Projets d'informatique au niveau BSc : défis et solutions

mai 2014

Joel Allred

joel.allred@unifr.ch

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

UNIFR Did@cTIC

Table des matières

Chapitre 1

Introduction

Ce travail de fin d'études a été effectué dans le cadre du *Diploma of Advanced Studies* de didactique universitaire (Did@ctic) donné à l'Université de Fribourg, Suisse. Il porte sur des aspects pratiques des projets d'informatique à l'université. En particulier, une analyse approfondie est menée sur le cours *Contrôle de processus*, issu du programme du deuxième semestre du Bachelor en informatique à l'Université de Fribourg.

Le type de cours et la diversité des étudiants qui y participent constituent un cas d'étude intéressant. Il s'agit d'un projet de programmation orientée objet qui s'effectue en petits groupes. A l'issue du semestre, les étudiants rendent le résultat de leurs efforts de programmation, accompagné d'un rapport. Le fait que ce soit un projet de groupe pose un certain nombre de défis, notamment pour l'encadrement et l'évaluation, et qu'il convient d'étudier.

Ce travail, sans avoir la prétention de délivrer une formule miracle, permet néanmoins de dégager quelques pistes sur les problèmes posés, en analysant plusieurs formules qui on pu être mise en œuvre durant les trois dernières éditions de ce cours. L'on pourra se baser sur les différentes analyses faites durant la formation Did@ctic, ainsi que sur le matériel de cours et les divers documents fournis par les étudiants.

Avant d'analyser les différentes modalités qui constituent le projet, il est nécessaire de définir quels sont les intérêts d'un tel projet et quels buts on désire atteindre.

Chapitre 2

Aspects généraux des projets en informatique

Les projets occupent une place prépondérante dans l'enseignement de l'informatique à tous les niveaux, notamment dans le domaine de la programmation. Plusieurs raisons sont à l'origine de ce phénomène. Premièrement, l'aspect pratique est primordial dans l'apprentissage de l'informatique. On peut difficilement espérer acquérir une quelconque compétence en se bornant aux ouvrages et aux cours ex cathedra. Deuxièmement, la gestion d'un projet, de son début à sa fin constitue un but d'apprentissage en soi. Là également, c'est un aspect qui ne peut s'apprendre s'en s'être vécu. Troisièmement, le fait de travailler en groupe est également un but d'apprentissage. Contrairement aux idées reçues, les métiers de l'informatique revêtent un caractère hautement collaboratif et la nécessité de se comprendre et de se coordonner avec ses pairs apparaît rapidement.

2.1 Importance de la pratique

Les cours «traditionnels» d'informatique (unités d'enseignements théoriques, avec exercices ou non) délivrent des connaissances indispensables à l'apprentissage de la matière, mais ils sont insuffisants pour la quasi-totalité des branches enseignées dans le milieu informatique, car le «savoir-redire» n'a que peu de valeur face au «savoir-faire».

Prenons l'exemple d'un cours de programmation en Java [Ora14]. On peut s'ima-

giner les concepts peuvent être expliqués complètement de manière descriptive, notamment avec la présentation d'exemples concrets. Indépendamment de la qualité d'un tel cours, les étudiants n'auront pas acquis de «compétence» en programmation, mais uniquement des connaissances des différents concepts. C'est pourquoi tout cours d'informatique qui se respecte inclus une partie pratique, soit sous la forme d'exercices, soit sous la forme de mini-projets. Durant ces temps de pratique, l'étudiant à l'occasion d'appliquer les connaissances acquises durant les cours et ainsi développer ses compétences en programmation.

2.2 Compétences de gestion de projet

Si les exercices de programmation donnés dans le cadre d'un cours sont indispensables pour l'apprentissage de l'aspect pratique de la programmation, la gestion de projet est aussi un but d'apprentissage. Dans de nombreux domaines de l'économie, les travaux informatiques sont gérés par projets, où l'échelonnement des tâches, la répartition du travail et les contraintes temporelles sont gérées de manière spécifique.

Bien que le cours en question ne soit pas un cours de gestion de projet, théorisant les concepts fondamentaux en la matière, celui-ci permet une première approche pratique au fait de travailler en groupe sur un projet complexe avec une limite de temps bien définie.

2.3 Travail en groupe

Le fait de travailler en groupe sur un même projet peut être une force autant qu'un inconvénient. Si le groupe est bien organisé, il pourra accomplir un résultat bien plus grand qu'une personne seule, bien sûr grâce à la multiplication de la force de travail, mais aussi grâce à l'émulation entre les collègues. Chacun arrivant avec son propre bagage théorique et pratique, les problèmes pourront aussi parfois être résolus plus rapidement que dans un travail autonome.

Chaque médaille à son revers, et il convient de bien analyser les différents motifs d'échec d'un groupe. Tout d'abord, il faut différencier deux types d'échec :

1. échec du projet
2. échec personnel

Échec du projet L'«échec du projet» se produit lorsque le groupe, incapable de trouver suffisamment de ressources, ne parvient pas à finaliser le projet. Le projet est donc abandonné en cours de route, ou alors terminé mais insuffisant par rapport aux critères d'évaluation.

Échec personnel Le deuxième type d'échec (qui correspond d'avantage à une situation académique que professionnelle) est celui où, indépendamment du fait que le projet soit mené à bien ou non, ne parvient pas à acquérir les compétences prévues dans les objectifs d'enseignement, ou a fourni un travail qualifié d'insuffisant. Transposé au monde professionnel, cela pourrait signifier un licenciement pour cause de sous-productivité.

Cet aspect révèle toute son importance lors du choix de la méthode d'évaluation. Plusieurs questions se posent alors :

— Evalue-ton le résultat du projet ou les compétences personnelles ?
— Un étudiant peut-il échouer alors que le projet est réussi ?
— A l'inverse, peut-on donner une note suffisante à des étudiants dont le projet ne l'est pas ?

Il faut là faire des choix et c'est l'un des objets du chapitre 4.

Chapitre 3

Descriptif du cours

L'objet de ce cours est la programmation dans le langage Java d'un système de circulation de trains électriques. Le laboratoire du cours (Fig. 3.1) contient une maquette de train électrique construite par les Chemins de Fer Fédéraux et pilotée par un ordinateur relié à la centrale de commande du circuit via un boîtier Selectrix (Fig. 3.2). L'interface permet le contrôle des trains, des aiguillages et la détection d'occupation des secteurs. C'est un projet très apprécié des étudiants, car les travaux de programmation sont visibles de manière physique, ce qui n'est que très rarement le cas dans des projets d'informatique.

3.1 Positionnement dans le cursus

Le cours «Projet : Contrôle de processus» est un cours de deuxième semestre de la formation de Bachelor of Science en Informatique. Il est également proposé pour les étudiants dont l'informatique est une branche secondaire, ainsi que pour les étudiants du cursus DAES [1], en remplacement volontaire d'un cours de mathématiques [2].

Il est la suite logique du cours de premier semestre de «programmation orientée objet». Il faut aussi noter que le premier semestre inclut un autre projet (robotique).

1. Diplôme d'Aptitude à l'Enseignement au Secondaire
2. Les étudiants de DAES sont des cas particuliers. Ils n'ont en général pas un bagage suffisant en matière de programmation et bénéficient donc d'un projet adapté, à savoir des exigences réduites par rapport au contenu du projet. Ceci est possible car, pour ces étudiants, le cours est catégorisé différemment dans le plan d'études (avec un autre code de référence). L'organisation d'un même cours pour deux branches différentes est en soi un sujet qui mériterait d'être développé, mais ce n'est pas l'objet de ce travail.

FIGURE 3.1 – Laboratoire du cours de contrôle de processus

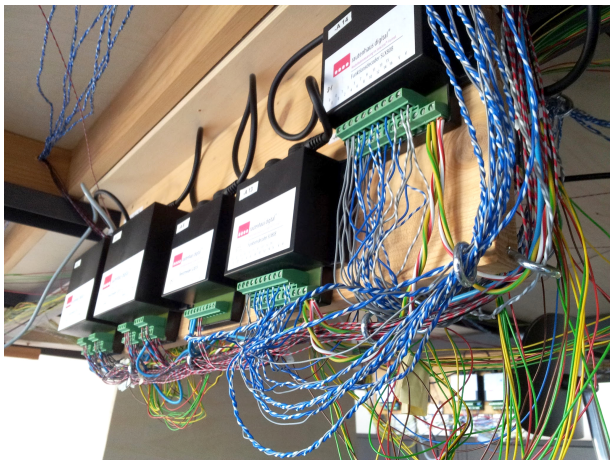


FIGURE 3.2 – Système de contrôle du circuit

Comme tous les cours du Bachelor sont obligatoires, on peut s'attendre à ce que les étudiants aient déjà une petite d'expérience en matière de projets informatiques.

Concernant les connaissances du langage de programmation Java, elle sont censées être suffisantes, car les participants ont logiquement suivi le cours de programmation orientée objet. Bien qu'une connaissance pratique de Java soit nécessaire pour suivre le cours, il y a chaque année quelques étudiants qui arrivent avec un niveau insuffisant. Les raisons sont généralement parmi les suivantes :

— L'étudiant a échoué au cours de programmation orientée objet,
— il a rejoint le cursus en cours de route (par exemple après un changement de branche),
— il est issu de la formation DAES.

La prise en charge des cas sus-mentionnés s'est faite de manière différente selon les années. Jusqu'à l'été 2013, les assistants du cours organisaient des cours d'appui de Java spécifiques au projet. Des ressources bibliographiques externes étaient également à disposition. Depuis le printemps 2014, date de l'adjonction du pré-requis «connaissances pratique de Java» , les cours de soutien ne sont plus donnés, mais les ressources externes sont toujours indiquées. Ceci n'est toutefois qu'en principe vrai, car dans la pratique quotidienne de l'encadrement des étudiants, le professeur et les assistants continuent à répondre à toute question, indépendamment du fait qu'elle concerne le projet en particulier ou le langage Java en général.

S'il est logique de parler des unités d'enseignement qui précèdent un cours, il est parfois utile de considérer celles qui le suivent. Le feedback qu'on peut peut avoir des enseignants qui «récupèrent» les élèves au semestre suivant est utile pour évaluer la pertinence du dispositif. Ainsi, la suite logique de cours de contrôle de processus est le projet «Technologies web» du troisième semestre. Pour l'anecdote, les enseignants de ce cours-là ont remarqué, par le passé, quelques lacunes dans les connaissances de programmation chez certains étudiants. Ceci a fait prendre conscience que, la manière dont le projet «Contrôle de processus» est conçu permet, comme c'est le cas en général avec les projet en groupes, de voir exceptionnellement des étudiants «passer» sans les connaissances minimales. En effet, dans certains cas, il est difficile d'analyser la

contribution de chacun car le groupe peut couvrir les manquements de certains de ses membres. Cette constatation a amené les responsables du projet à revoir quelques points du cours et les versions 2013 et 2014 du projet comprennent des dispositifs supplémentaires pour éviter les «réussites abusives» , notamment en introduisant une partie personnelle à chaque projet. Bien que ces cas de réussites abusives ne soient pas souhaitables, ils sont difficiles à éviter dans des projets en groupes. Pour ces raisons, les projets devraient être focalisés plus sur le développement de l'aspect collaboratif, plutôt que sur le contrôle des connaissances de programmation. Ce dernier aspect devrait à son tour être évalué de manière précise par un cours spécifique de programmation ou les étudiants sont évalués de manière individuelle.

