Génie Logiciel 2 : Synthèse

Christian Ingouff
Pierre-Alexandre Tyndal
Sonia Seddiki
Yann Charbonnier

EISTI

2013/2014
Semestre 2
Jalon 5

Table des matières

Partie 1

Introduction

Ce rapport présente une vue d'ensemble du projet et des réalisations en découlant. Nous reviendrons sur les différentes phases qui ont jalonné son déroulement, de la reformulation du cahier des charges jusqu'à la programmation du logiciel et son test.

Nous aurons également l'occasion, en dernière partie, de faire une rétrospective du travail en équipe mené tout au long du projet. La vision apportée sera plus axée sur la partie humaine ainsi que sur les rôles de chacun.

Partie 2

Reformulation du cahier des charges

La reformulation du cahier des charges est le point de départ de notre projet. Elle permet de s'assurer que chacun a bien compris les attentes vis à vis du logiciel demandé, et ainsi de poser des bases sures pour les réalisations à venir.

Une étude rapide du projet en tant que futur logiciel informatique nous a permis d'isoler quelques contraintes implicites à respecter, dans une optique de rigueur, de facilité d'accès et d'optimisation :

Contraintes	Intitulé
C1	Pouvoir interagir avec l'interface
C2	Être le moins spacieux possible
C3	Respecter les normes et législations imposées par le sujet
C4	Être esthétique et ergonomique

Le cahier des charges impose aussi d'autres exigences :

- Langage de développement : Java version 1.7
- Persistance des données : Sérialisation des objets proposée par Java
- Cette technique de sérialisation doit être masquée par une interface (au sens UML) afin de pouvoir changer simplement de méthode de persistance de données
- Analyse et conception en UML
- Utilisation du pattern MVC
- Dans un souci de maintenance et d'évolution possible, les objets doivent communiquer entre eux à travers des interfaces

- La sauvegarde des données se fait automatiquement à la sortie du programme ou à chaque fois que l'utilisateur le demande. Seules les données modifiées, crées ou supprimées doivent être mises à jour lors d'une sauvegarde.

Ci-dessous les principales fonctionnalités attendues. C'est la ligne directrice suivie pour l'élaboration du programme :

- Un administrateur définit :
  - Les utilisateurs
  - Les promotions (listes d'élèves)
  - Les modules, dont il peut définir ou modifier
    - Le nom du module
    - La liste des modules pré-requis
    - Le syllabus du module
- Un utilisateur est défini par :
  - Son nom
  - Son prénom
  - Son rôle : enseignant (préciser liste de modules), élève ou administrateur

- Un professeur peut :
  - Définir des QCM (privés ou publics) : liste de questions
    - Si un QCM est privé, seul le professeur qui l'a créé peut l'utiliser. Sinon, tous les professeurs peuvent l'utiliser.
    - Les questions ont un libellé et présentent une liste de réponses
    - Les réponses ont un libellé et un attribut vrai ou faux selon leur véracité
  - Définir des sessions de QCM :
    - Date de début
    - Date de fin
    - Module associé
    - Promotion associée
    - Nombre maximum de fois pour répondre au QCM
  - Consulter une session de QCM s'il l'a définie :
    - Réponses et score final de chaque élève
    - Statistiques sur l'ensemble des élèves : moyenne, écart-type, fréquence de bonnes réponses par question

- Un élève peut :
  - Répondre à une session de QCM à laquelle il est inscrit, entre la date

de début et la date de fin de session
- Visualiser ses résultats à une session de QCM à laquelle il a participé, à la suite de la session

Pour perfectionner le programme, un peu austère et incomplet, nous avions prévu d'y ajouter certaines fonctionnalités supplémentaires. Elles sont facultatives car non exigées dans le cahier des charges, mais sont toutefois indispensables pour rendre le logiciel utilisables par tous et sans problème de sécurité :

- Authentification : Tout utilisateur devra s'authentifier afin de pouvoir utiliser le logiciel
- Interface graphique : Esthétique et utilisation aisée

Le cahier des charges une fois reformulé, les objectifs sont clairs. Nous pouvons maintenant procéder à l'analyse.

