Licence 2e année - Semestre 3

Gestion du projet S3

Vincent Dugat

7 octobre 2019

Table des matières

1 Introduction générale

Le projet de S3 démarre au tout début du mois d'octobre et sera validé la première semaine de la rentrée de janvier. Il dure donc trois mois pleins, et représente environ 1000+ de vraies lignes de code (les commentaires ne comptes pas).

1.1 But du projet de S3

- Faire le lien entre les différents savoirs et compétences acquises dans les UE de L1 et L2.
- Illustrer des principes de ré-utilisasion de code, modularité, tests, documentation sur un programme plus ambitieux qu'un exemple de TP forcément plus limité.
- Rendre indispensable une organisation et une méthodologie de développement (préliminaire à l'étude des méthodes industrielles de gestion de projet)

Dans ce but, le projet que vous allez programmer ne se limite pas au code informatique. Il intègre une méthodologie de travail. En clair le projet comporte deux aspects :

- Le projet de programmation proprement dit (l'implémentation) que nous appellerons *partie technique*,
- Le cadre organisationnel que nous appellerons *partie gestion de projet ou pilotage de projet*[1] qui est indépendant du sujet proprement dit[2].

Le présent document traite de l'aspect organisationnel et donc de la gestion du projet. La partie technique (c'est à dire le détail de ce qu'il faut programmer) est traitée dans un document séparé. Chaque partie sera notée sur 20 points et la note finale sera la moyenne de ces deux notes (ceci est détaillé par la suite).

Dans le cadre de la L2 informatique, il y a une U.E. projet par semestre. Pour le S3, **le projet est individuel**. Les semestres postérieurs vous donneront l'occasion de travailler en équipe de plusieurs personnes. Précisons qu'il y aura en S4 des CTD de pilotage de projet.

1.2 La gestion de projet

Dès qu'un projet de programmation devient complexe, on emploie des techniques de gestion de projet. Ces techniques sont classiques dans l'industrie tous secteurs confondus. Dans le cas du logiciel on utilise des techniques de génie logiciel. Ces techniques seront abordées dans les semestres postérieurs. Pour ce projet de S3, nous allons utiliser un simple principe de management : PLAN, DO, REVIEW (ou PLAN, DO, CHECK, ACT).

L'objectif minimum est d'écrire une feuille de route définissant l'ensemble de ce qu'on va coder, avec quelle organisation et en quel temps. Un façon de le faire est d'identifier toutes les tâches, d'évaluer le temps nécessaire pour réaliser chacun d'elles, leurs dépendances, et de faire un échéancier global ou apparaît le début et la fin supposée de chaque tâche, les tâches qu'on peut réaliser en parallèle, les dépendances temporelles (X ne peut commencer que quand Y est terminée).

On peut résumer ces informations par un graphique : le diagramme de Gantt.

1.3 Les questions à se poser avant de commencer à coder

- **PLAN :**
  - Réfléchir à ce que vous allez/devez faire (compréhension du problème, spécifications informelles et formelles, prévisionnel)
  - Diviser le travail en sous-parties
  - Que quoi avez-vous besoin : machines, logiciels, environnement de programmation ?
  - De quelles connaissances/savoir-faire avez-vous besoin, allez-vous devoir les apprendre ?
  - Comment allez-vous tester les parties, le logiciel, quel peut être *a priori* le degré de sûreté de votre code ?

  Si on reprend ces points de manière plus détaillée :

  Le cadre de développement

  - Sur quelle(s) machine(s) et environnements(s) allez-vous développer ? Est-elle assez puissante, fiable, accessible ? Permet-elle d'atteindre les objectifs demandés ?
  - Y a-t-il tous les logiciels nécessaires (compilateurs, éditeur, Doxygen, utilitaires systèmes, communication, etc.) dont vous avez besoin ?
  - Comment gérer les sauvegardes du code en cas de crash ? (GitHub, dropbox, repl.it, clé USB, NAS, sauvegarde incrémentale automatique ? c'est l'admin système qui le fait ?,...)

1. Pour certains le terme "gestion de projet" intègre aussi l'aspect financier.
2. La gestion de projet intervient dans tout projet industriel : travaux publics, construction aéronautique, industrie etc.

