Projet Génie Logiciel

Charte

Aurélien VILMINOT
Damien CLAUZON
Guilherme KLEIN
Léon ROUSSEL
Pierre ARVY

10 janvier 2022

Résumé

Cette charte d'équipe définit les règles de fonctionnement que nous mettons en place collectivement pour rendre le plus efficace possible le travail de l'équipe projet. Ces règles précisent notamment l'organisation du travail au sein de l'équipe, les méthodes de travail utilisées au sein de celle-ci, ainsi que ses valeurs communes.

Table des matières

1 Équipe **2**
1.1 Grilles d'évaluations . . . 2
1.2 Matrice SWOT . . . 4
1.3 Valeurs communes . . . 4
1.4 Rôles et responsabilités . . . 5
1.4.1 Rôles permanents . . . 5
1.4.2 Rôles tournants . . . 5

2 Organisation **6**
2.1 Présence, retards et absences . . . 6
2.2 Réunions . . . 6

3 Méthodes de travail **6**
3.1 Travail à fournir . . . 6
3.2 Développement . . . 7
3.3 Outils . . . 7
3.3.1 Canaux de communication . . . 7
3.3.2 Versionnage du code . . . 7
3.3.3 Gestion de projet . . . 7
3.3.4 Rédaction de documents . . . 8
3.3.5 Environnement de développement . . . 8

1 Équipe

1.1 Grilles d'évaluations

Prénom NOM : Pierre ARVY
Formation d'origine, filière : Licence Mathématiques fondamentales, MMIS
Équipe : 1
Tuteurs :

Caractéristiques personnelles	**Évaluation**				
Leadership	1	2	3	④	5
Planification	1	2	3	④	5
Esprit d'équipe	1	2	③	4	5
Organisation	1	2	3	④	5
Persévérance	1	2	3	4	⑤
Ponctualité	1	2	③	4	5
Créativité	1	2	3	④	5
Débrouillardise	1	2	③	4	5
Orateur	1	2	3	4	⑤
Communication écrite	1	2	3	4	⑤
Compétences techniques					
Deca	1	②	3	4	5
Java	1	2	3	④	5
Assembleur	1	2	③	4	5
Git	1	2	③	4	5
Algorithmique	1	2	3	④	5

Prénom NOM : Aurélien VILMINOT
Formation d'origine, filière : IUT Informatique, ISI
Équipe : 1
Tuteurs :

Caractéristiques personnelles	**Évaluation**				
Leadership	1	2	③	4	5
Planification	1	2	③	4	5
Esprit d'équipe	1	2	3	④	5
Organisation	1	2	3	④	5
Persévérance	1	2	3	④	5
Ponctualité	1	2	3	④	5
Créativité	1	②	3	4	5
Débrouillardise	1	2	3	④	5
Orateur	1	2	3	4	⑤
Communication écrite	1	2	3	4	⑤
Compétences techniques					
Deca	1	②	3	4	5
Java	1	2	3	4	⑤
Assembleur	1	2	3	④	5
Git	1	2	③	4	5
Algorithmique	1	2	3	④	5

Prénom NOM : Léon ROUSSEL
Formation d'origine, filière : CPGE MPSI/MP, MMIS
Équipe : 1
Tuteurs :

Caractéristiques personnelles	**Évaluation**				
Leadership	1	2	③	4	5
Planification	1	2	3	④	5
Esprit d'équipe	1	2	3	④	5
Organisation	1	2	③	4	5
Persévérance	1	2	3	④	5
Ponctualité	1	2	3	4	⑤
Créativité	1	2	③	4	5
Débrouillardise	1	2	3	④	5
Orateur	1	2	③	4	5
Communication écrite	1	2	③	4	5
Compétences techniques					
Deca	1	②	3	4	5
Java	1	2	3	④	5
Assembleur	1	②	3	4	5
Git	1	2	③	4	5
Algorithmique	1	2	3	④	5

Prénom NOM : Damien CLAUZON
Formation d'origine, filière : CPGE MPSI/MP, ISI
Équipe : 1
Tuteurs :

Caractéristiques personnelles	**Évaluation**				
Leadership	1	2	③	4	5
Planification	1	2	③	4	5
Esprit d'équipe	1	2	3	④	5
Organisation	1	2	3	④	5
Persévérance	1	2	3	④	5
Ponctualité	1	2	3	4	⑤
Créativité	1	2	3	④	5
Débrouillardise	1	2	3	④	5
Orateur	1	2	③	4	5
Communication écrite	1	②	3	4	5
Compétences techniques					
Deca	1	②	3	4	5
Java	1	2	3	4	⑤
Assembleur	1	2	3	④	5
Git	1	2	3	4	⑤
Algorithmique	1	2	3	④	5