Partie 3

Analyse

La phase d'analyse va nous servir à effectuer une première modélisation du logiciel. Grâce à la crétion de multiples diagrammes, nous allons pouvoir aborder différentes parties de la création du logiciel, comme la création des éléments du projet ou encore le déroulement détaillé d'une fonction. Etant donné que nous allons programmer en Java, nous allons utiliser l'analyse UML (Unified Modeling Language) pour réaliser nos diagrammes.

3.0.1 Diagramme de cas d'utilisation

Ce diagramme définit les acteurs (Utilisateurs) ainsi que les différentes interactions possibles avec le logiciel (Cas d'utilisations). Il permet de voir clairement qui fait quoi (et par extension ce que tel ou tel utilisateur ne peut pas faire suivant son statut), mais également des points de passage oblige pour réaliser un cas d'utilisation. Par exemple, pour consulter les résultats d'une session QCM, il faut déjà avoir répondu à cette session de QCM. Enfin, nous y voyons les différentes extensions d'un cas d'utilisation : consulter une session QCM revient à visualiser le score et les réponses et/ou les statistiques.

FIGURE 3.1 – Diagramme de cas d'utilisation

3.0.2 Diagramme de classes

Nous posons ici les bases du code Java, qui est un langage orienté objet. Autrement dit, la programmation se fait en définissant :

- Des "classes" : les différentes entités de notre modèle, avec leurs attributs et méthodes associées
- Des "associations" entre ses classes : concrètement, ces liens se définissent par exemple par des appels d'instances de classe dans les fonctions d'une autre classe (Exemple : la création d'un QCM qui comporte plusieurs instances de la classe Question, i.e. les différentes questions qui composeront le QCM)

FIGURE 3.2 – Diagramme de classes

Nous exposons donc ici les différentes classes, leurs attributs et méthodes ainsi que les liens qui les unissent, le tout en respectant la norme UML. Notons que ce diagramme a été repensé tout au long du prjet, mais les modifications ont été directement appliquées dans le code sans être à nouveau annotées ici. Par ailleurs, pour des raisons de lisibilité, nous n'avons pas mentionné l'ajout des getters et setters pour chaque attribut de chaque classe. Les setters (de l'anglais "set") et les getters (de l'anglais "get") sont des méthodes permettant respectivement d'affecter la valeur d'un attribut ou de la récupérer (car ces valeurs sont protégées et inaccessibles autrement que via ces méthodes, pour éviter de mauvaises manipulations).

3.0.3 Diagramme d'activité

FIGURE 3.3 – Diagramme d'activité

Ce diagramme décrit dans les détails les étapes nécessaires pour réaliser une fonction donnée. Celui présenté ici décrit la création d'un QCM. Ce type d'analyse est utile pour réaliser les diagrammes de séquence.

3.0.4 Diagramme de séquence

C'est une version du diagramme d'activité qui est plus proche du code. Le même cheminement y est décrit, mais cette fois nous voyons plus clairement quelles sont les classes qui vont intervenir et la manière dont elles interagissent entre elles, i.e. quelles méthodes sont utilisées et dans quel but (création/destruction d'une instance de classe, affectation de valeur entre

autres).

FIGURE 3.4 – Diagramme de séquence

Le diagramme ci-dessus est encore une fois celui de la création de QCM. Il est donc possible de comparer les changements par rapport au diagramme précédent.

3.0.5 Diagramme d'états-transitions

Le diagramme d'états-transitions montre les différents états d'un objet, ainsi que les différents moyens de changer d'état (transitions). Par exemple, le diagramme ci-dessus montre les états et transitions d'une session de QCM. Quand on crée une session, elle est par défaut fermée. Elle devient ouverte quand nous arrivons à la date de début de session. Suivant si on a répondu à la session ou non, celle-ci est considérée comme rendue ou non rendue.

FIGURE 3.5 – Diagramme d'états-transitions

Une fois l'analyse terminée, nous pouvons passer à la conception du logiciel, étape nécessaire avant de passer à la programmation.