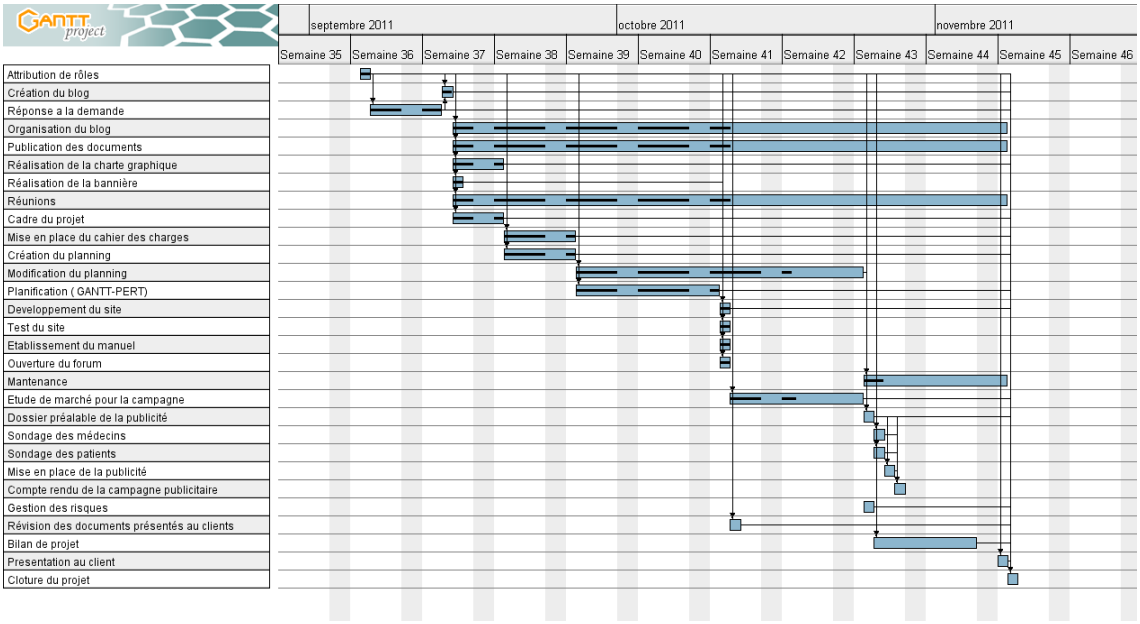

FIGURE 1 – Exemple de diagramme de Gantt

FIGURE 2 – Le cycle PDR

— Y a-t-il des softs à installer, des réglages à faires avant toute chose ? Combien de temps cela va-t-il prendre ?

Le projet

— Quelles structures de données : matrice, vecteurs, autres, combien, pourquoi, quels types ?
— Combien de fichiers de code, quels rôles, quelles sont les SD qu'ils partagent, comment ?
— Fichier header : combien, quels rôles ?
— Gestion du Makefile
— Quel ordre dans le développement des fonctionnalités ? Y a-t-il des dépendances ?
— liste exhaustive des fonctionnalités, des variables globales, des paramètres nécessaires à chaque fonctionnalité

Gestion du projet

— Quelles étapes principales voyez-vous ? Leurs ordre temporel ?
— Evaluation du temps de développement de chaque étape
— Comment tester chaque étape, fonctionnalité, sous-programme, bout de code ?
— Etes-vous capable de lister les tests avant même de commencer à développer ?

— Quel temps prévoyez-vous pour les tests ?
— Pouvez-vous vous projeter dans le temps et planifier les développements et les tests au point de donner une date de fin pour chaque étape (développement et tests) ? Pour le projet aussi ?
— Comment allez-vous gérer l'avancement du projet ? Les retards éventuels ?

- **DO :**
  — Obtenir ce qui manque
  — Apprendre les nouvelles compétences
  — Programmer les différentes parties et les tester en suivant le plan
  — Assembler les parties et tester les assemblages ou le programme final

- **REVIEW :**
  — En cours de développement :
    — Suis-je en train d'avancer vers la résolution du problème de départ ?
    — Ai-je besoin de modifier le plan ? Pourquoi ? Comment ?
    — Si oui, modifier le plan, documenter la modification
  — En fin de projet :
    — Ce qui marche, ce qui ne marche pas
    — Ai-je atteint les objectifs ? Pourquoi ?
    — Ai-je appris des choses ?
    — Comment ai-je résolu les problèmes ?
    — Suis-je resté en deça de mon savoir-faire ? Ai-je augmenté mon savoir-faire ?
    — Me suis-je challengé moi-même ?
    — Mon plan de départ était-il juste ? Ai-je dû le changer ? Selon quelle proportion ?
    — Ce projet m'a-t-il été utile ?