En ce qui concerne l'évaluation, il y a là aussi quelques spécificités. Jusqu'à l'été 2013, les projets informatiques du cursus BSc en informatique n'étaient pas notés, mais étaient simplement considérés comme «réussis» ou «échoués». Du point de vue de l'évaluation, comme nous le verrons au Chapitre 4.3, il est souvent plus simple de définir la réussite ou non d'un projet, que d'attribuer une note. Cependant, comme ces projets son obligatoires, si un étudiant échouait au projet, il devait automatiquement le refaire l'année suivante. Si le projet n'était pas réussi au cours de la deuxième année d'étude (indépendamment du fait que l'étudiant ait participé au projet en première année ou non), l'étudiant était irrémédiablement exclu du programme du BSc en informatique, pour ne pas avoir réussi à accomplir la première année du cursus dans le temps imparti, qui est de quatre semestres. On pouvait donc être exclu de l'Université sur le seul critère de la réussite de projet, donc la pression sur les épaules des enseignants du projet était grande. Mais depuis le printemps 2014, le projet est évalué par une note, ce qui pose certains problèmes supplémentaires pour l'évaluation, mais supprime le caractère critique de la décision, car une note insuffisante peut être compensée par un autre cours.

3.2 Scénario pédagogique

Le cours a fait l'objet d'une analyse approfondie dans le contexte de la formation Did@ctic. La création du scénario pédagogique (cf. Annexe A) a aidé a recentrer les

objectifs du cours et a aussi mis en lumière des améliorations possibles, notamment pour ce qui est de l'évaluation (cf. Chapitre 4.3). Pour le printemps 2014, les responsable du cours sont :

— Prof. Dr. Ulrich Ultes-Nitsche
— Samuel Vonlanthen, assistant
— Joel Allred, assistant
— Cedric Lothritz, sous-assistant

En ce qui concerne les langues, le plan d'études ne spécifie pas en quelle langue doit être donné le cours. Officiellement, il doit être donné, soit en français, soit en allemand. Les enseignants donnent donc le cours dans leur langue préférée, mais essaient de répondre aux questions des étudiants dans la langue de ces derniers. La documentation est intégralement en anglais.

3.3 Organisation générale

Distribué sur les 14 semaines du semestre de printemps (voir Annexe D), le cours, dont l'ancrage horaire est le jeudi de 17h à 19h, peut être vu comme comprenant 3 phases principales :

1. Introduction aux concepts et implémentation de base
2. Développement du projet personnel
3. Evaluation

Introduction aux concepts et implémentation de base Cette première phase s'étend sur les semaines de 1 à 7 et alterne entre apports théoriques et travaux d'implémentation. Les cours frontaux portent sur les éléments suivants :

— Introduction au cours (organisation, mode d'évaluation, ...)
— Outils de développement
— Éléments théoriques du projet

En particulier, l'apprentissage des bases théoriques du projet, c'est-à-dire l'assimilation des différents éléments se fait, d'une part, part la présentation des éléments en classe (ici, on parlera de locomotives, de trains, de blocs, de secteurs, d'aiguillages,

d'*observeurs* [Fre+04], ...). En ce qui concerne l'organisation du cours, les groupes de 3 (exceptionnellement 2) sont formés dès la première semaine, et chaque groupe reçoit un squelette du projet, c'est à dire une structure qui contient déjà tous les éléments de base, mais où ces éléments ne sont pas implémentés (le méthodes sont vides). La structure du projet étant elle-même complexe, on ne peut pas demander aux étudiants de partir de zéro, la structure est donc donnée, ainsi que certaines fonctionnalités jugées trop difficiles ou trop fastidieuses à programmer.

Chaque étudiant reçoit ensuite un code d'accès personnel vers le projet en ligne (sur un serveur Subversion, cf. chapitre 3.5).

Parallèlement à cet enseignement en cours frontal, les étudiants vont rapidement appliquer les concepts vus en classe. Après la présentation de chaque élément du projet, les étudiants auront comme tâche d'implémenter l'élément en question, c'est à dire de programmer toutes ses fonctionnalités à partir du squelette donné (voir Annexe E).

Ces tâches sont à faire en groupes [3] et sont en général assez simples. Le but n'est pas de leur donner un important travail de programmation, mais de les familiariser avec le projet dans un premier temps, afin d'avoir tous les outils et une connaissance suffisante pour appréhender la phase suivante.

Le squelette fourni est abondamment commenté, et il est clair à quels endroits le code doit être complété et quel est, sémantiquement parlant, le code à créer.

Développement du projet personnel Lorsque les concepts fondamentaux sont acquis, et que l'implémentation de base est terminée, les étudiants peuvent s'attaquer à la partie personnelle du projet, officiellement appelée «extension». L'extension est personnelle et ne se fait pas en groupe, sauf exceptions pour des extensions particulièrement grandes. Des propositions d'extension sont données, allant de la mise en place d'une interface utilisateur sophistiquée, à un système de détection d'obstacles sur la voie, mais les étudiants sont libres de développer leur propre idée. Cette phase s'étend des semaines 7 à 12. Au cours de cette période, traditionnellement à la semaine 9, chaque

3. Les groupes s'organisent selon leur bon vouloir. Certains décident de programmer ces éléments ensemble, certains décident de se répartir le travail.

groupe doit faire une petite présentation, en classe, de la teneur de son projet. En particulier, les groupes en présentent les spécificités, notamment l'interface graphique et les extensions. Cette présentation est obligatoire et compte dans la note finale (cf. chapitre 4.3).

Durant cette phase, le laboratoire est ouvert aux étudiants (sur réservation parmi une trentaine de plages horaires par semaine) et ces derniers peuvent venir tester leur implémentation et corriger leur code. Ils sont accompagnés par les enseignants lors de leurs quelques premières visites pour donner des consignes et répondre aux questions.

Evaluation La phase d'évaluation correspond au moment où les groupes rendent leur rapport écrit (un rapport par groupe) et donnent la présentation finale de leur projet (semaines 13 et 14). Cette présentation fait office d'examen et se tient donc à huis-clos, avec le professeur et les assistants. Le bon fonctionnement de l'implémentation est démontré et les extensions sont présentées. Le professeur et les assistants posent ensuite quelques questions de contrôle à chaque membre afin de vérifier la compréhension (et par extension, la contribution) de chacun.

Les présentations finales sont réparties sur deux jeudis, mais par souci d'équité, le projet doit être complètement terminé pour tous les groupes le lundi précédent le premier jeudi, date à laquelle le rapport doit également être rendu. Plus aucune modification sur le code n'est admise après cette date.

Par la suite, la note est calculée en fonction de la grille d'évaluation de l'Annexe F et communiquée par voie électronique aux étudiants.

3.4 Répartition entre théorie et pratique

Il s'agit ici d'un projet d'apprentissage. L'impact didactique doit donc être prioritaire sur le résultat technique du projet. Une organisation différente du cours pourrait améliorer le résultat final des projets, autant sur la fiabilité de l'implémentation que sur le contenu de extensions. Mais dans un projet de première année, on préfère mettre

de côté la qualité du produit final et se concentrer sur le processus d'apprentissage des étudiants.

Une des approches consiste à ne pas donner des explications trop complètes sur le code à implémenter, afin de forcer les étudiants à rechercher les informations par eux-mêmes, ainsi qu'à analyser le code donné. Toutefois, une base théorique précise et fiable est nécessaire, notamment en ce qui concerne les éléments du projet (blocs, secteurs, aiguillages,...) car les étudiants ne peuvent pas deviner comment sont conçus ces éléments et aucune ressource externe n'existe. Toutefois, il reste des zones d'ombres dans le projet qui ne peuvent être élucidées qu'en fouillant dans le projet, ce qui contribue à une meilleure compréhension de la structure, et facilite ainsi l'implémentation de l'extension personnelle.

On pourrait être tenté, dans un élan de sur-organisation, de spécifier chaque tâche avec une grande précision, mais on se retrouverait alors dans une configuration cours-exercices, et on perdrait ainsi les vertus de l'organisation particulière d'un projet.

L'apprentissage par projet peut prendre diverses formes, et la forme qu'on lui donne va fortement dépendre de l'objectif pédagogique à atteindre. Aguirre et Raucent [AR02] on formalisé quelques approches dans un document fort intéressant qui se concentre sur l'aspect pédagogique des projets. Dans cet article, ils mettent en avant trois répartitions possibles de la répartition entre théorie et application(fig. 3.3) :

- Le **post-projet d'application** est une structure standard de projet, où la totalité de la théorie est donnée avant la mise en œuvre de l'implémentation.
- Le **post-projet progressif** permet aux étudiants de commencer l'implémentation plus rapidement, mais avec moins de connaissances. La suite de la théorie est donnée en cours de route, en parallèle à l'implémentation.
- Le **pré-projet d'apprentissage** est une version expérimentale plus poussée du post-projet progressif. Les étudiants sont immédiatement confrontés au projet sans outils théoriques. Cet apport théorique vient ensuite et les étudiants sont bien sûrs très réceptifs aux informations reçues, car ils ont des questions concrètes auxquelles il faut des réponses.

Aguirre et Raucent mettent logiquement l'accent sur la troisième méthode. Pour

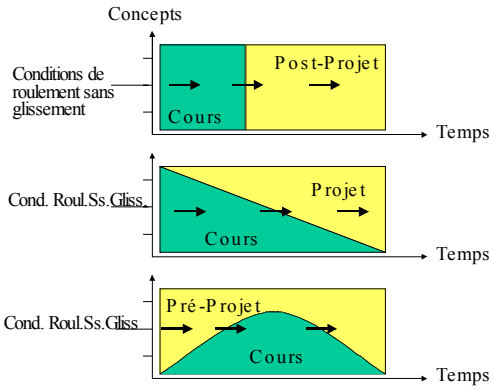


FIGURE 3.3 – Différentes variantes de projet selon [AR02], respectivement *Post-projet d'application*, *Post-projet progressif* et *Pré-projet d'apprentissage*

le projet «Process control», un tel dispositif est un peu idéaliste, car la plupart des étudiants (mais pas tous), n'ont pas encore la capacité, à ce moment de leur cursus, de prendre en main un projet sans un minimum d'explications et d'encadrement. Ceci dit, les enseignants de ce cours essayent de s'approcher au maximum de ce système, en donnant rapidement des tâches d'implémentations, de manière à ce que lorsque toute l'information théorique a été donnée, les groupes ont déjà implémenté une bonne partie de leur projet.