Partie 4

Conception

Durant la phase de conception, nous réfléchissons aux solutions techniques que nous implémenterons par la suite, par exemple à la manière de structurer le programme ou encore aux fonctionnalités facultatives à rajouter.

Pour répondre aux exigences du cahier des charges, nous structurons notre programme selon le design pattern MVC (Model-View-Controller), c'est à dire de la façon suivante :

- Le package Model : c'est le coeur du logiciel. Il regroupe toutes les classes contenant ses fonctionnalités principales (i.e. celles définies dans le diagramme de classes)
- Le package View : c'est l'interface du logiciel. Il regroupe donc toutes les classes qui créent cette interface (par exemple, les différentes fenêtres Elève/Professeur/Administrateur) permettant a l'utilisateur de voir ce qu'il fait et communiquer avec le logiciel. Une réalisation graphique à été tentée mais étant donné que nous n'arrivions pas a rafraîchir la page, nous avons été dans l'obligation d'arréter l'idée afin de respecter au mieux les délais
- Le package Controller : permet de faire le lien entre Model et View. Il détecte les actions effectuées sur l'interface (un clic sur un bouton par exemple), contrôle leur validité, et envoie les modifications à opérer au Model (le lancement de telle ou telle fonction associé au clic sur le bouton correspondant)

Par ailleurs nous avons implanté deux fonctionnalités supplémentaires :

- La sauvegarde des données sur un fichier XML à part, pour plus de lisibilité
- Un système d'authentification login/mot de passe, avec un cryptage du mot de passe par la fonction MD5 (fonction de hachage). Le mot de

passe sauvegardé sur le fichier XML est déjà crypté, il est donc plus difficile de pirater le compte d'un utilisateur en ayant simplement accès aux fichiers XML associés

Partie 5

Travail en équipe

5.1 Présentation

Cette partie se focalisera sur l'équipe en elle-même, ainsi que sur son fonctionnement tout au long du projet. Les principaux axes abordés seront les rôles tenus par chacun, la gestion du projet et enfin l'évolution de l'équipe au cours des mois.

5.1.1 Rôles

Comme énoncé dans le livre Team roles at work de Meredith BELBIN, il est important de distinguer le rôle d'une personne dans l'équipe de son rôle fonctionnel. Autrement dit, il faut faire la différence entre ce qu'elle fait d'un point de vue technique (son rôle fonctionnel dans le projet), et sa fonction dans l'équipe (son comportement au sein de l'équipe).

5.1.1.1 Dans la réalisation du projet

Dans l'ensemble, la répartition des tâches s'est faite en fonction des compétences et points forts de chacun afin de gagner en efficacité.

Voici un bref récapitulatif des acteurs du projet ainsi que leurs différentes compétences :

- INGOUFF Christian
  Connaissances : Pascal, C, OCaml, HTML/CSS/PHP, SQL/gestion BDD, VBA

- TYNDAL Pierre-Alexandre
  Connaissances : Pascal, C, Java, OCaml, HTML/CSS/PHP, SQL/gestion BDD
- SEDDIKI Sonia
  Connaissances : C, Maple, SQL/gestion BDD
- CHARBONNIER Yann
  Connaissances : C, SQL/gestion BDD

Etape 1 : Cahier des charges fonctionnel

La reformulation du cahier des charges a été faite par l'ensemble de l'équipe. Cela nous a permis de nous assurer que tout le monde comprenait bien les exigences ainsi que les différentes contraintes techniques imposées.

Etape 2 : Analyse

L'étape de l'analyse était conséquente, nous l'avons donc séparée en deux parties. Dans tous les cas, les tâches étaient réparties de sorte que chaque membre s'est attelé à la réalisation d'un type de diagramme en particulier. Toutefois, nous ne travaillions pas de manière isolée. En effet, l'avancée dans un type de diagramme pouvait révéler les lacunes d'un autre diagramme, et donc améliorer ce diagramme par la suite.

Etape 3 : Conception/Programmation

Cette étape a été retardée par rapport au planning initial, le diagramme de Gantt. Nous avons donc exploité aussi efficacement que possible les facilités de chacun pour rattraper ce retard.

