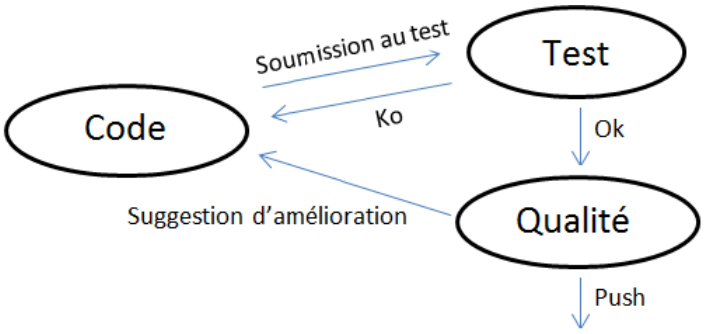


Figure II.1 - Schéma descriptif de la chaîne de développement

Partie III

Planification

III.1 Méthode de développement et de programmation

III.1.1 Scrum

Nous nous efforcerons d'appliquer la méthode Scrum, très utilisée actuellement et reconnue pour son efficacité. En somme, nous définissons tout d'abord un *product backlog*, définissant tous les livrables du produit final. Le présent document fait office de *backlog*. Par la suite, nous diviserons le projet en trois *sprints* (ou *itérations*). Pour chacun d'entre eux, nous définirons le *sprint backlog*, résumant tous les objectifs à atteindre à l'issue de cette itération. Chaque *sprint* s'étendra sur une période de deux semaines et consistera à améliorer le logiciel de façon incrémentale en y intégrant un élément du *product backlog*.

Après chaque sprint, nous tiendrons une réunion pour faire un bilan de l'avancement, et proposer des améliorations ou rectifications de trajectoire (au cours d'une itération, la définition du *sprint backlog* ne peut être modifiée). Enfin, chaque journée débutera par un *scrum meeting* où chacun présentera son travail de la veille et exposera ses objectifs pour la journée ainsi que les éventuelles difficultés qu'il traverse actuellement.

III.1.2 Répartition du travail d'implémentation

Nous utiliserons une approche s'inspirant de la méthode XP. En effet, nous jugeons que l'ampleur du projet ne nécessite pas que chaque membre du groupe programme séparément, et nous estimons que la programmation en binôme est un excellent moyen de prévenir les erreurs en amont et gagner beaucoup de temps en tests et débugage. Nous programmerons donc par binôme, et le cinquième du groupe développera seul. La composition des groupes pourra varier à mesure que les tâches sont remplies.

III.2 Décomposition en tâches

III.3 Planning

Légende :

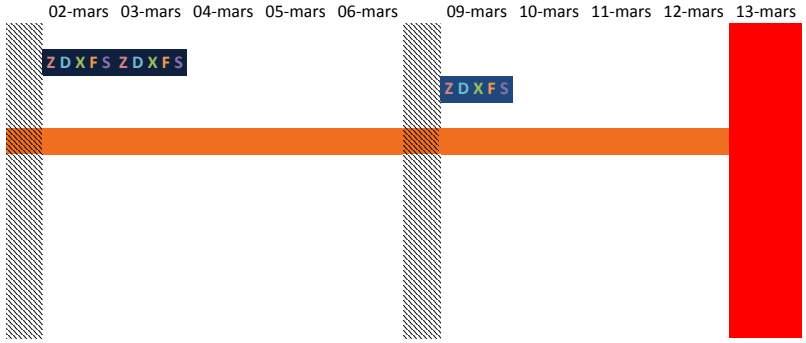
Ressource	Rôle
Zohour Abouakil	Chef de projet
David Courtinot	Responsable qualité
Xiaowen Ji	Responsable de la gestion de configuration
Fabien Sauce	Responsable de la documentation
Sofia Boutahar	Responsable des tests

Tâches récurrentes :

- Design review
- Code review

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



Partie IV

Gestion des risques

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	Scope of project	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.	Scope of project	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	Scope of project	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