2 Exposé du travail à faire dans cette partie : Les livrables (20 pts)

Afin de communiquer avec les autres parties (client, chef de projet), dans notre cas le tuteur, le responsable d'UE, un certain nombre de documents vont être demandés avec une date butoir. On les appelle les livrables. **Ils sont à déposer sur Moodle exclusivement, en respectant la date butoir.**

Donc, ce sont des documents à rendre. La clarté et la lisibilité de vos documents sera prise en compte. Tout écrit confus, superficiel, alambiqué risque d'être peu rétribué. Le but de cette documentation est d'être informative dans une entreprise où les développeurs passent d'un projet à l'autre [3].

2.1 Document "organisation et feuille de route" (Plan) 6 pts

La partie technique impose des structures de données et des prototypes de sous-programmes C. Vous disposez également d'une documentation HTML générée avec Doxygen. Vous devez :

- Confronter cette documentation et le header avec le cahier des charges du projet.
- Repérer ce qui manque en terme de SD et de sous-programmes et le spécifier (prototypes).
- Etablir la structure globale de votre programme : identification des différentes parties, modules, sous-modules, interaction entre les parties.
- Faire la liste des tâches à réaliser, établir leurs dépendances, estimer leur durée, les organiser dans le temps.
- Créer un projet Project Libre avec toutes ces données.

La feuille de route peut être une simple feuille de tableur. Vous pouvez aussi utiliser un logiciel dédié

3. Dans la vraie vie (professionnelle) on lit plus de code qu'on en écrit. C'est pourquoi on doit programmer de la manière la plus lisible possible car on va être lu. C'est une question d'efficacité générale (de l'entreprise, du labo, du projet libre, de l'asso de développeur, etc.)

comme Project Libre (voir en fin de document) qui permet d'afficher un échéancier sous forme de diagramme de Gantt qui donne la feuille de route de la réalisation de toutes les tâches identifiées ci-dessus.

Remarque : Votre échéancier doit lister les informations administratives d'usage pour la communication avec votre tuteur :

- Nom, prénom, groupe TP, mail UPS
- Code apogée UE (c'est sur la page Moodle), titre du projet, année civile en cours
- Tuteur avec son mail

Le document à rendre est un simple fichier (feuille de calcul ou fichier Project Libre) dont les tâches reflèteront l'organisation du code, et le diagramme de Gantt les dépendances et l'échéancier.

2.2 Etat d'avancement (Plan et Review)

L'idéal est de tenir à jour votre fichier feuille de route au niveau des tâches terminées, validées, en retard et de leur durée effective. Cela vous aidera à vous organiser.

2.3 Doc de bilan (Review) 6 pts

A la fin du projet, vous indiquerez dans un bref document :

- Ce qui a effectivement été fait, ce qui fonctionne et ce qui ne fonctionne pas dans cette version (une simple liste), les bug repérés mais non résolus (un simple tableau avec un pourcentage peut suffire à faire le job).
- Une estimation à la louche du pourcentage de votre plan qui s'est révélé juste (un nombre et deux lignes de commentaires au plus),
- Une estimation du degré de fiabilité de votre programme,
- Un bilan personnel : ce qui vous a posé des difficultés, comment les avez-vous résolues (ou pas), ce que vous avez appris au cours de ce projet, tant sur le plan organisationnel que sur le plan technique. Cette partie inclus une analyse d'échecs éventuels et de la recherche leurs causes. Il s'agit d'un simple débriefing de l'opération "projet" (en quelques lignes).

2.4 Validation (Review) 6 pts

En fin de projet votre programme sera validé. Cette étape est souvent appelée recette dans l'industrie du logiciel :

http ://fr.wikipedia.org/wiki/Test_d'acceptation

En informatique, le test d'acceptation (ou recette) est une phase de développement des projets, visant à assurer formellement que le produit est conforme aux spécifications (réponse donnée à un instant « t » aux attentes formulées). Elle s'inscrit dans les activités plus générales de qualification. Cette étape implique, en la présence effective des différents acteurs du projet, maîtrise d'œuvre et maîtrise d'ouvrage, le déroulement rigoureux de procédures de tests préalablement décrits, et l'identification de tout écart fonctionnel ou technique.