Prénom NOM : Guilherme KLEIN
Formation d'origine, filière : Ciência da Computação - UFRGS, ISI
Équipe : 1
Tuteurs :

Caractéristiques personnelles	**Évaluation**				
Leadership	1	2	3	④	5
Planification	1	2	3	4	⑤
Esprit d'équipe	1	2	③	4	5
Organisation	1	2	3	④	5
Persévérance	1	2	3	④	5
Ponctualité	1	2	③	4	5
Créativité	1	2	3	④	5
Débrouillardise	1	2	3	4	5
Orateur	1	②	3	4	5
Communication écrite	1	2	③	4	5
Compétences techniques					
Deca	①	2	3	4	5
Java	1	2	3	④	5
Assembleur	1	②	3	4	5
Git	1	2	3	④	5
Algorithmique	1	2	3	④	5

1.2 Matrice SWOT

STRENGTHS

1. Nous nous connaissons tous, la cohésion de l'équipe est donc déjà présente ce qui nous permet d'être dynamique dès le commencement du projet
2. Capacité à réaliser rapidement une tâche demandée tout en s'adaptant aux contraintes fixées

WEAKNESSES

1. Pas d'expérience dans le développement dirigé par les tests
2. Connaissances liées à l'assembleur restreintes

OPPORTUNITIES

1. Les compétences de chacun sont complémentaires
2. Un membre de notre équipe a réalisé un IUT Informatique

THREATS

1. Possibilité de s'enraciner dans un problème trop longtemps
2. Les outils ne sont pas maîtrisés par tous (GitLab, Trello ...)

1.3 Valeurs communes

Le règlement lié aux retards et aux réunions est décrit plus bas. La présence doit être respectée afin de construire une équipe soudée.

Les outils utilisés doivent être ceux décrits dans la partie concernée en page 7. Les différentes parties réalisées ou qui sont encore à faire sont indiquées dans le tableau de bord. Les tâches, qui découpent les parties précédentes, à faire ou qui sont déjà faites doivent être indiquées sur Trello.

La fraude est proscrite. Pour plus d'information, se référer à la partie **[Fraude]** du polycopié du projet. Le "copié-collé" est une pratique à bannir. Il est demandé d'écrire chaque partie de code. Une quelconque interaction avec un projet réalisé pendant les années antérieures est proscrite.

Les anglicismes doivent être limités au maximum. Les documentations doivent être réalisées dans le langage suivant : *français*. Les abréviations doivent être définies avant d'être utilisées.

Il est impératif de consulter quotidiennement cette page qui présente les actualités du projet (modification d'énoncé, informations supplémentaires, ...).

Chacun s'engage à tenir son rôle fixe, tournant et les différentes responsabilités qui en découlent.

En cas de conflit, une réunion générale de l'équipe est organisée afin de comprendre le problème. Chaque membre de l'équipe peut alors apporter son avis. Dans le cas où nous ne trouvons pas de consensus, un vote sera réalisé pour trancher. [1] Le problème doit être résolu le plus rapidement possible afin de ne pas perdre de temps.

La répartition du temps de parole, pendant une réunion, doit être équilibrée. Tous les membres peuvent donner leurs avis et participer au débat. Les idées de chacun doivent être considérées et respectées.

Chaque membre du groupe a le droit à l'erreur. Il est donc possible de régulariser une situation sans se voir attribuer une sanction. L'objectif principal est avant tout d'apprendre de ses erreurs et de progresser.

1.4 Rôles et responsabilités

En plus de certains rôles fixes, nous allons utiliser la méthode agile SCRUM pour développer le projet. N'ayant pas assez de matière pour certains rôles, nous avons fait le choix d'en regrouper certains.

1.4.1 Rôles permanents

Responsable Trello : Aurélien
- Gestion des tâches et des cartes du tableau de bord
- Identification des anomalies

Responsable Git : Damien
- Définir les règles pour les commits
- Régler les conflits inter-git
- S'assurer que les commits sont propres, que les revues de code sont effectuées

Responsable Drive : Aurélien
- Gérer l'organisation du Drive
- Sauvegarde des fichiers externes au code (tableau de bord...)