L'avantage de mener en parallèle un enseignement théorique et le développement du projet, c'est que les étudiants, qui ont été confrontés à l'implémentation et aux problèmes qui sont apparus, ont une motivation à être attentifs aux explications pour résoudre leurs propres problèmes. Il peuvent également poser des questions «d'initiés», chose qui n'est habituellement pas le cas dans un cours traditionnel, où l'enseignement par le professeur et la compréhension par l'étudiant sont souvent désynchronisés dans le milieu scientifique.

3.5 Utilisation des TIC

Ce projet utilise de manière intensive les outils informatiques disponibles à l'Université de Fribourg. D'une part, l'implémentation du projet se fait via un serveur Subversion [Apa14], et la gestion du cours se fait intégralement dans le système Moodle

[Moo14], qui est un standard à l'Université de Fribourg.

3.5.1 Subversion

Subversion, ou SVN, est un outil de gestion de versions de projet informatiques. Il est conçu pour permettre à plusieurs utilisateurs de travailler sur un même projet simultanément, sans que cela ne pose de problème au niveau de l'écrasement de données. Dans un cas de travail collaboratif sans gestion des versions, la situation suivante peut se produire :

1. Alice télécharge le document X sur sa machine et le modifie (création de la version X_A).
2. Pendant ce temps, Bob télécharge le même document X sur sa machine et le modifie (création de la version X_B).
3. Alice écrit ses modifications sur le serveur. X devient X_A.
4. Bob écrit ses modifications sur le serveur. X_A devient X_B et le travail d'Alice est écrasé.

Une solution consisterait à verrouiller les documents lorsque ceux-ci sont utilisés, mais cela empêcherait les personnes de travailler de manière parallèle.

Dans le système Subversion, on ne peut écrire sur le serveur que lorsqu'on possède la dernière version du fichier sur notre machine locale. On évite ainsi d'écraser le travail de quelqu'un d'autre par inadvertance. Si les versions ne concordent pas, les différences (ligne par ligne) sont affichées et l'utilisateur peut choisir (toujours ligne par ligne), s'il veut garder la version serveur ou la remplacer par sa version locale.

Un autre avantage d'un système de gestion de versions est, bien sûr, la gestion des versions. Le serveur garde en mémoire toutes les opérations sur les fichiers. On peut donc revenir à une version précédente et annuler des modifications récentes, si on constate que celles-ci ont introduit des erreurs dans le projet. Ce retour en arrière est, à notre connaissance, peu utilisé par les étudiants, qui préfèrent vérifier que leur code est correct avant de l'envoyer sur le serveur.

La mise en place d'un système Subversion n'est pas trivial. On utilise généralement un serveur dédié, mais le Département d'Informatique de l'Université de Fribourg offre ce service.

3.5.2 Moodle

La page Moodle est le point d'ancrage du cours. Il contient toutes les informations utiles aux étudiants (horaires, délais, ...) (fig. 3.4), ainsi que les informations et des outils pour chaque semaine du cours.

FIGURE 3.4 – Haut de la page Moodle du cours

Groupes L'utilisation des groupes dans Moodle permet, d'une part, de centraliser l'information sur la composition des groupes (fig. 3.5) et, d'autre part, de profiter des fonctionnalités de groupes de Moodle.

Groups (9)	Group members	User count
Group 1		3
Group 2		3
Group 3		3
Group 4		3
Group 5		2
Group 6		3
Group 7		2
Group 8		3
Group 9		2

FIGURE 3.5 – Moodle : affichage de la composition des groupes

Forums Parmi ces fonctionnalités de groupe, on peut citer la possibilité de créer des forums dédiés par groupes. La mise en place est simple, il suffit de créer un forum standard, et de sélectionner l'option « separate groups». Ainsi, les participants ne verront que les messages des membres de leur groupe. On crée ainsi un forum qui peut-être utilisé par le groupe pour s'organiser et régler des problèmes, sans communiquer leur secrets de fabrication aux autres groupes.

Un forum technique est aussi présent. Les étudiants peuvent y poser leurs questions et n'importe qui peut y répondre, y compris les enseignants. Malheureusement, ce forum est très peu utilisé (les étudiants qui ont des problèmes n'osent probablement pas en faire part et ceux qui ont des solutions les gardent pour eux). Il est toutefois maintenu chaque année pour des questions de principe. Nous savons qu'il existe des moyens d'accroître l'attractivité d'un tel forum, mais nous ne considérons pas que c'est une priorité. Toutefois, les enseignants l'utilisent à chaque fois que c'est possible pour les conseils d'ordre technique.

Le dernier forum, «news», est utilisé par les enseignants pour les communications officielles (rappels des délais, marche à suivre, ...).

Informations hebdomadaires Chaque semaine, les enseignants affichent les apports théoriques ainsi que les données nécessaires à la réalisation des tâches (fig. 3.6).

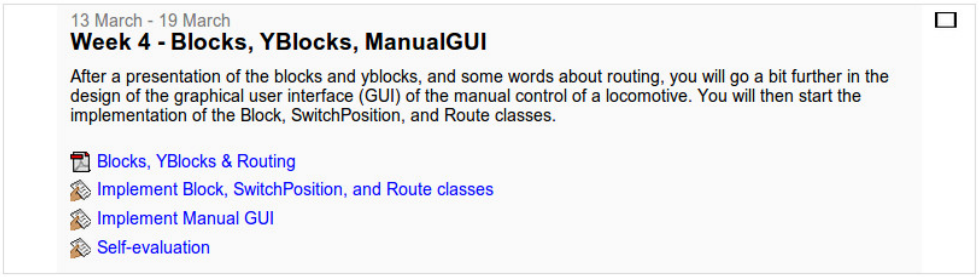


FIGURE 3.6 – Moodle : Information pour le cours de la semaine

Tâches d'implémentation Pour chaque tâche de l'implémentation de base, une «tâche» Moodle est créée et donne les informations sur la tâche à accomplir. Une date de rendu est inscrite, mais elle n'est qu'indicative car seul le résultat final compte dans l'évaluation de l'implémentation. D'autre part, il n'y a rien à «rendre» via Moodle car les étudiants sont seulement tenus de télécharger le résultat de leur implémentation sur le serveur Subversion. Dans Moodle, on parle alors de «devoir hors-ligne». La spécification de la date de rendu est tout de même utile, car c'est à partir de ce moment-là que les enseignants communiquent le feedback sur l'implémentation aux étudiants, toujours via cette tâche Moodle. La séparation par groupes facilite également la transmission du feedback (car ce dernier est souvent identique pour tous les membres du groupe).

Réservation du laboratoire La réservation du laboratoire se fait aussi via Moodle. Il n'y a pas d'outil dédié, mais les enseignants mettent en place pour chaque jour un module «choix» qui contient les différentes plages horaires du jour. La possibilité de limiter le nombre de choix par plage à 1 permet de simuler un système de réservation. Ce système comporte deux désavantages :

1. il est fastidieux à mettre en place pour chaque jour,
2. les enseignants ne sont pas notifiés pour les entrées, ce qui implique de devoir constamment vérifier si quelqu'un a réservé le laboratoire, afin de pouvoir lui ouvrir la porte.

Malgré tout, nous avons opté pour ce système jusqu'à présent car il est directement intégré à Moodle.

Chapitre 4

Points de discussion

Au travers des différentes éditions du cours, année après année, nous avons pu constater certains problèmes, et nous avons essayé de les résoudre de différentes manières, avec plus ou moins d'efficacité. La formation Did@ctic a d'ailleurs beaucoup apporté, autant sur le plan de l'analyse du dispositif, que de la recherche de solutions. Nous dégageons ici trois points primordiaux qui soulèvent de nombreuses questions dans le domaine des projets en groupes :

- L'encadrement des groupes : Quels conseils leur donner ? Quelles libertés leur laisser ? comment prévenir et éviter les problèmes ?
- Le contrôle continu : Quel poids (évaluatif) donner au contrôle continu ? Quelle charge de travail pour les enseignants ? Faut-il prévenir les échecs et comment ?
- L'évaluation : Comment évaluer un groupe ? Comment répartir évaluation collective/personnelle ? Comment quantifier le travail de chacun ?

4.1 Encadrement des groupes

En termes de ressources humaines, on peut difficilement faire mieux que dans ce cours, avec pratiquement 1 enseignant pour 6 étudiants. Pourtant, même dans ces conditions, une mauvaise organisation et des méthodes d'encadrement trop lourdes peuvent rapidement rendre la charge de travail insoutenable. Il convient donc de bien cibler les objectifs et de mettre en place le dispositif le plus efficace.

4.1.1 Constat sur l'autonomie des groupes

Avant de pouvoir définir des objectifs d'encadrement, on doit se poser la question de la nature des étudiants qui participent au projet. Bien que l'auditoire soit assez hétérogène, les étudiants ont en général peu d'expérience en programmation, et encore moins en projets informatiques. Si certains groupes insistent pour se débrouiller seuls, on ne peut pas tous les laisser à eux-mêmes, car l'inexpérience, les lacunes techniques et les importantes différences de niveau peuvent facilement avoir raison du projet.

Les deux types d'échecs ont déjà été mentionnés au Chapitre 2.3. Si l'«échec du projet» est très rare, l'«échec personnel» est plus fréquent. Ce dernier peut avoir deux causes principales :

1. travail insuffisant ou manque de compétence de l'étudiant,
2. mauvaise organisation du groupe.

Outre la propension naturelle de l'enseignant à transmettre une motivation aux étudiants, un échec du à un manquement de ces derniers ne doit pas être considéré comme étant de sa responsabilité. Sur ce point, il faut bien être conscient du double rôle de l'enseignant qui passe subtilement du statut de facilitateur à celui d'évaluateur. Nous reviendrons sur cette dualité au chapitre 4.2.1.

A l'inverse, on se doit d'éviter les échecs qui seraient indépendants de l'étudiant lui-même, par exemple dans le cas où le groupe comprendrait un élément perturbateur. Voici deux exemples de situations problématiques (tirées de la réalité) :

— Désynchronisation :

> Un groupe est formé de 2 personnes : Pierre et Julie. Pierre est très consciencieux et exécute les tâches d'implémentation (1ère phase) très rapidement. A chaque fois que Julie décide de s'attaquer à l'implémentation, elle constate que tout est déjà fait. Ainsi, elle ne réussit jamais à faire les travaux de programmation car Pierre a déjà tout fait rapidement chaque semaine. Lorsque vient la phase de travail personnel, Julie n'aura pas pu, par manque de pratique, se familiariser suffisam-

ment avec le projet et ne pourra pas exécuter son travail personnel de manière satisfaisante.