Comme nous pouvons le constater, personne ne conaissait le Java avant le début du projet. Or c'était le langage de programmation imposé pour le logiciel. Toutefois, Christian et Pierre-Alexandre, en tant qu'ex-CPI, avaient une expérience de l'informatique et du code plus développée, ce qui leur permettait, pour une même tâche, de l'exécuter plus rapidement que des ex-CPGE. C'est pourquoi ils ont supervisé la partie programmation du projet. Yann, quant à lui, s'est occupé de toute l'interface graphique du logiciel (l'interface homme-machine), et Sonia de l'écriture des algorithmes des fonctions les plus complexes.

5.1.1.2 Dans l'équipe

Les rôles de chacun dans l'équipe sont un peu difficiles à déterminer, puisque non seulement ils n'étaient pas uniques, mais ils changeaient en fonc-

tion de l'étape traitée. Par exemple

Toutefois, nous pouvons distinguer des rôles principaux basés sur le caractère pour chacun d'entre nous.

- Christian : Bien souvent leader de l'équipe, il structurait l'avancée du travail (Coordinateur, Organisateur). Il proposait également de nombreuses idées (Concepteur)
- Pierre-Alexandre : Souvent rêveur, apportait des idées (Concepteur)
- Sonia : Relançait parfois l'équipe et recentrait le groupe sur le travail (Propulseur). "Parfois" car ce rôle a été quelque peu inhibé pour certaines raisons (cf. Vision "personnelle"). Corrigeait et peaufinait les rapports (Perfectionneur)
- Yann : Proposait des idées (Concepteur)

5.2 Gestion du projet

Une autre partie de l'analyse de l'équipe est sa capacité à gérer l'avancement d'un projet et à le mener à bien. Ceci dépend à la fois de la gestion du temps (respect du planning, efficacité pendant les séances de travail) et de la dynamique de l'équipe (motivation, bonne entente, etc).

5.2.1 Gestion du temps

La gestion du temps disponible était prévue par le diagramme de Gantt en début de projet. Celui-ci tenait compte des différents évènements qui pourraient ralentir l'avancée du projet (cours, vacances, campagnes BDE, etc), mais également les dates de soutenances et un planning prévisionnel de dates de rendu pour les différents livrables.

Le diagramme a été difficile à tenir, notamment par sa mauvaise conception. En effet, les notions qui allaient être abordées par la suite nous étaient totalement inconnues au moment où nous l'avons élaboré. Cela a conduit a une surestimation (ou sous estimation selon les cas) du temps nécessaire à la réamisation d'une tâche donnée. De plus, nous avions conçu notre planning de manière très linéaire et déconnectée : les différentes tâches se suivaient, et une fois l'une finie, on passait à la suivante. Les choses ont été bien différentes dans la réalité, puisqu'on n'a rarement pu considérer une phase comme totalement achevée. En effet, chaque nouvelle tâche révélait quelques oublis, la-

cunes, erreurs dans les tâches précédentes, d'autant plus que certaines d'entre elles étaient étroitement liées. Par exemple, pour l'analyse, certains types de diagrammes ne pouvaient pas être réalisés sans avoir fait un autre type de diagramme avant. Nous avons ainsi fait moult allers-retours entre les différentes parties, afin de les améliorer au maximum et de garder une certaine cohérence.

5.2.2 Dynamique de l'équipe

D'un point de vue général, l'ambiance dans l'équipe était bonne. Il n'y avait pas de conflits particuliers, du moins pas de conflits inutiles.

Toutefois, le fait de ne pas s'être fixés un créneau hebdomadaire pour travailler le projet a généré quelques problèmes de motivation. En effet, nous avons travaillé de façon trop épisodique, avec parfois plusieurs semaines qui s'écoulent sans se pencher à nouveau dessus. Il était donc d'autant plus difficile de s'y remettre que nous ne savions plus, après toutes ces semaines, où nous en étions exactement.
À cela s'ajoute une mauvaise gestion du temps : avoir un délai très long sans jalons imposés est difficile à appréhender, la tentation de remettre au lendemain en se disant qu'on a le temps (et d'autres priorités à plus courts délais sur le moment) est grande. Par ailleurs, il était difficile d'avancer efficacement étant donné que toutes les connaissances nécessaires étaient en cours d'apprentissage. Ainsi, d'une part les notions n'étaient pas toutes abordées à un instant t du projet, et celles qui l'étaient n'étaient pas totalement maîtrisées car nouvelles. Il a donc fallu à chaque fois un temps d'adaptation, le temps de s'approprier ce qu'on apprenait.

