Documentation de validation

Equipe 46 :
- Hassou Karim
-Ait Taleb Aymane
-Jaoudar Reda
-Boussemid Youssef
-Bendou Safouane

Janvier 2022

Contents

1 Descriptif des tests :

1.1 Indispensabilité des tests :

Il va sans dire que tester est la meilleure façon de s'assurer du bon fonctionnement d'un produit. Ainsi, faire un nombre important de tests unitaires au fur et à mesure qu'on avançait sur le code était un objectif indispensable, les tests unitaires étaient faits par la personne qui programmait une certaine fonctionnalité afin de s'assurer du bon fonctionnement de son code ; Néanmoins, d'autres tests unitaires se sont rajoutés à chaque fois par d'autres membres de l'équipe afin de tirer des cas plus spéciaux et s'assurer complètement de l'exactitude du programme écrit.

1.2 Étape A : Tests Lexicographiques et Syntaxiques

1.2.1 L'organisation des tests :

Les tests de pour cette étape sont réunis dans le répértoire : Projet-GL/src/test/deca/syntax/ Dans ce répertoire on a classé les tests par catégorie de tests valides : valid/provided/ et tests invalides : invalid/provided/

1.2.2 Les types de tests et leurs Objectifs :

Pour les tests de cette étape on a écrit des tests unitaires pour les différents types valides et invalides afin de tester

si notre compilateur reconnaît bien les différentes suites lexicales, et la génération correcte de l'arbre abstrait. Pour valider cette étape, on s'est basé sur les tests et leurs résultats corrects fournis dans le Polycopie du Projet. On s'est pas trop concentrés sur les tests de cette étape car au fur et à mesure de notre avancement dans les prochaines étapes et surtout l'étape B, on retournera sur les erreurs lexicales et syntaxiques qui vont réapparaître.

1.3 Étape B : Tests Contextuels

1.3.1 L'organisation des tests :

Les tests de pour cette étape sont réunis dans le répertoire : Projet-GL/src/test/deca/context/ Dans ce répertoire on a classé les tests par catégorie de tests valides : valid/provided/ et tests invalides : invalid/provided/

1.3.2 Les types de tests et leurs Objectifs :

Pour les test invalides, on a rédigé des tests unitaires qui se limiteront sur le test et le bon fonctionnement des différentes instructions et expressions basiques. pour chaque test on vérifie son fonctionnement en se basant sur le message d'erreur retourné et sa capacité de bien expliquer l'erreur afin d'être corrigée. Par contre, vu l'inutilité des tests unitaires pour les tests valides, on a plutôt décidés d'écrire des programmes longs et performants afin de valider les différents aspects de notre

compilateur. On valide ces tests par la génération de l'arbre abstrait décoré ,vérifiant le cahier de charge.

1.4 Étape C : Tests de Génération de code

1.4.1 L'organisation des tests :

Les tests de pour cette étape sont réunis dans le répertoire : Projet-GL/src/test/deca/codegen/ Dans ce répertoire on a classé les tests par catégorie de tests valides : valid/provided/ et perf/provided/ et tests invalides : invalid/provided/

1.4.2 Les types de tests et leurs Objectifs:

En ce qui concerne les tests invalides, on a rédigé des tests unitaires afin de vérifier les différents messages d'erreurs;citant en guise d'exemple, la division par zéro. De l'autre coté pour les tests valides, on les a classés en des tests unitaires dans le répertoire /valid, et des tests fonctionnels dans le répertoire /perf, on valide le bon fonctionnement de notre tests en vérifiant la cohérence avec les résultats attendus.

2 Les scripts de tests :

Faire des scripts de tests nous a été très utile, car ceci nous a permis de parcourir et exécuter plusieurs tests à la fois ; On avait économisé par conséquent une durée de temps considérable, qui était exploitée à écrire plus de tests et enrichir notre base de tests. les scripts sont tous écrit dans le répertoire : Projet-GL/src/test/script/launchers

2.1 Script Test context :

Dans le répertoire cité précédement on a ajouté le fichier "testing-context.sh" qui exécute tout les tests valides et invalides, et retourne un message validant le bon déroulement du test en message : "Succès/Echec"+"Attendu"+"pour le test" sinon le message s'affichant est "Succès/Echec"+"Inattendu"+"pour le test"; pour ce faire, on 'a utilisé des scripts écrits en Python se trouvant dans la racine du projet et qui est nommé "test-script.py".