— Mise de côté : Un groupe est formé de 3 personnes : Anne, Manon et Tanja. Anne et Manon se connaissent depuis longtemps et ont l'habitude de travailler ensemble. Elle se voient donc souvent sans Tanja. Tanja, qui ne parle pas bien le français, a du mal à s'intégrer au groupe. Malgré, sa timidité, elle demande à faire des réunions de groupe pour avancer le projet, sans grand succès. Tanja n'a que peu d'informations sur l'état du projet, ne trouve pas de réponses à ses questions, et perd graduellement sa motivation. Au final, elle ne parviendra pas à délivrer une contribution suffisante.

En ce sens, les groupes ne peuvent pas être totalement autonomes, car ces situations doivent être détectées rapidement par un contrôle continu qui peut prendre plusieurs formes, et résolues, généralement par une personnes externe au groupe.

Dans ces situations problématiques, le temps est un facteur crucial. S'il faut 3 semaines ou plus pour détecter un dysfonctionnement, la résolution peut être compliquée, car le projet s'effectue dans des contraintes temporelles serrées, et un retard de plus de trois semaines dans la programmation ne peut en général pas être rattrapé, sauf extension du délai de rendu final, mais un tel délai supplémentaire n'a pas été accordé ces dernières années. De plus, si le groupe est totalement désynchronisé, une extension de délai ne résoudra pas le problème.

4.1.2 Spécification des objectifs

En termes d'objectifs d'encadrement, on distingue deux aspects :

1. l'encadrement technique,
2. l'encadrement organisationnel.

L'encadrement technique Sur le principe, on souhaite que les étudiants acquièrent une certaine autonomie dans le processus de programmation, on évite donc de donner une information trop « pré-mâchée». Pour ce qui est des spécificités du projet des trains,

on va leur fournir le maximum d'informations, car il n'existe pas d'autre source. En ce qui concerne la technique de programmation, on considère qu'ils ont déjà les acquis nécessaires et qu'il existe suffisamment de ressources de qualité, notamment sur le web, pour qu'ils puissent mener le projet à bien. Dans les faits, les enseignants ne refusent aucune question liée à la technique de programmation, car c'est l'occasion d'éclaircir facilement une zone d'ombre. Ceci se fait en général avec distance, c'est-à-dire que l'enseignent évite de dicter le code à écrire (ou pire de l'écrire lui-même), mais essaie de le faire écrire à l'étudiant par un cheminement raisonné. Cela prend plus de temps mais c'est au bénéfice d'une meilleure compréhension.

L'encadrement organisationnel Le but ici serait d'éviter les échecs dus à la mauvaise organisation du groupe. Comme stipulé plus tôt, la détection du problème doit se faire le plus tôt possible. Il conviendra donc de trouver un système qui permette d'analyser l'organisation du groupe de semaine en semaine, sans que cela ne soit trop intrusif pour les étudiants, ni trop lourd pour les enseignants.

Au delà de ça, on peut évidemment se poser la question :

> L'échec à cause de l'organisation du groupe doit-il être évité à tout prix par les enseignants ? Ou une mauvaise organisation est-elle, à l'inverse, de la responsabilité des étudiants ?

Sur ce point, nous répondons clairement que les variables de groupes ne doivent pas être la cause d'un échec dans ce cours de première année. Une réponse toute autre pourrait être apportée dans le cadre d'un cours de Master, où les étudiants ont plus d'expérience et sont censés avoir une responsabilité accrue quant à la gestion de leur groupe.

4.1.3 Conseils d'organisation

Plutôt que de donner des directives précises quant à l'organisation des groupes, on préfère leur laisser certaines libertés car cela augmente la motivation et il est certain qu'«apprendre par soi-même» est plus efficace sur le long terme que de suivre des directives. De toute manière, il nous serait difficile de prescrire un certain mode

d'organisation qui conviendrait à tous, car tous n'ont pas la même méthode de travail, ni la même façon de collaborer.

Toutefois, on leur présente en tout début de semestre un petit document graphique (élaboré durant la formation Did@ctic) représentant les différentes manières possible de s'organiser efficacement, suivant la phase du projet. Le document est visible dans l'Annexe B.

La première page de ce document montre de façon simple les interactions qui doivent exister entre les membres du groupes et le projet. Chaque personne entretien une relation à double sens avec le projet : il donne une contribution, et il doit en retirer la compréhension de la contribution des autres membres. Les interactions entre les membres sont aussi représentées et affichent les aspects de la communication et de la collaboration, deux aspects plutôt évidents.

La seconde page explicite deux schémas de collaboration possibles afin d'accomplir de petite tâches, comme par exemple les tâches d'implémentation de la première phase. Soit le groupe attribue le travail à un des membres, soit il travaille ensemble. Les spécificités des deux modes sont explicitées dans les flèches du haut de la page, et l'accent est également mis sur la responsabilité de chaque membre de comprendre la contribution de l'autre.

La dernière page illustre deux modes possibles d'organisation en vue d'effectuer une tâche plus conséquente, comme écrire le rapport ou programmer une extension (ce document date de l'époque ou une seule extension par groupe était développée). Soit les étudiants travaillent ensemble, soit ils se répartissent les tâches. Là encore, on insiste sur la nécessité de s'auto-évaluer et de vérifier qu'on a bien compris ce que les autres on fait sur le projet.

N'ayant pas de feedback sur ce document en particulier, nous ne savons pas s'il est vraiment consulté par les étudiants. En tous cas, il semble utile de les sensibiliser aux différentes manières de travailler, ainsi que de dégager les spécificités de chaque méthode.

4.1.4 Analyse de l'implémentation

Le résultat d'un projet informatique peut facilement être constitué de plusieurs milliers de lignes de codes. Bien sûr, la structure modulaire rendue possible par l'émergence des langages orientées objet ([MPW06]) fait que ces lignes ne sont pas toutes écrites par les étudiants, mais sont plutôt un ensemble de blocs de code de différents auteurs. Toujours est-il que la quantité de code est bien là. Multipliez ce nombre de lignes par le nombre de groupes du projet et vous vous retrouvez avec un gros problème sur les bras : celui de ne pas pouvoir vérifier tout le code soumis par les étudiants à l'issue du projet. Autant casser tout de suite les illusions : personne ne peut raisonnablement le faire, ce qui pose le problème de l'évaluation, auquel nous essaierons de répondre dans le chapitre 4.3.

Comme réponse, l'idée d'analyser le code de chaque groupe de manière régulière dès le début du projet permet, d'une part, de détecter d'éventuelles lacunes assez tôt et, d'autre part, de se familiariser avec le projet de chaque groupe en vue de l'évaluation finale. Pour chaque tâche de la première phase du projet, les enseignants vont parcourir le code de chaque groupe sur le serveur et donner des commentaires via Moodle. Prenons un exemple :

— La tâche de la semaine consiste, entre autres, en l'implémentation de la classe `Sector` (cf. fig. 4.1).

FIGURE 4.1 – Tâche d'implémentation dans Moodle

— Une des méthodes (non-encore implémentées de la classe `Sector` se présente comme suit :

```
/**
 * @return whether the sector is physically occupied or not
 */
public boolean isOccupied() {
    //TODO: implement
}
```

Les balises *TODO* indiquent où le code a besoin d'être complété et le commentaire quoi précède la méthode donne sa sémantique.

— Durant la semaine, le groupe va implémenter cette méthode et enverra sur le serveur, par exemple, la version suivante :

```
/**
 * @return whether the sector is physically occupied or not
 */
public boolean isOccupied() {
    //TODO: implementation done
    if(SX.instance().getStatusBit(address, bitpos) == 1)
        {
            return false;
        }
        return true;
}
```

— Lorsque la date limite d'implémentation (spécifiée par la tâche Moodle) sera dépassée, les enseignants vont aller lire sur le serveur les différentes contributions des groupes. Pour ce cas particulier, erroné, ils écriront un feedback sur la page de la tâche en question et attribueront la note *nearly OK* (fig. 4.2).

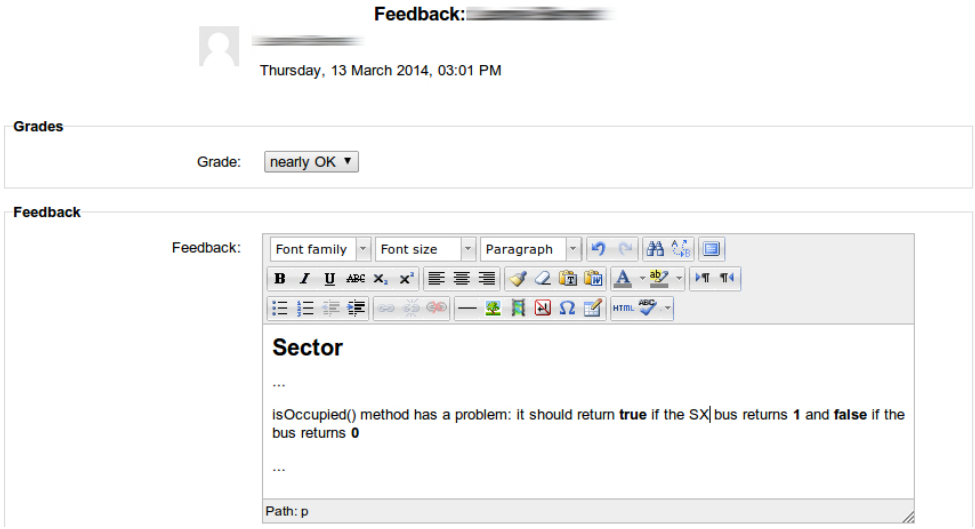


FIGURE 4.2 – Feedback d'une tâche sur Moodle

— Le groupe pourra ensuite retravailler sur le code et, s'il a été corrigé correctement, les enseignants pourront mettre à jour le commentaire et attribuer la note *OK*.

Dans le cas où l'implémentation contient de grosses erreurs, ou alors que celle-ci n'a pas encore été faite, on attribuera la note *not OK*[1].

1. Ce système de notation n'est pas inclus de manière standard dans Moodle mais a été ajouté pour satisfaire les besoins du cours

Ce processus de contrôle du code prend beaucoup de temps, car il nécessite de lire chaque contribution de chaque groupe et de détecter s'il y a une erreur, ce qui n'est pas toujours évident. Il faut noter que ce processus n'a lieu que pour les tâches d'implémentation de la première phase. En effet, il serait trop fastidieux d'analyser le code des implémentations plus développées, comme l'interface graphique ou l'extension.

Tests unitaires Afin d'offrir une deuxième ligne de défense contre les erreurs de programmation dans les classes de base, les enseignants on mis en place des *tests unitaires*. Ces modules ajoutés au projet ont pour unique but d'effectuer quelques tests sur les classes implémentées par les étudiants. Si l'implémentation est correcte, tous les tests rendront un résultat positif. Si l'implémentation est erronée, certains tests pourront signaler une erreur, mais ce n'est pas garanti (on ne peut pas tout tester). Ces tests sont très faciles d'utilisation et les groupes peuvent eux-mêmes les exécuter pour vérifier leur code, sans intervention des enseignants. Evidemment, ces tests permettent uniquement de détecter les erreurs et non de les corriger. L'exemple de la fig. 4.3 montre le résultat du test effectué sur le code erroné ci-dessus.