Enfin, la difficulté à se réunir en dehors des cours a également freiné l'avancement. Bien que nous ayons trouvé une alternative (cf. "Evolution de l'équipe"), le travail d'équipe qui en découlait n'étaut pas aussi efficace que si nous étions tous réunis autour d'une table. Il manquait un certain lien social en somme.

5.3 Evolution de l'équipe

5.3.1 Rappel : cycle de vie de l'équipe

Une équipe possède différents stades de vie :

- Etape 1 : Constitution
  Il s'agit de former une équipe poursuivant un but commun (ici, l'aboutissement du projet).
- Etape 2 : Stabilisation
  Les membres de l'équipe apprennent à travailler ensemble et préparent le terrain pour un bon avancement du projet (répartition des tâches, des rôles, etc)
- Etape 3 : Intégration
  Les membres de l'équipe développent une synergie accrue, autrement dit, se complaisent de plus en plus dans le travail de groupe. L'équipe n'en devient alors que plus efficace. C'est aussi le moment où la définition des tâches se précise.
- Etape 4 : Réalisation
  L'équipe devient toujours plus productive car elle met l'accent sur une communication de plus en plus efficace.
- Etape 5 : Maturation
  La notion d'équipe est totalement intégrée par les différents membres. Ceux-ci joignent leurs efforts pour faire progresser le projet et l'équipe. La communication et le partage d'informations est plus élevé.
- Etape 6 : Dissolution
  À la fin du projet, les membres se séparent.

5.3.2 Evolution de l'équipe

En se basant sur le cycle de vie décrit au-dessus, nous allons tenter de suivre le cheminement de notre équipe.

La formation de l'équipe s'est faite par cooptation. Nous nous connaissions tous plutôt bien, certains plus que d'autres. Par exemple, Christian et Pierre-Alexandre se connaissaient car ils étaient en CPI à Pau ensemble et avaient déjà coopéré pour le projet de Génie Logiciel 1. Christian et Yann disposent également d'un lien tissé par le Projet Différencié du premier semestre.

Nous avons vite mis en place un certain système de travail. Nous avons commencé par établir un diagramme de Gantt à partir du peu d'informations que nous connaissions. Puis nous avons rédigé le premier livrable (Reformulation du cahier des charges). Cette étape était quelque peu chaotique. Nous cherchions une façon de travailler qui soit compatible avec une contrainte majeure : nous ne pouvions pas nous voir facilement en dehors des cours. Le premier livrable a été rédigé pendant les séances de Génie Logiciel, mais

aussi parfois après les cours. Dans tous les cas, tout se faisait pendant que nous étions à l'EISTI.

Toutefois, cette rédaction nous a pris beaucoup de temps, car notre partage d'informations était cruellement limité. Nous avons donc mis en place des outils pour pallier ce problème :

- L'utilisation de *GitLab*, un logiciel permettant le partage de code en ligne
- L'utilisation de *ShareLaTeX*, un logiciel pour rédiger des rapports en LaTeX en ligne

Ces deux outils nous ont permis de travailler ensemble, comme si nous étions réunis autour d'une table, alors que nous étions chacun chez nous. Toute modification était instantanément enregistrée et visible par tous.
Cependant, ces outils n'étaient pas connus de tous, et il a fallu attendre la fin du projet pour que tous se soient suffisamment familiarisés avec. Cette progression était visible tout au long du projet, d'autant que les membres de l'équipe se sentaient de plus en plus à l'aise ensemble, nous étions tous de plus en plus habitués à travailler ensemble.

Cette habitude a également été catalysée par l'activité extra-scolaire : le développement des liens entre Christian, Yann et Sonia ont été aidés par leur participation et coopération à l'association *E-EISTI*.