2.2 Script Test codegen :

De même que précèdement, on a ajouté "testing-codegen.sh" qui exécute tout les tests valides et invalides, ainsi que leurs décompilation , et valide le bon déroulement du test an affichant le même message cité dans la partie précédente.

3 Résultat de Jacoco :

Jacoco est un outil qui nous a été très utile, il nous a permis de mesurer la couverture du code par la base des tests, et nous a fourni par conséquent une idée sur la qualité des tests qu'on avait rédigés . L'image ci-dessous représente le résultat de Jacoco :

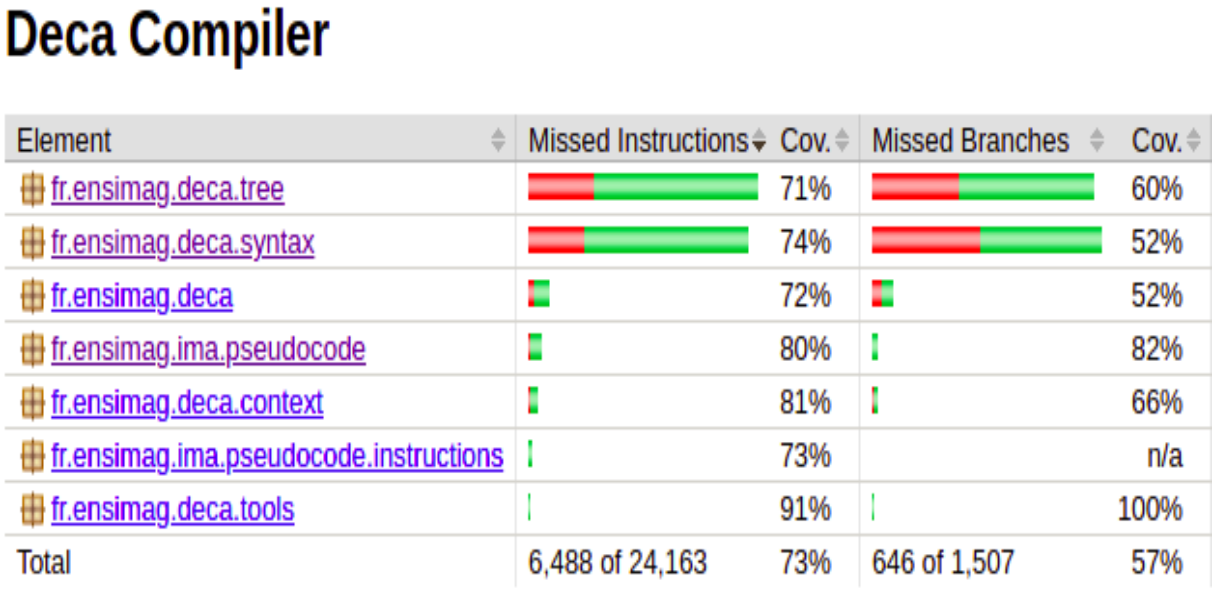

Figure 1: Résultat de Jacoco

4 Gestion des risques :

4.1 Risques liés à l'environnement de travail :

Afin de mener à bien la réalisation du projet, il est primordial d'avoir tout un environnement de travail installé au préalable dans la machine sur laquelle on travaille. Bien évidemment, les machines de l'Ensimag sont suffisamment équipées pour travailler en toute sérénité. De plus, il n'est pas nécessaire d'être sur place pour pouvoir utiliser les machines de l'Ensimag, il est toujours possible d'effectuer une connexion SSH afin de travailler à distance sur une de ces dernières. Néanmoins, on est jamais à l'abri d'une panne système qui nous empêchera d'utiliser les ordinateurs de l'école. De surcroît, tra-

vailler à distance sur une machine de l'école réduit considérablement la productivité des membres. Ainsi, pour gérer ce risque, on a procédé à l'installation de l'environnement de travail nécessaire au bon déroulement du projet dans chacune des machines des différents membres de notre équipe. Cela nous a permis à terme à travailler aisément et ce même en dehors des horaires de l'ouverture de l'école.

4.2 Maladie et/ou absence de longue durée d'un des membres de l'équipe :

Pour éviter l'occurrence de tels problèmes qui auront des conséquences désastreux sur l'avancement du projet, on notifiera les profs dès le début de ce genre d'imprévu et on organisera évidemment une réunion de groupe pour partager les taches du membre absent. Plus particulièrement, Si un membre de l'équipe attrape le Covid19, les autres équipiers non infectés continueront à se rassembler et travailler ensemble en gardant toujours contact (appel discord ou par zoom) avec notre collègue malade . Si jamais, le membre de l'équipe en question se juge incapable de travailler avec la même cadence, or organisera évidemment une réunion d'équipe pour alléger ses responsabilités et diviser les taches entre les autres .