Typiquement cela se passe en deux temps :

1. vous devez faire la démonstration du fonctionnement de votre programme à votre enseignant tuteur,
2. dans un deuxième temps vous devez répondre à ses questions techniques et organisationnelles. Dans cette partie, le tuteur est libre de demander une modification du programme en live ou l'exécution du programme sur un jeu de test qu'il fournira.

Conséquence : absence à la recette = abi à la partie code.

2.5 Utilisation de Doxygen (2 points)

Le logiciel Doxygen permet de générer automatiquement une documentation à partir des commentaires du programmes (ceux qui respectent un certain format). Il vous est demander de générer une documentation pour la partie "algorithme génétique".

3 La programmation (Do) 20 pts maximum

Le sujet de la partie technique est à télécharger sur Moodle. Il sert de cahier des charges et il liste diverses fonctionnalités rapportant chacune, potentiellement, un certain nombre de points. Les fonctionnalités peuvent être regroupées en parties ou modules. Vous devez tout faire dans une fonctionnalité pour qu'elle vous soit créditée.

Tout faire signifie :

1. Avoir implémenté l'ensemble de la fonctionnalité en respectant les spécifications et avoir testé et validé le code qui doit fonctionner sans bug (50% des points).
2. Avoir respecté les règles de programmation propre (clean code) et commenté ce code avec Doxygen (50% des points).

Dans ce cas, et dans ce cas seulement, vous pouvez espérer l'ensemble des points de cette partie.

4 Le problème des tests

L'organisation classique des tests d'un projet consiste à prévoir trois types de tests :

1. Les tests unitaires qui concernent les tests des parties de code les plus élémentaires, celles qui ne font appel à rien d'autre. Ces tests cherchent à être le plus exhaustifs possibles. Il est utile de garder les tests effectués dans un fichier afin de vérifier en cas de doute par la suite.
2. Les tests d'intégration concernent les tests de parties de code qui utilisent les parties élémentaires ou autres, testées par ailleurs.
3. Les tests fonctionnels qui concernent les tests de l'intégralité du code.

Quand on estime d'une partie est suffisamment testée, on la déclare validée. Cela sous-entend que la probabilité qu'il reste des bugs dans un code validé est faible.

FIGURE 3 – L'organisation des tests

5 Ressources

Chaque étudiant aura un enseignant tuteur qui sera son référent pour le projet. Bien sûr, vous pouvez fouiller le web, la BU, vos relations, travailler à plusieurs, mais, **aucun plagiat ne sera toléré. Votre code est personnel.**

Les tuteurs peuvent fonctionner avec des réunions avec les étudiants qu'ils suivent, ou par email, ou peuvent vous contacter pour un bilan. Le tuteur joue le rôle du client, partiellement du chef de projet, et bien sûr de l'enseignant qui va évaluer votre travail.

Un projet informatique (à la fac, dans l'industrie, etc.) est un chalenge technique mais aussi une communication avec vos pairs et votre tuteur. Il sera grandement apprécié de répondre aux sollicitations du tuteur.

Rappel : Dans tous les cas, ne pas venir à la validation, même en ayant déposé un code sur Moodle, sera considéré comme ABI.

6 Synthèse des livrables

Début de projet
- Document technique d'organisation et la feuille de route (un seul document ProjectLibre)

Tout le long du projet(non déposé)
- Mettre à jour la feuille de route avec le temps réel passé les ajouts ou suppressions de tâches.

Fin de projet
- Déposer la doc de bilan.
- Mettre à jour la documentation Doxygen et déposer le code et la doc sous forme d'un fichier zippé.
- Déposer l'état final du fichier "feuille de route" (non obligatoire).

7 Le logiciel Project Libre

Afin de gérer la feuille de route du projet nous allons utiliser un logiciel de gestion de projet : ProjectLibre. Il s'agit d'un logiciel où vous pouvez décrire l'ensemble des tâches et de leurs sous-tâches de votre projet et leur donner une date de début, une date de fin estimée, une étiquette comme : en attente, prêt à coder, en cours de développement, en test, validé, en retard, etc. Le logiciel permet des commentaires, de faire un échéancier. Il existe beaucoup d'autres logiciels bien meilleurs mais celui-ci est libre et gratuit et fonctionne sur toutes les architectures.

`http://www.projectlibre.com/`

Une doc pdf est disponible sur Moodle.