Responsable Documentation : Pierre
- Définir les besoins précis en terme de documentation
- S'assurer que la documentation est bien remplie en parallèle de l'avancement du projet

1.4.2 Rôles tournants

Scrum Master (animateur/vision générale du projet)
- Organiser les réunions hebdomadaires et quotidiennes
- Écrire les compte-rendus de réunion
- S'assurer que le tableau de bord et le Trello sont remplis

Product Owner / Architecte / Designer
- Écrire les user stories
- Définir les besoins utilisateurs et les besoins pour l'extension
- Gestion et planification des livrables
- Définir l'architecture du projet

Développeur

1. Le déroulement du vote est défini par le Scrum Master.

- Responsable du plan du sprint : évaluer ce qui est possible de faire pendant un sprint, revenir sur des décisions après-coup, compte-rendu des tâches effectuées
- Responsable de la qualité du code : participer à des revues de code en tant que candide, faire relire son propre code par les autres

Testeur
- Participer à la réalisation et à la vérification de tous les tests
- S'assurer de la bonne couverture des tests en utilisant les outils Maven et Jacoco
- Avoir une vue d'ensemble sur les tests à compléter et/ou réaliser

2 Organisation

2.1 Présence, retards et absences

Du lundi au vendredi, présence obligatoire à l'Ensimag de 9h à 11h30 et de 14h à 16h30 (sauf si jour de repos). Chacun s'organise pour totaliser au minimum 35h de travail par semaine (dont 25h à l'Ensimag sans jour de repos).

Arriver après 9h05 ou 14h05 est considéré comme un retard. À chaque retard, le retardataire doit faire 3 fois 1 minute de gainage. S'il cumule plusieurs retards dans la même semaine, des sanctions s'ajoutent :
- 2ème retard : il doit apporter des croissants pour le reste de l'équipe d'ici la fin de la semaine (ou semaine prochaine si vendredi).
- 3ème retard : il doit venir travailler 3h (minimum) le samedi matin.

Sur l'ensemble du projet, 4 demi-journées d'absence sont accordées (sauf maladie). Si possible, elles doivent être posées à l'avance. La pratique du sport universitaire peut imposer de réserver ces demi-journées.

2.2 Réunions

Toutes les réunions hebdomadaires et journalières ont lieu à 9h. En cas de besoin, des workshops peuvent être organisés pour échanger sur un sujet en particulier. Le Scrum-Master de la semaine est chargé d'animer les réunions/workshops et d'en rédiger les compte-rendus.

Chaque semaine, une réunion hebdomadaire obligatoire (même si demi-journée de repos posée) est tenue le lundi matin. Elle dure entre 20 et 40 minutes, et contient au moins :
- Un résumé et une rétrospective collective de la semaine précédente (retours, ce qui devrait être changé, ce qui n'a pas été réussi...)
- Planification du Sprint de la semaine

Tous les mardis, mercredis, jeudis et vendredis, une réunion journalière est tenue. Elle dure environ 10 minutes. Chacun s'exprime environ 2 minutes pour :
- Indiquer ce qui a été fait la veille et ce qui doit être fait le jour de la réunion
- Organiser le travail de la journée (avec qui travailler, sur quoi...)

3 Méthodes de travail

3.1 Travail à fournir

Présence obligatoire à tous les cours. Chacun doit visionner les vidéos de cours avant les jours indiqués.

Il faut prévoir un peu de temps tous les jours pour aider les autres, participer à l'écriture de tests, relire une partie de code, etc.

3.2 Développement

L'ensemble du code produit devra être vérifié par des tests, avec pour objectif une couverture de tests de 100%. Les tests de chaque partie doivent être écrits avant de l'implémenter, et doivent être exhaustifs. Si le temps le permet, deux personnes doivent rédiger chacun des tests, dont au moins une qui ne participera pas au développement de la partie testée. Cette personne sert de "Candide".

Le maximum de méthodes/fonctions doivent être écrites en de manière "défensive" (se référer au polycopié). Une `javadoc` doit également expliquer les différentes valeurs d'entrée, de retour et le rôle de la méthode concernée. La convention de codage Java doit être respectée (largeur d'indentation de 4 caractères, CamelCase pour les variables...). Afin de pouvoir être interprété par le plus grand nombre de personnes, l'ensemble du programme doit être écrit en anglais. Cette conformité ne s'impose pas lors de contraintes spécifiées par le cahier des charges (messages d'erreurs attendus...).

Chaque partie de code doit être relue par un tiers qui sert de "Candide" (si possible le même jour que l'implantation). Cela lui permet de s'approprier le code et d'apporter une autre critique.