4.3 Mauvaise répartition des taches :

C'est un danger considérable car une mauvaise répartition des taches causera des retards importants et bien évidemment le non respect des délais sur notre planning. Pour éliminer

les menaces liées à ce danger, on organise souvent des réunions d'équipe pour rééquilibrer les taches.

4.4 Manque d'honnêteté professionnelle : plagiat d'un des membres d'équipe.

Avoir une honnêteté professionnelle est une condition indispensable pour bien travailler en équipe, si un des membres de l'équipe remarque le plagiat d'un de ses collègues, il doit notifier immédiatement les professeurs, et si jamais, cet acte de tricherie passerait indétecté par les autres membres de l'équipe, la claire répartition des taches indiquera aux profs l'étudiant responsable du plagiat.

4.5 problèmes de communication au sein du groupe :

C'est un danger important, car si les mauvaises ententes au sein de l'équipe persistent, l'ambiance se détériora et les membres d'équipe perderont la motivation de travail, et ceci affectera par conséquent le rendement global du groupe.

4.6 Risques liés à l'interdépendance entre les taches :

Lors de l'élaboration du planning prévisionnel au début du projet, on a tout de suite pris conscience de l'existence d'interdépendances entre les différentes tâches du pro-

jet, les unes devant impérativement être achevées avant que les autres puissent être entamées. De ce fait, on a repéré les différentes sections critiques du projet et on a fait moultes efforts pour veiller à ce que ces sections soient finies à temps. Néanmoins, il est tout à fait possible qu'un ou plusieurs membres de l'équipes bloquent au niveau d'une de ces sections. Le cas échéant, pour remédier à ce problème, on a décidé d'augmenter l'effectif du groupe qui travaille sur la tâche qui cause problème. Néanmoins, il faut faire attention à ce que ces retards ne se reproduisent pas beaucoup car le cumul de ces retards pourrait à terme entraver la réalisation du cahier de charge complet à temps et on se devra de réévaluer nos objectifs.

4.7 Risques liés à la malcompréhension des consignes du projet:

Il est également possible de mal interpréter une des consignes du sujet. Ceci s'avère être très pénalisant que ce soit en terme de la qualité du rendu ou en terme de temps (si on se rend compte de l'erreur). Ainsi, on essaie de prendre notre temps pour comprendre les consignes du sujet et si éventuellement des zones de flou subsistent, on n'hésite pas à prendre contact avec notre professeur enseignant.

4.8 Risques liés à une mauvaise révision du code :

On fait également attention à ce que notre projet passe les tests basiques à chaque fin de journée. Ces tests étant triviaux, ne pas les passer serait très alarmant et sera indicateur d'une très mauvaise implémentation dans une partie de notre projet. Il est par suite très important de repérer ces anomalies au plus vite et ce par le biais de tests réguliers.

4.9 Risques liés à la mauvaise utilisation de git :

Il est très important d'utiliser git à bon escient car une mauvaise utilisation de cet outil pourrait pénaliser toute l'équipe. Par exemple, un push peu réfléchi peut provoquer une perte de données. Pour éviter ça, on garde toujours des copies du projet dans une des machines personnelles des membres de l'équipe. Une autre dérive d'une mauvaise utilisation de git est le fait que des parties du projet manquent. Ceci peut être dû au fait que les membres responsables de l'implémentation de la partie en question n'aient pas effectués leur push correctement. Par conséquent, on a pris le temps au début du projet d'apprendre les règles de base afin de bien utiliser git.

5 Gestion des rendus :

Les points forts de notre équipe étant la compréhension et la maitrise quasi-totale des différents aspects du projet par tous les membres du groupe, tout le monde participera à la documentation des différentes parties du projet, ceci en parallèle avec l'avancement du développement et des tests unitaires. La documentation est constamment mise à jour et modifiée par les différents membres de l'équipe si jugé nécessaire. Avant chaque rendu (rendu intermèdiaire, rendu final), une réunion d'équipe est organisée, lors de cette réunion, on exécute les tests fonctionnels (Les tests unitaires sont éxécutés au fur et à mesure de l'écriture du code, ces tests sont censés etre tous valides) , chacun des membres d'équipe donne ses suggestions, ses idées pour améliorer encore plus la qualité de notre compilateur et enrichir notre base de tests avant le rendu .