Projet Génie Logiciel

Gestion des Risques et Gestion des Rendus

Réalisé par :
Mouez JAAFOURA, Ilyas MIDOU, Meriem KAZDAGHLI,
Salim KACIMI, Mehdi DIMASSI

Date : 09 janvier 2025

Grenoble INP - Ensimag & Phelma
Université Grenoble Alpes

Table des matières

1 Introduction

Ce projet, réalisé dans le cadre du programme de Génie Logiciel à l'Université INP-Grenoble, porte sur le développement d'un compilateur deca dont les différentes phases incluent la conception, le développement, les tests et la livraison des fonctionnalités. L'objectif principal de ce document est de fournir un aperçu des approches adoptées pour minimiser les risques et assurer la réussite des livrables, tout en soulignant les outils et méthodes de validation utilisés pour garantir la conformité du projet aux exigences.

2 Gestion des risques

2.1 Pourquoi la Gestion des risques ?

Dans un projet logiciel, la gestion des risques est primordiale car elle nous aide à anticiper les problèmes qui pourraient survenir à chaque étape du développement. Que ce soit des erreurs techniques, comme des bugs ou des incompatibilités entre outils, ou des difficultés organisationnelles, comme des retards ou des problèmes de communication, ces risques peuvent sérieusement affecter la progression du projet.

Pour éviter cela, nous mettons en place des outils comme des tests réguliers, des revues de code et une gestion de projet claire avec clickup. Cela nous permet de réagir rapidement et de minimiser l'impact de ces risques. En anticipant les problèmes dès le départ, nous assurons une meilleure qualité du produit final et respectons nos délais.

2.2 Date de rendu

Notre projet impose des délais stricts pour la remise des livrables, ce qui nécessite une organisation rigoureuse. Pour garantir que toutes les tâches soient terminées à temps, nous avons élaboré un planning détaillé qui nous indique les différentes étapes à accomplir, de l'implémentation des fonctionnalités aux tests de validation.

Afin d'assurer le respect des délais, nous utilisons ClickUp pour gérer notre calendrier. Des rappels automatiques sont configurés pour nous alerter lorsqu'une tâche doit être terminée avant une date limite. Par exemple, une notification est envoyée à chaque membre de l'équipe avant chaque phase importante, comme un rendu ou une révision. Ces alertes permettent à l'équipe de s'assurer que toutes les étapes sont complétées et que les livrables sont envoyés à temps.

2.3 Erreurs triviales

Pour éviter les erreurs triviales, telles que des erreurs de syntaxe ou des oublis dans le code, nous utilisons des IDE intelligents comme *IntelliJ* et *VSCode*, qui offrent des fonctionnalités telles que l'autocomplétion, la détection de fautes de syntaxe en temps réel, et des suggestions d'amélioration du code. Ces outils nous permettent de gagner du temps et de réduire les risques d'erreurs simples qui pourraient ralentir le projet.

Pour les erreurs plus globales, notamment celles liées à l'interprétation du cahier des charges ou à des décisions de conception, nous privilégions la communication au sein de l'équipe. Si un membre de l'équipe a des doutes sur une partie du cahier des charges ou sur la manière de résoudre un problème, nous organisons des discussions pour clarifier les points flous. Cela nous permet de nous assurer que tout le monde est sur la même longueur d'onde et que les choix réalisés sont cohérents avec les objectifs du projet. Ces échanges réguliers contribuent à minimiser les risques de malentendus et à garantir que chaque membre de l'équipe travaille dans la même direction.

2.4 Tests

Pour garantir la qualité du code et assurer la fiabilité du projet, chaque fonctionnalité est prise en charge par deux personnes : l'une s'occupe de l'implémentation de la fonctionnalité, tandis que l'autre est responsable de l'écriture des tests associés. Cette division des tâches permet de se concentrer sur la mise en œuvre du code tout en garantissant qu'il soit systématiquement testé dès sa création.

Tous les jours, les deux membres concernés échangent régulièrement pour s'assurer que les tests sont rédigés en parallèle avec le développement de la fonctionnalité. L'objectif est que, dès qu'une partie du code est implémentée, la personne en charge des tests puisse immédiatement rédiger les scénarios de tests correspondants. Cette approche permet d'éviter que des fonctionnalités ne soient oubliées lors des tests et garantit que chaque partie du projet est couverte de manière exhaustive. Ces échanges quotidiens assurent également que le code est testé dans son contexte et répond aux exigences du cahier des charges.

Ainsi, cette méthode garantit une validation continue du travail effectué, tout en permettant de repérer rapidement des problèmes ou des incohérences avant qu'ils ne deviennent plus difficiles à corriger.

Plusieurs scripts shell ont été développés pour effectuer des tests et évaluer différentes fonctionnalités du compilateur Deca, tels que *valid-context.sh* et *valid-synt.sh*, qui exécutent les *tests fournis* (*test-synt* et *test-context*) sur un répertoire de tests afin de vérifier que tous les tests sont valides ou non. Nous avons également intégré ces scripts dans la commande *mvn test*.

2.5 Méthodes générales

Test et code : Pour éviter les erreurs liées à des biais de compréhension, chaque membre de l'équipe a un rôle clairement défini. Les développeurs et les testeurs ne sont pas en contact direct lors de l'implémentation du code, mais un bilan est effectué à la fin de chaque tâche journalière. Cette méthode permet aux testeurs de ne pas avoir de préjugés sur le code source. Ainsi, lorsqu'ils évaluent chaque fonctionnalité, ils le font de manière objective, sans supposer à l'avance ses faiblesses ou ses points forts. Cela garantit que les tests sont réalisés dans un cadre impartial, assurant ainsi que tous les aspects du code sont correctement validés, sans biais de connaissance ou d'interprétation.

documentation : En ce qui concerne la documentation, elle est rédigée tout au long du projet, et non à la fin. Cela nous permet de garder une trace continue de notre travail et de nous assurer que chaque membre de l'équipe est informé des avancées du projet à chaque étape. Un diagramme UML a été créé et mis à jour chaque fois qu'une nouvelle classe est implémentée.

gestion de temps : Pour les risques liés à la gestion du temps et à la coordination de l'équipe, des outils comme Trello et ClickUp sont utilisés pour planifier les tâches, suivre leur avancement, et alerter chaque membre des échéances importantes.

3 Gestion des rendus

La gestion des rendus permet de planifier et de contrôler les déploiements pour garantir une livraison conforme aux attentes du client. Voici la liste des actions à effectuer avant chaque rendu :

1. Développement et correction de bugs.
2. Réunion de validation des fonctionnalités.
3. Validation par les testeurs.
4. Exécution des tests avec `mvn test`.
5. Merge des branches sur `master`.
6. Vérification sur une machine indépendante.

7. effacer les fichier executable avec `mvn clean`.
8. pusher puis verifier sur git.
9. Livraison de la version.

4 Conclusion

Ce document résume les pratiques mises en place pour la gestion des risques et des rendus dans le cadre du projet Génie Logiciel. Ces démarches garantissent la qualité et la fiabilité des livrables.