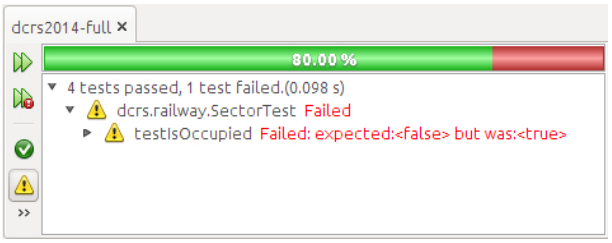


FIGURE 4.3 – Résultat de l'éxecution du test sur la classe *Sector*

4.1.5 Analyse de l'organisation du groupe

Ce point est certainement le plus délicat. Il est important de détecter les problèmes d'organisation et de communication au sein du groupe avant qu'ils ne deviennent insolvables. Mais comment y parvenir ?

Définitions de chefs de tâche Une approche que nous avions abordée était que, pour chaque tâche donnée, le groupe devait désigner un responsable. Ce dernier, soit

implémenterait la tâche lui-même, soit coordonnerait le travail et s'assurerait qu'il a été fait. Pour chacune de ces tâches, les groupe devait noter le nom du responsable dans Moodle. On atteindrait ainsi deux buts :

— améliorer l'organisation du groupe,

— s'assurer que tous les participants contribuent de manière significative au projet.

Cette approche fut rapidement abandonnée, car les groupes n'ont que peu suivi cette méthode de travailler. S'ils font la programmation ensemble, ils considèrent que ce système de responsable n'a pas lieu d'être et constitue juste une tracasserie administrative inutile. De plus, ce système est plutôt inefficace dans la détection et la prévention des problèmes d'organisation du groupe, car si celui-ci n'arrive pas à communiquer, il n'arrive pas non-plus à organiser la répartition des tâches de manière satisfaisante.

Auto-évaluations Partant du constat qu'il est difficile et fastidieux de juger de l'organisation d'un groupe par les seuls signaux qu'il donne via les travaux d'implémentation, l'idée est venue (par le biais de la formation Did@ctic, plus précisément Mme Nathalie Deschryver) d'organiser des auto-évaluations. Voici le principe :

Chaque semaine, à midi le jour du cours, chaque étudiant, individuellement et de manière confidentielle, répond à quelques questions sur Moodle :

— Qu'avez-vous implémenté/fait cette semaine ?

— Etes-vous satisfait de votre implémentation ?

— Avez-vous travaillé seul ou en groupe ?

— Avez-vous d'autres commentaires à propos de votre travail ou votre groupe ?

C'est un principe très peu utilisé en informatique. Lors de la mise-en-place, le scepticisme était de mise : Les étudiants vont-ils remplir ces auto-évaluations ? Vont-ils le faire honnêtement ? Peut-on ainsi prévenir des problèmes ?

Sur la question de la participation des étudiants à ces auto-évaluations, on a tout de suite insisté sur le caractère obligatoire de la chose. Qui plus est, le délai doit être respecté à la lettre. C'est même le seul délai qu'on leur impose de manière hebdomadaire. Il est obligatoire d'écrire quelque chose chaque semaine dans l'auto-évaluation,

même si l'étudiant n'a rien fait un niveau du projet.

Cette rigueur permet deux choses :

— repérer les participants inactifs [2],

— forcer les étudiants à faire une analyse de leur propre travail.

Ce deuxième point est important. Si un étudiant doit écrire trois semaines de suite qu'il n'a rien pu faire sur le projet pour telle ou telle raison, il y a fort à parier qu'il se rendra compte lui-même qu'il doit remédier au problème, sans intervention des enseignants. Le même cheminement peut être fait pour un étudiant qui a perdu le contact avec son groupe. Peut-être que de l'écrire sur Moodle lui fera prendre conscience qu'il faut remédier à la situation rapidement.

Sur le point de l'honnêteté, il est probable que certains embellissent un peu leur situation, mais nous n'avons jamais constaté de véritable mensonge. En réalité, il est là difficile d'écrire autre chose que la réalité car, admettons qu'un étudiant écrive : «J'ai implémenté la classe `Sector`», il est très facile pour les enseignants de vérifier si c'est bien le cas. Il y a donc peu de chance qu'il écrive quelque chose de totalement erroné. Quand bien même ce serait le cas, ça n'aurait pas grande importance, car ce n'est pas de cette manière que nous mesurons la contribution de chaque étudiant, mais en allant effectivement voir leurs contributions sur le serveur.

Avec ces informations en main, nous pouvons détecter d'éventuels problèmes au sein des groupes, sans avoir à les espionner, et ainsi prendre les mesures adéquates.

4.1.6 Périodes d'appui

Il y a deux moments qui sont spécifiquement dédiés à l'aide aux étudiants. Le temps en classe et le temps de laboratoire.

Aide en classe A l'issue de chaque cours, c-à-d durant les 7 premières semaines, les enseignants restent à disposition des étudiants qui, munis de leur ordinateur portable, peuvent poser des questions en relation avec l'implémentation de leur projet

2. Il arrive régulièrement que des étudiants soient inscrits au projet, et même dans un groupe, mais ne participent pas. Ce sont des participants «fantômes» qui peuvent perturber l'organisation du groupe en ne donnant que très peu de nouvelles à leurs collègues.

Temps de laboratoire Si les groupes viennent au laboratoire, c'est pour tester leur projet de manière concrète, mais c'est aussi pour obtenir l'aide des enseignants (il y en a toujours au moins un au laboratoire) pour leur poser des questions ou les aider à résoudre des problèmes d'implémentation.

Disponibilité hors-cadre des enseignants En dehors de ces temps d'appui, les étudiants peuvent encore contacter les enseignants de deux manières : soit par courrier électronique, soit en se rendant au bureau de l'un d'eux. Il n'y a pas d'horaire d'assistanat, donc théoriquement les étudiants peuvent venir à tout moment de la journée. Par e-mail, les enseignants répondent en général assez rapidement (dans la journée) aux questions.

4.2 Contrôle continu

Dans un projet, comme dans un cours traditionnel, on peut se demander si on veut mettre en place un contrôle continu. Le contrôle continu a plusieurs vocations :

— la stimulation du travail tout au long du semestre (et non seulement à la fin),
— la répartition de l'évaluation pour permettre d'alléger le poids de l'évaluation finale,
— la prise de conscience de l'étudiant vis-à-vis des exigences du cours.

Si on décide de mettre en place un contrôle continu, on doit ensuite s'intéresser à la forme et au mode d'évaluation. Pour ceci, le premier réflexe est d'énumérer les objectifs d'enseignement du cours, mais la formulation de ceux-ci peut parfois être théorique et arbitraire, alors qu'un projet est par essence de nature pragmatique et utilitaire. Pour nous, l'objectif d'enseignement est en rapport direct avec les différents aspects du projet (implémentation, collaboration, présentation). On va donc mettre un place un contrôle continu en accord avec ces objectifs.

La grille d'évaluation du cours (Annexe F) stipule que la présentation en classe est comptée dans la note finale. C'est la seule instance de contrôle continu officiellement comptabilisée. Néanmoins, il y a d'autres mesures de contrôle continu qui sont opérées, mais qui ne sont pas notées. Après tout, pourquoi un contrôle continu doit-il

obligatoirement être comptabilisé dans la note finale ? Nous reviendrons sur le détails de l'évaluation de façon plus concrète au chapitre 4.3, mais nous soulevons déjà un problème : comment passer du statut de tuteur à celui d'évaluateur ?

4.2.1 Dualité entre formation et évaluation

Dans de nombreux cours plus «classiques», cela ne pose pas de problème d'avoir un rôle d'enseignant ou de tuteur à un moment donné, puis de passer à un rôle d'évaluateur à un autre moment. Dans l'organisation actuelle du projet, les responsables du cours travaillent de manière proche avec les étudiants, et sont familiers avec leurs capacités, leur méthode de travail et le résultat de ce dernier. Dans ce cadre, comment prendre ponctuellement un rôle d'évaluateur ? En particulier, en supposant que les travaux sont notés simplement sur la qualité de l'implémentation, est-ce raisonnable de donner la même note à Anja et Caroline si, pour un résultat identique, Anja a travaillé de manière autonome et Caroline a écrit son code à grand renfort d'aide des assistants ? A première vue, on pourrait penser que «oui», car «seul le résultat compte». Mais sous un autre point de vue, comme les assistants ont passablement aidé Caroline à écrire son code, elle n'en est que partiellement l'auteure.

Un autre obstacle à la séparation entre formateur et évaluateur, déjà mentionné plus tôt, est celui qu'il n'est pas possible, vue la quantité de code, d'évaluer les projets en ne regardant que le résultat final. Il faut donc glaner des informations durant le semestre pour se faire une idée de la qualité du code, soit en lisant le code sur le serveur lors des évaluations hebdomadaires, soit lors du tutorat avec les étudiants.

Cette délicate situation dans laquelle les enseignants sont à la fois soutiens et juges inévitable. L'impartialité, dans quel cours que ce soit, n'est qu'une illusion. On ne peut pas penser qu'un échange avec un étudiant lors d'une phase de développement ne va pas subtilement modifier notre image inconsciente de cette personne, et ainsi influencer sur son évaluation. Le tout est d'en être pleinement conscient. Un grille d'évaluation précise peut aussi aider à cadrer le jugement et laisser moins de place à l'interprétation subjective.

Dans cette optique, à défaut de pouvoir clairement séparer les aspects formatifs et évaluatifs, on essaye de préciser au maximum les points sur lesquels une évaluation subjective ou qualitative est faite.

4.2.2 Contrôle du code

De manière régulière (chaque semaine lors de la première phase), les enseignant consultent le code fourni par les étudiants et donnent un feedback. Si ce code est incorrect, les étudiants doivent impérativement le corriger. Lorsque le code a été corrigé, nous considérons que la tâche est réussie et il n'y a aucune raison de diminuer la note finale sous prétexte que la première version était erronée.

Ceci dit, il existe un autre critère qui peut influer sur l'appréciation : la qualité. Même si un code fonctionne correctement et répond aux exigences, il peut être écrit :

— soit de façon inélégante[3].

— soit de façon trop extensive (on essaye de rester succinct),

— soit de manière dangereuse[4].

Ces analyses, mises ensemble, donnent une idée de la qualité générale du projet et permettent de dégager une évaluation qui comptera dans la note finale. Pour ne pas oublier le rôle formateur de contrôle continu, ces feedbacks sont communiqués aux étudiants via Moodle, y compris en ce qui concerne la qualité du code.

4.2.3 Analyse de la contribution personnelle

Ce point est certainement le plus délicat. Il est clair qu'une personne ayant réalisé seule un bon projet, mérite d'obtenir la validation du cours. A l'opposé, une personne n'ayant nullement contribué à un projet ne pourra qu'échouer à l'évaluation.