5.4 Vision personnelle

Ci-dessous la vision personnelle de chacun sur le projet, et son ressenti dans l'équipe.

5.4.1 Christian

Motivation

Si je devais résumer la grande ligne de ce projet de Génie Logiciel 2 en un seul mot, le mot que je choisirais plus que volontiers pour celui-ci serait : "motivation".

Effectivement, nous pouvons parler de motivation (et de *motivations*) tout au long de ce projet. Nous avons commencé avec une motivation frôlant

le tapis de roses en recevant le sujet de Génie Logiciel 2. Ayant déjà effectué du traitement de questionnaires dans le projet de Génie Logiciel 1 (ce qui était déjà perçu comme moyennement intéressant) et ayant auparavant entrepris le Projet Différencié du premier semestre (traitement d'images, un sujet plus attrayant à notre avis), la réaction générale en étudiant le sujet de traitement de QCM était pessimiste. C'est une réaction que je relève pour la promotion entière, pas uniquement pour mon groupe. Par exemple, des citations démonstratives disaient déjà *"boucler le projet GL2 dans les deux dernières semaines"*.

Cette première trame constitue l'axe intérieur au projet du manque de motivation. Vient maintenant le rayonnement extérieur : je pense que nous étions davantage enclins à travailler d'autres matières que le Génie Logiciel. D'une part, certains y voyaient plus d'intérêt, d'autre part, la configuration de la matière parmi les modules (le Génie Logiciel se trouve parmi le module de la RH, des langues vivantes et des activités extérieures, un *"module à points"*) n'encourageait pas à mettre trop d'efforts dessus.

En conséquence, la manière dont nous avons entrepris le rendu des livrables commençait étrangement toujours par le type de phrase suivante, dans un créneau de groupe : *"Bon, ça serait bien si on le rendait ce livrable !"*.

Ressentis

D'un point de vue plus émotionnel, le bilan d'ensemble est assez particulier. D'abord, disons que ce projet a davantage été entrepris avec une philosophie utilitariste de ma part, ce qui a fait que je me retrouvais souvent en isolation personnelle dans mon travail pour être le plus productif possible. Je suis parmi ceux qui travaillent beaucoup mieux en solitaire qu'en équipe, et il m'est déjà arrivé de faire un rendu 100% personnel pour un projet d'équipe dans le passé.

5.4.2 Sonia

Etant donné que nous avons déjà parlé de l'équipe vis à vis du projet, je me concentrerai ici sur mon ressenti dans l'équipe.

Je suis arrivée dans cette équipe en connaissant bien Christian et Yann, pour les fréquenter régulièrement à l'association E-EISTI. Toutefois, je ne connaissais pas autant Pierre-Alexandre. J'étais confiante, sachant que mon

équipe était sérieuse. En outre, n'ayant pas trop confiance en mes talents en programmation, j'étais rassurée d'être avec des étudiants qui avaient plus l'habitude de l'informatique que moi.

C'est d'ailleurs ce qui a été mon principal problème. Je n'osais pas proposer d'idées, ou remettre en question les idées des autres quand elles me paraissaient perfectibles, car je partais du postulat qu'il avaient plus de connaissances que moi à ce sujet, plus l'habitude de gérer des projets informatiques, et que par conséquent toute intervention était inutile puisqu'au fond je ne savais pas de quoi je parlais en comparaison à eux. Je me voyais donc apporter mon aide dans une approche exécutive et non décisionnelle. Ainsi, j'ai eu de façon récurrente l'impression de ne pas être grandement utile. Mais je ne voulais pas pour autant confirmer mes craintes aux yeux des autres (et à moi-même). J'évitais donc très souvent de poser des questions quand quelque chose ne me paraissait pas tout à fait clair, pour ne pas paraître totalement incompétente et dépourvue d'initiative. Et j'ai privilégié la prise d'initiative parfois, quitte à recommencer car mon travail ne correspondait pas totalement aux attentes du reste de l'équipe, plutôt que de demander ce qu'on attendait clairement de moi. Une perte d'efficacité certes, mais un gain personnel en autonomie. Paradoxalement, ça m'a aidé à avoir plus confiance en ce que j'étais capable de faire.

