Gestion des risques et Gestion des rendus

Groupe 1, GL3

—

Membres

Nils Depuille, *Chef de Projet*
Jorge Luri Vañó, *Git Manager & Human Ressources Director*
Vianney Vouters, *Secrétaire Général*
Virgile Henry, *Code Manager*
Logan Willem, *Test Manager*

PARTIE I:

[Introduction]

Risque: Oubli des dates de rendu.

Action(s): A chaque daily meeting, on regarde les prochains suivis et évènements de groupe.

[ExempleSansObjet]

Risque: Ne pas avoir le même résultat que l'exemple fourni.

Action(s): Créer un test reprenant exactement les mêmes instructions et l'exécuter une fois les parties A et B implémentées pour la partie sans objet.

[RenduIntermediaire]

Risque: La banque de tests est insuffisante

Action(s): Avoir un document recueillant l'entièreté des tests réalisés et relevant quelles règles n'ont pas encore été testées.

Risque: Baisse de qualité contre l'avancement du projet.

Action(s): Avant de passer à l'étape suivante (exemple: de sans objet à objet), s'assurer que la qualité de la phase terminée est satisfaisante et que la banque de tests est complète.

[A-Rendre]

Risque: Oublier un fichier du projet

Action(s): S'assurer que la branche develop a bien été push sur master

Risque: Lors de la mise à jour de la branche master, une erreur apparait dans master

Action(s): Garder la dernière journée pour réaliser tous les tests possibles afin de nous assurer que le code de master fonctionne comme il devrait.

[Soutenance]

Risque: Oublier certaines spécifications ou aspect du projet

Action(s): Réaliser un bilan après chaque sprint en notant les points majeurs réalisés pour pouvoir synthétiser le livrable.

Risque: Retard le jour de la soutenance

Action(s): S'assurer de l'heure de la soutenance, mettre différents réveils et nous donner rendez-vous deux heures avant pour prévoir les retards et éventuellement pratiquer la soutenance à l'avance.

Risque: Inégalité des temps de parole.

Action(s): Préparer la présentation à l'avance en faisant attention à bien distribuer les temps de parole et pratiquer plusieurs fois la soutenance.

Risque: La machine utilisée pour réaliser les démonstrations présente un problème durant la soutenance.

Action(s): Avant la soutenance, s'assurer que démonstrations prévues sont possibles et correctes dans au moins deux machines. Le jour-J, apporter les deux machines chargées afin d'avoir une machine principale et une machine de secours.

[Communication]

Risque: Ne pas communiquer sur l'avancement de chacun. Ou bien sur la répartition du travail.

Action(s): Daily meeting tous les matins pour savoir comment avance le projet et comment se répartir le travail de la journée.

Risque: Ne pas comprendre les règles de syntaxe de la même manière entre 2 étapes

Action(s): Tester et analyser le comportement des programmes à l'exécution.

Risque: Il y a un conflit au sein du groupe.

Action(s): Essayer de régler le problème diplomatiquement, et demander à quelqu'un de médier si le problème est toujours présent (c.f. Charte d'équipe).

Risque: Ne pas avoir bien compris ce qu'un autre membre a demandé de faire.

Action(s): Demander des précisions et des explications tant que le récepteur de l'ordre a toujours des doutes de ce qu'il faut faire.

[Suivis]

Risque: Négliger le suivi et perdre des points alors qu'on fait le projet GL du siècle.

Action(s): Préparer à l'avance nos diapos et les sujets évoqués pendant le suivi.

Risque: Ne pas adopter les conseils donnés par les professeurs.

Action(s): Noter les conseils proposés et les évoquer au prochain Daily Meeting afin de changer notre fonctionnement de travail.

Risque: La machine utilisée pour réaliser les démonstrations présente un problème durant le suivi.

Action(s): C.f. Soutenance.

[Consignes]

Risque: Mauvaise compréhension des consignes.

Action(s): Demander aux professeurs.

Risque: Doutes sur une consigne.

Action(s): Demander aux professeurs.

PARTIE II:

[Lexicographie]

Risque: Mauvaise transcription des règles lexicales.

Action(s): Le membre relira son code plusieurs fois avec le sujet à côté, puis il demandera de relire à un membre qui travaille avec lui dans la partie A et, finalement, le membre demandera à un autre membre travaillant dans une autre partie de vérifier le code écrit.

Risque: Oubli d'une règle lexicale.

Action(s): C.f. risque précédent.

Risque: Erreur d'écriture de code.

Action(s): C.f. risque précédent

Risque: Les TOKENS ne sont pas créés comme on le souhaiterait.