Entre ces deux extrêmes, toute la difficulté est de quantifier la contribution personnelle de chacun et de juger si elle est suffisante ou non. Le jugement sera l'objet du

3. L'élégance du code est un sujet central en programmation. Cela ne concerne pas seulement la recherche d'un idéal esthétique, mais aussi la facilité qu'auront d'autres à le comprendre. Un code écrit de façon inélégante sera difficilement utilisable par d'autres personnes. Dans un projet informatique, le code est un important moyen de communication. Il se doit donc d'être le plus clair possible

4. Lorsqu'on s'éloigne des standards du codage, on ouvre la porte à des possibilités d'erreurs qu'on ne voit pas immédiatement, mais qui peuvent intervenir beaucoup plus tard dans le projet. L'origine du problème est alors souvent difficile à trouver.

chapitre 4.3. Nous nous intéressons donc ici à la quantification.

Qu'entendons-nous par contribution personnelle ? On n'entend pas ici le nombre d'heures passées sur le projet. Ceci est bien égal et ce qui compte est le résultat.

Une manière de quantifier serait de compter le nombre de lignes de code de chacun et de calculer un barème. Dans le code, ce n'est pas compliqué, vu que l'on a accès au projet de chaque groupe et que le système subversion nous permet de voir qui a écrit quoi. En réalité, ce n'est pas si simple car toute les lignes ne se valent pas. Certaines sont faciles à écrire et d'autres peuvent nécessiter de longues recherches et réflexions. De plus, nous aimerions privilégier la brièveté du code. Une telle mesure serait donc contre-productive à ce niveau-là et inciterait les étudiants à faire du remplissage inutile. Une autre question vient s'ajouter : Que se passe-t-il si les étudiants on écrit leur code ensemble et non de façon séparée ? En supposant qu'ils n'aient utilisé qu'un seul de leur compte pour se connecter au serveur, le nom des autres utilisateurs n'apparaîtraient jamais, et dans ce cas on ne doit pas immédiatement conclure qu'ils n'ont pas contribué à l'implémentation.

Un autre outil d'analyse est la lecture des auto-évaluations. Il faut là aussi être prudents avec ce qu'on en fait. Les auto-évaluations sont issus de la bonne foi des étudiants, et ne doivent pas être utilisés directement dans le calcul d'un taux de contribution, car si le lien est trop direct, ces évaluations perdront en honnêteté et deviendront inutiles, et leur important rôle d'introspection sera également perdu.

Comme il paraît difficile de faire une quantification directe à partir des ces éléments, on choisira une approche basée sur l'exception dans laquelle une dévaluation de la note ne sera considérée que si les éléments à disposition montrent de manière univoque que la contribution a été insuffisante.

Si ce point a été placé dans la catégorie du contrôle continu, c'est qu'il a aussi un rôle informatif, et même préventif. Si l'aspect quantitatif de la contribution est un élément éliminatoire de l'évaluation, il faut que les étudiants aient une idée des exigences requises. Comme c'est compliqué de communiquer là-dessus de manière précise, et pour éviter les mauvaises surprises de fin de projet, on choisit de détecter les cas

problématiques en amont. L'analyse des éléments à la disposition des enseignants permettra de voir si un étudiant contribue de manière nulle ou insuffisante à un projet. En premier lieu, il s'agit d'organiser une discussion avec l'étudiant, car l'analyse peut être biaisée. Par exemple, s'il est constaté que l'étudiant fait très peu de contributions sur le serveur, celui-ci pourra avancer l'argument qu'il a toujours travaillé avec son groupe, sans utiliser son propre compte (ce qui est acceptable pour le première phase du projet), et cela sera facile à vérifier. Si par contre la contribution insuffisante (selon le jugement de l'enseignant) est avérée, alors l'étudiant sera clairement averti du risque d'échouer au projet si le retard n'est pas rattrapé. Ainsi, dans la quasi-totalité des cas, les étudiants qui ont finalement échoué à l'évaluation finale on été avertis suffisamment tôt de leur situation périlleuse.

4.2.4 Présentation en classe

Lors de la semaine 9, donc près de 4 semaines avant le rendu du projet, chaque groupe présente les particularités de son projet. Ceci fait l'objet d'une évaluation qui compte pour la note finale et permet de remettre les groupes sur la bonne voie si certains problèmes sont constatés. Accessoirement, cette session permet à tous les étudiants de voir ce qu'on fait les autres groupes. C'est là dernière fois qu'ils se rencontrent dans le cadre du cours.

4.2.5 Contrôle de connaissances

Jusqu'à l'été 2013, et afin de détecter assez tôt les cas problématiques d'étudiants qui auraient des lacunes trop importantes, un test était organisé en semaine 5. Ce questionnaire à choix multiple, qui se déroulait durant 15 minutes en début de cours, posait des questions de théorie sur le projet ainsi que des questions pratiques d'implémentation. Les personnes ayant compris la structure du projet et ayant été actives dans l'implémentation n'avaient pas de problème à remplir le questionnaire. En revanche, en raison de la variété des questions, les personnes ayant quelques lacunes arrivaient tout de même à faire un score assez élevé.

Ceci avait deux inconvénients :

1. ce test ne permettait pas de détecter les personnes avec des lacunes car il n'était pas assez sélectif,
2. il donnait un mauvais signal à certains étudiants, qui pouvaient faire un bon score sans maîtriser le projet de manière suffisante.

Bien entendu, ce test aurait pu être ré-adapté, mais en l'absence de bonnes proposition pour sa révision, il fut abandonné pour l'édition 2014.

4.3 Evaluation finale

Il y a deux moments ou l'évaluation est effective, c'est-à-dire qu'un score est calculé en vue de la note finale :

1. lors de la présentation en classe,
2. lors de la présentation finale du projet (au laboratoire).

Selon la grille d'évaluation de l'Annexe F, six aspects du projet sont pris en compte dans le calcul de la note :

— la présentation en classe
— l'implémentation de base
— l'interface graphique
— le rapport
— l'extension
— le facteur de contribution personnelle.

Certains de ces points évaluent le travail du groupe, d'autres le travail de l'individu. Avant de discuter d'éventuels barèmes, il est primordial de bien définir quel poids on donne à l'évaluation personnelle par rapport à celle du groupe.

Dans le cadre de la formation Did@ctic, l'évaluation du cours à fait l'objet de l'analyse qui est présentée dans l'Annexe C.

4.3.1 Equilibre entre évaluation du projet et évaluation personnelle

Dans la plupart des travaux de groupes, la même note est donnée à tous les membres, sauf si le projet comporte une composante individuelle. Dans bien des cas, cet élément personnel est le rapport. Les étudiants rendent donc chacun un rapport personnel, et le projet est noté en fonction de celui-ci. Cette solution est loin d'être idéale, car bien que le rapport soit une composante importante, on ne juge là que le rapport et non le travail en lui-même. Aussi, il est dangereux de noter un projet uniquement en lisant le rapport, car celui-ci a souvent tendance à être hypocrite [5], et ne rendent pas compte de la réalité de l'implémentation.

Jusqu'à l'été 2012, l'entier du projet était fait ensemble dans le groupe et il n'y avait pas de partie personnelle. Il y avait bien une extension à programmer, mais une seule par groupe. La différentiation des notes était alors difficile, car il fallait évaluer avec précision la contribution personnelle de chaque membre dans le projet commun. Depuis le printemps 2013, il est obligatoire que chaque étudiant développe une extension, à titre de travail individuel. Il est ainsi plus facile d'attribuer une note personnalisée à l'étudiant. Depuis le printemps 2014, cette extension compte même pour 40% de la note finale.

Fondamentalement, il s'agit ici d'un projet par groupes, on ne peut donc pas totalement séparer les évaluation des membres d'un même groupe. La seule alternative serait de faire de projets totalement personnels, mais on passerait à côté d'un des buts d'apprentissages, à savoir la gestion d'un projet en équipe. Comme on ne peut pas faire de séparation totale, il faut émettre quelques principes afin d'éviter les injustices :

1. Un étudiant ne doit pas échouer au projet à cause des autres membres de son groupe.
2. Un étudiant ne doit pas obtenir une note suffisante s'il a collaboré de façon insuffisante au projet.

Ces contraintes out débouché sur la grille d'évaluation de l'Annexe F. En quelques mots, sur un total de 50 points, 30 proviennent du travail de groupe, et 20 de l'extension

5. dans le sens qu'il délivre une critique résolument positive du projet, plutôt qu'objective

personnelle. Les principes présentés ci-dessus sont globalement respectés, car :

1. Si l'étudiant se trouvait dans un groupe défaillant où les autres membres n'ont rien fait, il parviendra facilement à faire 20 points ou plus pour la partie «groupe» plus 20 points pour la partie personnelle.
2. Si l'étudiant n'a que très peu contribué au projet, premièrement il obtiendra peu de points pour l'extension personnelle, et on sera en mesure de baisser le ratio de contribution personnelle (PC). Il n'a donc pas la possibilité d'obtenir une note suffisante simplement en s'appuyant sur le travail des autres membres du groupe.

Hormis ces cas extrêmes, la grille permet une graduation basée uniquement sur la qualité du projet, car la règle est que le facteur de contribution personnelle soit fixée à 100%, sauf exception notoire. Donc, dans la configuration générale où il n'y a rien à noter par rapport à une contribution insuffisante, ce critère n'entre pas dans le calcul de la note. On présente ci-après les différents points de l'évaluation avec un peu plus de précision.

4.3.2 Présentation en classe

C'est la première rencontre des groupes avec le processus d'évaluation. Le tout est organisé de manière plutôt légère. Chaque groupe dispose de 8 minutes pour présenter son projet devant la classe. C'est un temps très court qui ne permet pas de présenter le projet dans son intégralité, mais seulement de se concentrer sur les spécificités du projet en question par rapport aux autres. On présentera notamment l'interface graphique ainsi que les extensions. Il n'est pas nécessaire de lancer une exécution du projet car, d'une part, l'implémentation n'est pas terminée à cette date et, d'autre part, on se trouve dans la salle de cours et non dans le laboratoire. On ne peut donc guère plus montrer qu'une simulation du projet.

La brièveté force les groupes à présenter le projet de manière originale, parfois en présentant quelques diapositives, ou un petit film montrant le fonctionnement de leur implémentation. Il y a une grande variabilité dans la qualité des présentations, mais

jusqu'à ce jour elles ont toujours été notées de manière généreuse, à moins qu'il manque des aspects fondamentaux.

Du point de vue de la notation, la présentation compte pour 4 points, dont 2 pour le contenu et 2 pour la forme.