Malgré tout je ne me suis pas sentie rejetée, bien au contraire. Mais j'ai eu l'impression que notre équipe aurait pu être bien soudée. J'avais en permanence la sensation qu'il y avait deux types de fractures dans l'équipe :

- La fracture entre les membres qui étaient à E-EISTI (Christian, Yann et moi) et Pierre-Alexandre. Je le voyais moins souvent que les autres, et je n'ai pas réussi a créer de lien avec lui au sein de l'équipe. Quand j'avais quelque chose à dire, je le partageais plus souvent avec Yann ou Christian
- La fracture entre moi et le reste de l'équipe. L'impression que parfois je n'arrivais pas à suivre leurs conversations, que mon niveau était un cran en dessous. Et paradoxalement, ressentir dans cette situation une fatalité immuable, comme si je n'arriverais jamais à rattraper leur niveau. Paradoxalement, car c'est totalement contraire à ma vision des choses en général

Le principal point que je corrigerai à l'avenir sera donc de mettre l'accent sur une communication plus étroite avec les autres membres, et ne plus garder ce genre de problèmes secret. Arrêter les non-dits pour mieux m'intégrer et mieux avancer avec mon équipe. La manière dont je me suis détachée de

l'équipe est quelque chose que je ne m'explique toujours pas. Je n'avais pas eu ce genre de problèmes durant le premier projet Génie Logiciel.

5.4.3 Pierre-Alexandre

Se basant essentiellement sur des notions en cours d'étude, ce projet a été compliqué à planifier. L'attente de connaissances a généralement joué en notre défaveur, nous mettant en retard en raison de sous-estimation du temps de réalisation de certaines tâches. De plus, étant une poursuite partielle de l'étude de Génie Logiciel réalisée au premier semestre, la vision du projet était en partie déjà visualisée dans les esprits. Cependant, l'attente de connaissances sur certains outils a fait en partie baisser le rythme et la motivation.

5.4.4 Yann

Concernant ma vision personnelle sur ce projet, voici mon ressenti sur ce que ce second projet de Génie Logiciel voulait réaliser : une simulation d'un projet classique en entreprise.

Tout les éléments théoriques étaient là : le planning par diagramme de Gantt, la conception par analyse orientée objet, l'encadrement par le département de RH pour la partie d'encadrement d'une équipe, des rendus par Jalons avec des dates que nous avions fixées.

Pour ma part j'ai eu du mal à me motiver pour ce projet car je trouvais le sujet pas très intéréssant, il n'était qu'une application directe de nos cours : une continuation du projet Génie Logiciel 1, contrairement au projet différencié de début d'année qui nous demandait plus d'initiative pour la réalisation des fonctions demandées.

La partie qui aurait pu être intéréssante aurait été celle de la création d'une interface graphique complète car elle demande un peu plus d'imagination et concrétise réellement le projet. Toutefois, cette partie est considérée comme un bonus et la matière est enseignée très tard : nous avons eu notre dernier cours dessus la semaine dernière.
Concernant mon ressenti sur l'équipe , j'ai appris quelques notions grâce aux cours de travail en équipe. Toutefois je n'ai pas vraiment de choses à dire sur notre équipe en elle-même. La motivation joue beaucoup là dedans, j'aurais aimé avoir un rôle qui demande un peu de réflexion ou d'imagination, ce que j'aurais pu trouver dans la partie d'interface Java mais comme je l'ai dis

précedemment cela n'a pas été possible.

Partie 6

Conclusion

À travers cette synthèse, nous sommes revenus sur les différentes étapes du projet, ce qui nous a permis d'en dégager une vision claire. Nous avons pu isoler les points forts ainsi que les écueils à éviter en vue d'un futur projet. La gestion du temps est clairement un point qu'il ne faudra pas négliger.

Nous avons également pu faire un bilan de l'équipe, avec atouts et faiblesses, ainsi que le ressenti de chacun. Par le biais de cette réflexion, nous connaissons désormais les conditions nécessaires à chacun pour s'épanouir dans le travail de groupe.