Action(s): Réalisation de tests vérifiant que tous les différents tokens possibles sont bien implémentés. Les membres essairont de créer des tests piegeux (exemple : "nul l" afin de vérifier si le lexer créer un token NULL ou bien deux tokens variables "nul" et "l").

[Syntaxe]

Risque: Mauvaise transcription des règles syntaxiques.

Action(s): Le membre relira son code plusieurs fois avec le sujet à côté, puis il demandera de relire à un membre qui travaille avec lui dans la partie A et, finalement, le membre demandera à un autre membre travaillant dans une autre partie de vérifier le code écrit.

Risque: Oubli d'une règle lexicale.

Action(s): C.f. risque précédent.

Risque: Erreur d'écriture de code.

Action(s): C.f. risque précédent

Risque: Le code ne fait pas ce qui était attendu.

Action(s): Réalisation de tests couvrant toutes les règles syntaxiques afin de vérifier que chaque règle réalise ce qu'on souhaite avoir.

[Decompilation]

Risque: Mauvaise transcription des règles de décompilation.

Action(s): Le membre réalisera des tests pour vérifier que le code résultant de la décompilation est le même que le code originel fourni en test. Les tests chercheront à couvrir l'entièreté des classes de l'arbre.

Risque: Erreur de compréhension

Action(s): Suivre le sujet et, si une loi n'est pas bien comprise, demander aux autres membres.

Risque: Le code ne réalise pas ce qui a été attendu et on ne trouve pas l'erreur.

Action(s): Réaliser des tests pour trouver l'erreur. Si l'erreur n'est pas repérée, demander aux autres camarades de réviser l'erreur et proposer des tests à faire. Si l'erreur n'est pas encore repérée, demander à un professeur.

[SyntaxeContextuelle]

Risque: Oublier d'implémenter certaines règles, ou oublier d'implémenter toutes les conditions d'une règle.

Action(s): Création d'un Google Sheet, répertoriant toutes les règles et toutes les conditions et en notant à coté s'il elles ont été réalisées ou non. Cela nous permet, pour chaque règle de réellement nous concentrer pour étudier tout les cas possible avant de passer à la règle suivante. De plus, une grande diversité des tests essayant de couvrir tous les cas permet de trouver certaines erreurs (décors non implémentés, règles oubliées, conditions non vérifiées, ...)

[Sémantique]

Risque: Erreur de compréhension

Action(s): Lire le sujet plusieurs fois et, si une loi n'est pas bien comprise, demander aux autres membres. Noter aussi toutes les notions à prendre en compte dans des fiches pour nous assurer de rien oublier.

[Decac]

Risque: Oublier d'implémenter les options du compilateur.

Action(s): Assignation de la tâche dès le début à un membre du groupe qui est en tout temps conscient du travail à fournir.

Risque: Non respect des attendus de chacune des options.

Action(s): Vérification de la bonne exécution de chaque option par un autre membre du groupe.

Risque: Ne pas respecter la syntaxe des messages d'erreur imposée permettant l'évaluation automatique.

Action(s): Vérification par un autre membre du groupe que les messages sont correctement écris, en accord avec la partie résultat des tests invalides pour la partie contextuelle par exemple.

[MachineAbstraite]

Risque: mauvaise organisation de la génération du code, qui mène à des complications et erreurs du code généré.

Action(s): S'imposer des conventions de génération du code, et penser des méthodes pour simplifier la génération.

[ConventionLiaison]

Risque: Ne pas respecter les conventions de liaisons.

Action(s): Relire plusieurs fois la doc : écrire clairement toute les conventions, et les relires après chaque implémentation en s'assurant qu'elles sont respectées.

Risques sur Git

Membre ne peut pas pull à cause d'un conflit dans les commits

- Le membre réalisera une résolution de conflits à la main, en vérifiant quelles lignes de quels fichiers garder et lesquels modifier.

Membre code sur une branche qui n'est pas la sienne et s'aperçoit trop tard

- Le membre devra push sur la brance où il désirait de pusher et, après avoir pushé, il devra refuser les modifications réalisées dans la branche où il se trouve : la branche où il se trouve n'aura pas de modifications mais les commits seront bien présents dans ls branche désirée.
- Si git ne laisse pas push dans l'autre branche à cause d'un merge conflit, le membre devra manuellement copier et coller les codes dans l'autre branche. Une option pourra être de copier les fichiers en dehors du dossier du projet, refuser les modifications, changer à la branche désirée et coller les fichiers dans la nouvelle branche.

Membre a pushé un commit qui ne devait pas push

- Si l'erreur est petite, le membre résoudra l'erreur en priorité et pushera la solution pour ne pas affecter le travail d'autres camarades.
- Si l'erreur n'est pas rapide à résoudre, le membre révertira le commit (pushera le commit précédant au commit problématique) et résoudra le problème en local.