4.3.3 Implémentation de base

On s'intéresse maintenant aux aspects qui sont évalués lors de la phase finale du projet. Comme expliqué dans le chapitre 4.2, les enseignants se familiarisent durant le semestre avec le projet des étudiants. Mais comme il n'est pas possible de faire une analyse du code de chaque groupe (et de chaque étudiant, pour l'extension) en fin de projet en raison de la quantité de code, l'évaluation se fait sur la base de la présentation finale. Cette démonstration dure en principe 10 minutes. Durant ce temps, les groupes exécutent un test du projet et en montrent les fonctionnalités. Si les enseignants considèrent que le test n'est pas complet, ils demandent quelques opérations spécifiques.

Dix points sont consacrés à cet aspect. Si les trains circulent correctement le long d'une ligne, c'est que tous les éléments de base du projet, ainsi que la plupart des observeurs ont été implémentés correctement. De plus, 2 points sont attribués si les trains peuvent circuler le long de YBlocks [6] ; c'est le signe que les observeurs restants sont implémentés correctement, et que les étudiants ont bien compris les concepts de calcul de route, car on doit mettre le circuit dans une configuration particulière pour forcer l'utilisation d'un YBlock.

Sauf exception, chaque groupe reçoit les 8 points de la circulation de ligne, car on vérifie au préalable que cet aspect fonctionne sur tous les projets. Les 2 points restants dépendent de la faculté des étudiants à générer ces YBlocks, mais dépend aussi de la qualité des extensions, car si une des extensions a été incorrectement programmée, elle peut perturber la circulation sur les YBlocks.

6. Ce sont des éléments spécifiques qui permettent au train de changer de direction.

4.3.4 Interface graphique

Chaque groupe doit présenter une interface graphique[7] qui soit pleinement fonctionnelle et élégante. On donne 3 points pour la conception et 3 autres points pour le bon fonctionnement. Seules les fonctions de bases obligatoires (contrôle manuel, lancement des lignes, monitoring du circuit) sont évaluées ici, même si l'interface graphique a été étendue et améliorée par un des membres du groupe, dans le cadre de son extension.

Le code de l'interface n'est pas inspecté de façon systématique par les enseignants. Seul le bon fonctionnement est observé dans le cadre de la présentation finale.

4.3.5 Extension

Afin de donner une composante plus personnelle à ces travaux de groupes, chaque étudiant doit imaginer et implémenter une extension au projet. Pour cette partie, 20 points sont attribués, dont 10 pour le concept (ingéniosité, pertinence, ...) et 10 pour le bon fonctionnement de l'implémentation. Là encore, il est difficile d'analyser le code de chaque extension, on se borne dans la plupart des cas à observer le fonctionnement, ce qui donne déjà une bonne idée de l'implémentation sous-jacente. Lors de la présentation finale, les enseignants n'hésiteront pas à demander des tests supplémentaires pour vérifier le bon fonctionnement dans toutes les situations. En cas de plus grand doute, ils pourront toujours analyser le code ultérieurement.

4.3.6 Rapport

Le travail écrit doit être réalisé en LaTeX [Mit+04] en suivant les recommandations données au cours. Chaque groupe rend un seul rapport, mais il doit être spécifié qui a écrit quelle partie. Un template LaTeX est mis à disposition et les groupes sont libres de l'utiliser ou non, donc la présentation visuelle n'est pas un critère fondamental, car elle est globalement identique dans tous les rapports. 4 points sont attribués à la structure du document et 6 points au contenu (exactitude, pertinence, illustrations, ...).

7. interface utilisateur qui permet de contrôler le système

4.3.7 Facteur de contribution personnelle

L'extension n'est pas le seul point où les étudiants sont évalués de manière individuelle. Le facteur de contribution personnelle est un levier qui permet d'ajuster pour chaque personne le total des points du groupe (implémentation de base, interface graphique, rapport). La valeur «normale» est 100% mais celle-ci peut baisser si l'on constate que la contribution de l'étudiant est minime dans le projet.

Le calcul de ce facteur est passablement arbitraire, car il s'appuie sur des aspects qui sont difficilement quantifiables. Pour les cas détectés durant le semestre comme étant problématiques, on va analyser leur contributions sur le serveur, l'impression qui se dégage des auto-évaluations, et le résultat de discussions avec la personne et les autres membres du groupe. Viennent s'ajouter à ces données les réponses fournies aux quelques questions individuelles [8] posées lors de la présentation finale.

Etant donnée sa subjectivité, ce facteur n'est que peu ajusté dans les faits. Si on considère que l'étudiant n'a pas du tout participé à la première phase du projet, on lui attribuera généralement un facteur de 50% et non de 0%. La marge de sécurité est donc grande en faveur de l'étudiant.

Dans certains cas plus subtils, on peut donner un ratio de 90% ou 80%, mais dans ces cas les enseignants discutent avec les étudiants et chacun expose ses arguments. Le ratio peut ensuite être revu s'il s'avère que la dévaluation n'était pas justifiée. Les justifications sont souvent difficiles à argumenter, c'est pourquoi il convient de bien expliquer le principe de ce calcul de la note, en exposant par exemple la grille d'évaluation en classe. Pour ne pas se retrouver coincé dans son propre formalisme, il est parfois plus sage de montrer et d'expliquer la grille d'évaluation et les principes de calcul de la note, plutôt que de distribuer ces grilles entre les mains des étudiants, car il est ensuite délicat d'y apporter des modifications ultérieures.

8. Ces questions de contrôle sont moins sur la théorie que sur l'implémentation. Par ce procédé, on veut évaluer si l'étudiant a vraiment participé à l'élaboration du projet, et ne se contente pas de réciter des concepts théoriques appris au cours.

4.3.8 Rendu de la note

Jusqu'à l'été 2013, la note n'était qu'indicative, car dans le système de comptabilisation du plan d'étude, seule la mention « réussi» ou « échoué» est inscrite. La note calculée était donc communiquée aux étudiants via courrier électronique (via Moodle, en fait). Depuis l'édition 2014, la note sera communiquée via le système de gestion de l'enseignement. Comme cela prend un certain délai, on proposera de communiquer la note de manière personnelle à la demande.

Dans la mesure du possible, si un échec est déjà avéré au moment de la présentation finale, celui-ci sera communiqué immédiatement. Pour quelques cas limites, on proposera à l'étudiant de refaire sa démonstration quelques jours plus tard, car il peut arriver que la nervosité l'emporte et que l'étudiant perde ses moyens. La deuxième présentation sera l'occasion de pouvoir répondre aux questions de contrôle dans une atmosphère plus calme, et souvent avec une meilleure préparation.

Chapitre 5

Conclusion

Les projets sont des passage obligés et souvent très appréciés d'un cursus d'informatique. Faire ces travaux en groupes introduit de nombreux aspects qui sont une plus-value pour la formation de l'étudiant. Pour les enseignants en revanche, c'est un casse-tête continu, notamment au niveau de l'évaluation.

Nous avons pu montrer, au travers d'une analyse complète d'un cours, quelles sont les difficultés d'un tel dispositif d'enseignement. Dans un rôle strictement évaluatif, les projets en groupes ne sont pas la panacée et il serait dangereux de rendre ces unités d'enseignement trop sélectives. Il vaut mieux rabattre ce rôle de sélection sur des cours de programmation propre, car les buts d'apprentissage des projets (gestion du temps, travail en collaboration, ...) sont difficilement quantifiables.

Mais il serait faux de supprimer ces projet du cursus, ou même de les rendre individuels, car on perdrait des composantes importantes de savoir-faire. On propose donc dans ce document quelques pistes qui permettent d'améliorer l'encadrement et de rendre le processus d'évaluation le plus juste possible, sans charge de travail administrative excessive pour les étudiants comme pour les enseignants, et sans diminuer la motivation.

Bibliographie

[Apa14] APACHE. *Apache Subversion*. 2014. URL : http://subversion.apache.org/.

[AR02] Edurne AGUIRRE et Benoît RAUCENT. « L'Apprentissage par projet... Vous avez dit projet? Non, par projet! » Université Catholique de Louvain, Belgique, 2002.

[Fre+04] Elisabeth FREEMAN et al. *Head First Design Patterns*. O' Reilly & Associates, Inc., 2004. ISBN : 0596007124.

[Mit+04] Frank MITTELBACH et al. *The LaTeX Companion*. 2e éd. Addison-Wesley, avr. 2004.

[Moo14] MOODLE. *Moodle (Unifr)*. 2014. URL : http://moodle2.unifr.ch/.

[MPW06] Brett D. MCLAUGHLIN, Gary POLLICE et Dave WEST. *Head First Object-Oriented Analysis and Design: A Brain Friendly Guide to OOA&D (Head First)*. O'Reilly Media, Inc., 2006. ISBN : 0596008678.

[Ora14] ORACLE. *Oracle and Java | Technologies | Oracle*. 2014. URL : http://www.oracle.com/.

Annexe A

Scénario Pédagogique

DEVELOPPEMENT D'UN SCENARIO PEDAGOGIQUE

Joel Allred, joel.allred@unifr.ch

Description

Nom de l'activité : Projet de BSc (2ème semestre) en Informatique: Contrôle de Processus

Description synthétique :

Le projet est basé sur un modèle réduit de train électrique, piloté par un ordinateur. Les étudiant-e-s, par groupes de 2 ou 3, doivent développer un logiciel de contrôle des trains, comprenant des contrôles manuels pour commander les trains individuellement, et un système de lignes avec recherche de route libre, gestion de la vitesse et de la direction des trains, des aiguillages, des signaux lumineux, etc. Ils doivent également développer une interface graphique.

Le projet se fait entièrement en Java. Chaque groupe reçoit un squelette du projet, avec quelques méthodes déjà implémentées. Ils doivent, au fil des semaines, implémenter les méthodes de base, et progressivement les fonctions de plus haut niveau.

Inscription dans le contexte institutionnel (programme ou plan d'études ; lien avec le NQF) : Projet obligatoire de deuxième semestre, BSc en informatique. Permet de développer l'autonomie des étudiants, ainsi que la faculté de mener à bien un projet en groupe.

ECTS prévus pour l'activité : 5

Durée estimée pour l'apprenant-e (heures/semaine)	face à face : 3-4h / sem. pour les cours, et l'utilisation du labo.	en ligne : X	travail personnel : 3-8h / sem. pour faire les devoirs, ainsi que pour développer le projet par groupe. (temps hebdomadaire variable en fonction de l'étape du projet).
Personnes ressources (nombre)	enseignant-e-s : 3 enseignants (un professeur, un assistant et un sous-assistant qui se partagent les tâches d'enseignement. L'assistant et le sous-assistant s'occupent également du suivi des projets.		

Pré-requis

Comme le cours s'inscrit dans le cursus obligatoire d'informatique, il n'y a pas de prérequis formel. Après le premier semestre d'informatique, les étudiant-e-s connaissent le langage Java, mais n'ont que peu d'expérience pratique. Ils ont aussi peu d'expérience dans la gestion d'un projet, en groupe. Il n'y a pas d'attente en particulier vis-à-vis des étudiants.

Compétence(s) principale(s)

A l'issue du projet, les étudiant-e-s auront acquis des **compétences de programmation**, de **résolution de problèmes** concrets, de **gestion d'un projet** dans son ensemble (temps, ressources) et des compétences sociales afin de mener à bien un **travail de groupe**.