3.3 Outils

3.3.1 Canaux de communication

L'équipe utilise principalement :
- Messenger pour communiquer rapidement des informations récentes
- Discord pour les réunions, avec possibilité d'utiliser le partage d'écran pour des explications visuelles
- Mail pour contacter le corps professoral
- GitLab pour commenter des commits et/ou les demandes de fusion

3.3.2 Versionnage du code

GitLab est utilisé pour le versionnage, en travaillant sur le dépôt `ProjetGL2022/G2/GL07`.

La branche principale `master` est mise dans un état protégé de façon à ne pas pouvoir commit directement des changements dessus. Cette branche ne devra pas être une branche de travail, elle fera office de branche stable.

Les développeurs devront travailler sur une branche séparée pour implémenter des fonctionnalités, et pourront, après avoir une branche satisfaisante (fonctionnalité implémentée, bonne couverture de tests, revues de code effectuées), faire une demande de fusion vers la branche principale `master`.

Les demandes de fusion nécessitent au moins une approbation qui ne provient pas du créateur de la branche ou d'un contributeur de la branche. Ceci permet, d'une part à forcer un candide à vérifier le code de la requête, d'autre part à garder une branche stable avec un historique propre. Ces branches peuvent faire office d'interface pour une revue de code, en utilisant le préfixe "Draft :" dans le titre de la requête pour empêcher une fusion de se faire par erreur.

Les commentaires qui sont fait dans les commits ou dans les demandes de fusion doivent être précis, clairs et exhaustifs. Un exemple de conventions pour écrire proprement des commits.

En cas de doute concernant des commandes Git ou l'utilisation de GitLab, se référer au responsable Git.

3.3.3 Gestion de projet

Trello est utilisé pour la gestion de projet. Ce logiciel nous permet de gérer la répartition des tâches. Chaque tâche est rangée selon son état courant : à prévoir, pour le sprint courant, non commencée, en cours, terminée. De cette façon, chaque tâche est réalisée une seule fois, et les différents membres du groupe sont au courant du travail de chacun.

Il sera également demandé d'indiquer le temps approximatif passé pour réaliser chaque tâche. De cette façon, nous pourrons obtenir la répartition des durées sur les différentes parties du projet à la fin.

Les tâches sont créées par le Scrum-Master avant la répartition de ces dernières. L'ensemble de l'environnement Trello est géré par le responsable Trello. Le responsable Trello s'assure de la position de chaque tâche dans le tableau, identifie les anomalies, et reprends ses collègues si des erreurs ou des oublis ont été commis.

Les tâches ne sont pas réservées uniquement pour la programmation, mais pour tout l'éventail suivant : analyse, programmation, tests, relecture.

3.3.4 Rédaction de documents

Overleaf est utilisé pour rédiger les différents documents à rendre. Cet environnement permet de collaborer pour la rédaction de documents écrits en Latex.

La documentation d'une partie de code doit être écrite par ses développeurs, le plus rapidement possible pendant ou après l'implantation, si nécessaire avec l'aide du responsable documentation.

Afin de rédiger des documents propres et uniformes, quelques règles s'imposent :
- Utiliser le modèle (*modele.tex*) pour chaque nouveau document. Il y a ainsi un en-tête (titre du document, numéro de l'équipe, nom de l'école), un pied de page (numérotation).
- Chaque image, capture d'écran ou graphique doit présenter un titre et une légende.
- Le langage utilisé doit rester professionnel. L'utilisation du pronom "on" est proscrite, nous utilisons le pronom "nous". Il n'y a pas de " :" après un titre.
- La relecture est exigée. Un correcteur automatique comme bonpatron.com peut se révéler efficace.
- "Ensimag" ne s'écrit pas tout en majuscule. Le nom de l'école est : Grenoble INP - Ensimag, UGA.

3.3.5 Environnement de développement

Pour chacun des membres de l'équipe, le projet est réalisé sous une distribution Linux ayant les outils nécessaires d'installés au préalable. Intellij IDEA est l'environnement de développement principal utilisé par les membres du groupe. Cet environnement simplifie la gestion de Git (branches, commits...). D'autres fonctionnalités intéressantes sont aussi intégrées : console, création de diagramme UML automatique, génération de code pour les méthodes de bases. Pour l'automatisation des tests, des scripts Shell sont utilisés.

Les différents signataires s'engagent à respecter cette charte pendant toute la durée du projet.

Fait à : SAINT-MARTIN-D'HÈRES

Le : 10 janvier 2022

Signatures

Aurélien VILMINOT	Damien CLAUZON	Guilherme KLEIN	Léon ROUSSEL	Pierre ARVY