Objectifs

savoir-refaire / savoir-redire	Dans les premiers temps les étudiant-e-s doivent se familiariser avec les concepts et la terminologie du projet (en relation avec les chemins de fer). Mais ceci n'est pas un objectif du cours, il s'agit simplement de conditions pour pouvoir bien comprendre ce projet en particulier. Il n'y a donc, dans ce cas, **pas d'objectifs** (à proprement parler) au niveau du savoir-refaire / -redire.
savoir-faire convergents	Un des buts principaux est la **familiarisation avec un langage de programmation** (ici Java). Celle-ci peut se situer au niveau du savoir-faire convergent, ainsi que divergent, car il y a certains aspects, notamment l'**aspect syntactique**, qui suivent des « règles d'écriture ». Il n'y a qu'un nombre limité de « bonnes façons » de faire. D'autres savoir-faire sont demandés, en particulier l'utilisation d'un **système de contrôle des versions** (svn) et de programmation collaborative. L'aspect technique de la création d'une **interface graphique** relève aussi d'un savoir-faire convergent.
savoir-faire divergents	Comme précité, la **programmation** appelle aussi un savoir-faire divergent. Outre l'aspect purement technique de la syntaxe, il y a un côté **sémantique**, où là il y a souvent une infinité de manières de programmer une certaine fonction de manière correcte. A un plus haut niveau, les étudiant-e-s doivent **résoudre** certains **problèmes** spécifiques au projet. Par exemple, ils doivent trouver une manière de faire circuler les trains en évitant les collisions. Il y a de nombreuses manières de le faire. La conception de l'**interface graphique**, qui fait intervenir de nombreux choix de conception, de fonctionnalité et d'ergonomie, fait également partie des savoir-faire divergents développés.
savoir-être / savoir-devenir	Dans les objectifs plus abstraits du **savoir-être** et du savoir-devenir, il y a la **faculté de pouvoir travailler en groupe**. Il ne s'agit pas là que d'un côté pratique du projet, le but étant très clairement de développer les facultés de communication des individus, ainsi que de gérer la répartition des différentes tâches de manière autonome dans chaque groupe. Cette faculté de **comprendre les attentes, forces et faiblesses** des autres membres du groupe entre dans la catégorie du savoir-être.

	Dans le cadre de la notion plus subtile de **savoir-devenir**, qui sous-tend une idée de progression de l'étudiant-e, on peut relever la volonté que les étudiant-e-s se rendent compte que le réalisation finale du projet passe obligatoirement par une **phase d'apprentissage** des notions, suivie d'une **familiarisation avec le language**, le système, les contraintes, etc.

	Prise en compte de l'apprenant-e **(dans la préparation, la réalisation, l'intégration et le réinvestissement de la situation d'apprentissage)**
De ses caractéristiques individuelles ▪ ses projets, ses buts ▪ ses prérequis ▪ ses conceptions de l'apprentissage ▪ son évaluation de soi ▪ son attitude par rapport à la formation	Ce cours étant inscrit au programme obligatoire du BSc, on peut considérer que la motivation principale des étudiant-e-s de ce cours est simplement de mener à bien le cours de leurs études. Même si les étudiant-e-s proviennent d'horizons très divers, on considère, dans les premières semaines, que leurs compétences en programmation sont assez limitées. Le cours de base se donne dans ce sens-là. Les compétances particulières de chaque étudiant-e (imagination, capacités de programmation, habileté dans le design d'interface, compréhension des problèmes relatifs au trafic dans les chemins de fer,...) sont sollicitées plus tard, avec la possibilité de développer des extensions (au moins une) à leur projet. Ces extensions peuvent être très diverses, et peuvent aller d'un simple ajout d'une fonctionnalité dans l'interface graphique, à une refonte complète des principes de circulation des trains, en passant par des systèmes de détections d'obstacles,...
De sa motivation Comment susciter et maintenir sa participation ?	Ce projet, un des plus appliqués de leur cursus, se veut être une source de motivation en soi. La possibilité de voir ses efforts de programmation transformés en réel mouvement de trains électriques est souvent très gratifiante. La possibilité de visiter le laboratoire toute les semaines pour tester le projet en est aussi une composante importante. Le cours en classe se veut très visuel, avec des transparents très illustrés et incluant des composantes humoristiques. L'attention des étudiant-e-s est maintenue par des questions tout au long du cours, ainsi que par une phase « pratique » de réponse aux questions individuelles sur l'implémentation et de résolution de problèmes techniques à l'issue du cours.

Planification des activités d'apprentissages		approches
Activités en présence :	**Activités à distance :**	
Semaine 1 **Cours sur transparents :** – Introduction – Description des étapes du projet – Description des couches d'abstraction **Formation des groupes (2 ou 3 personnes)** **Visite du laboratoire, démonstration et explications des différents composants.**	Indiquées sur Moodle : - Requête d'un compte Linux - Installation de la plateforme Netbeans et du SVN - Installation du VPN	transmissif individualiste
Semaine 2 **Cours sur tranparents :** – Le bus SX et l'API DCRS – les diagrammes d'activités	Indiquées sur Moodle : - petit exercice de programmation impliquant la compréhension des opérations bit-à-bit. La réalisation de cet exercice nécessite Netbeans, installé la semaine précédente.	transmissif individualiste
Semaine 3 **Cours sur transparents :** – Présentation du système collaboratif Subversion – Description des tests unitiaires **Temps pour réponse aux questions individuelles et résolution de problèmes d'implémentation dans les groupes**	indiquées sur Moodle : Implémentation de deux classes du projet.	transmissif collaboratif (les travaux se font en groupe) individualiste (les tâches peuvent éventuellement être réparties à l'intérieur du groupe)

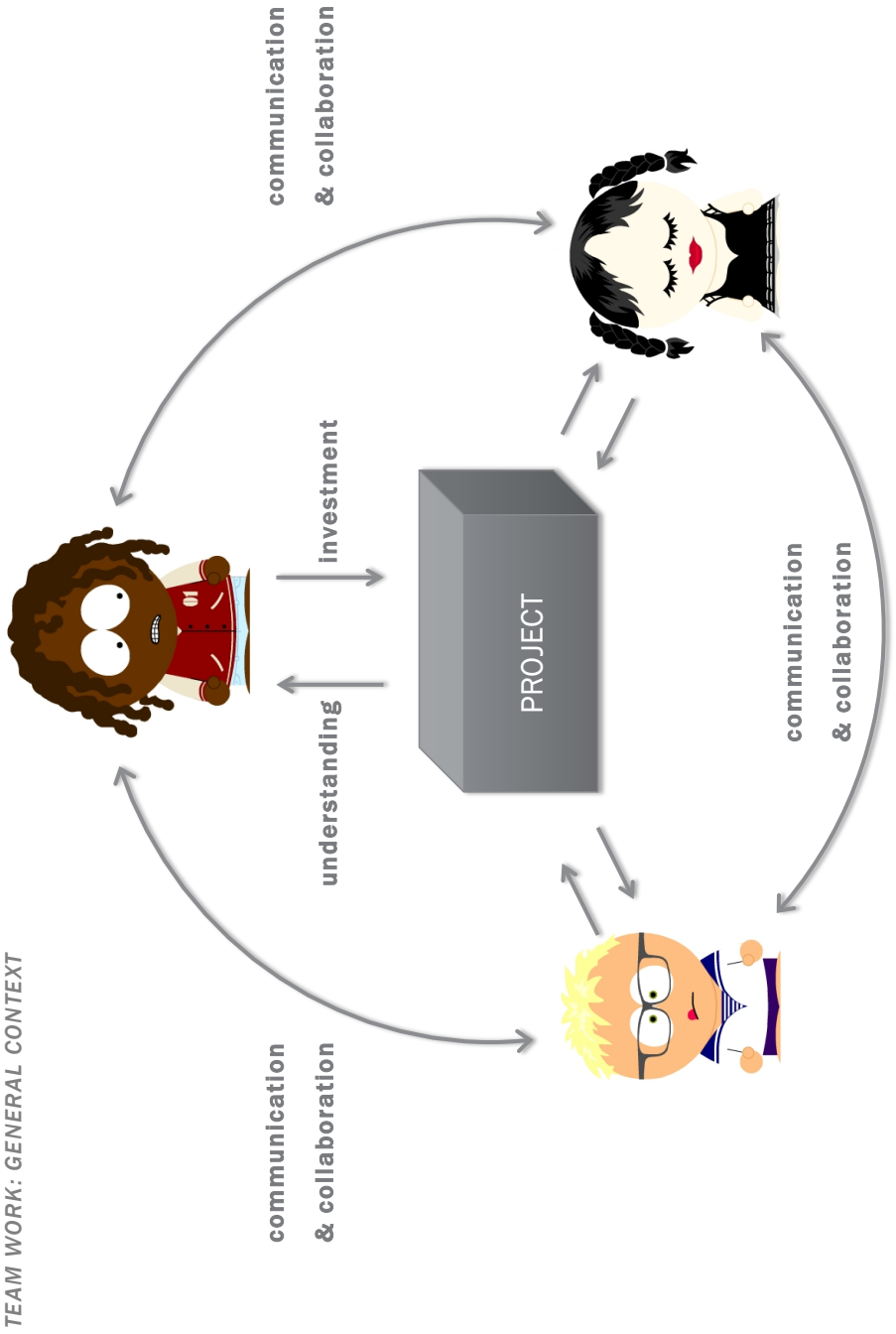
Semaine 4 **Cours sur transparents :** – composants (physiques et abstraits) du chemin de fer **Temps pour réponse aux questions individuelles et résolution de problèmes d'implémentation dans les groupes**	indiquées sur Moodle : – lecture – implémentation d'une classe – implémentation de l'interface graphique – réservation d'une plage horaire pour l'utilisation du laboratoire	idem pour la suite
Semaine 5 **Cours sur transparents :** – Description d'éléments du système – Description du système de routage **Temps pour réponse aux questions individuelles et résolution de problèmes d'implémentation dans les groupes** **Temps de laboratoire pour les groupes**	indiquées sur Moodle : – implémentation d'éléments du système – réservation de plages horaires pour l'utilisation du laboratoire	
Semaine 6 **Cours sur transparents :** – Présentation des observeurs – Précisions sur les prochains devoirs **Temps pour réponse aux questions individuelles et résolution de problèmes d'implémentation dans les groupes** **Temps de laboratoire pour les groupes**	indiquées sur Moodle : – implémentation d'éléments du système – réservation de plages horaires pour l'utilisation du laboratoire	
Semaine 7 **Cours sur transparents :** – Descriptions d'éléments du système – Présentation de l'extension que chaque groupe devra faire. Les groupes peuvent librement choisir une ou	Indiquées sur Moodle : – implémentation d'observeurs – réservation de plages horaires pour l'utilisation du laboratoire	

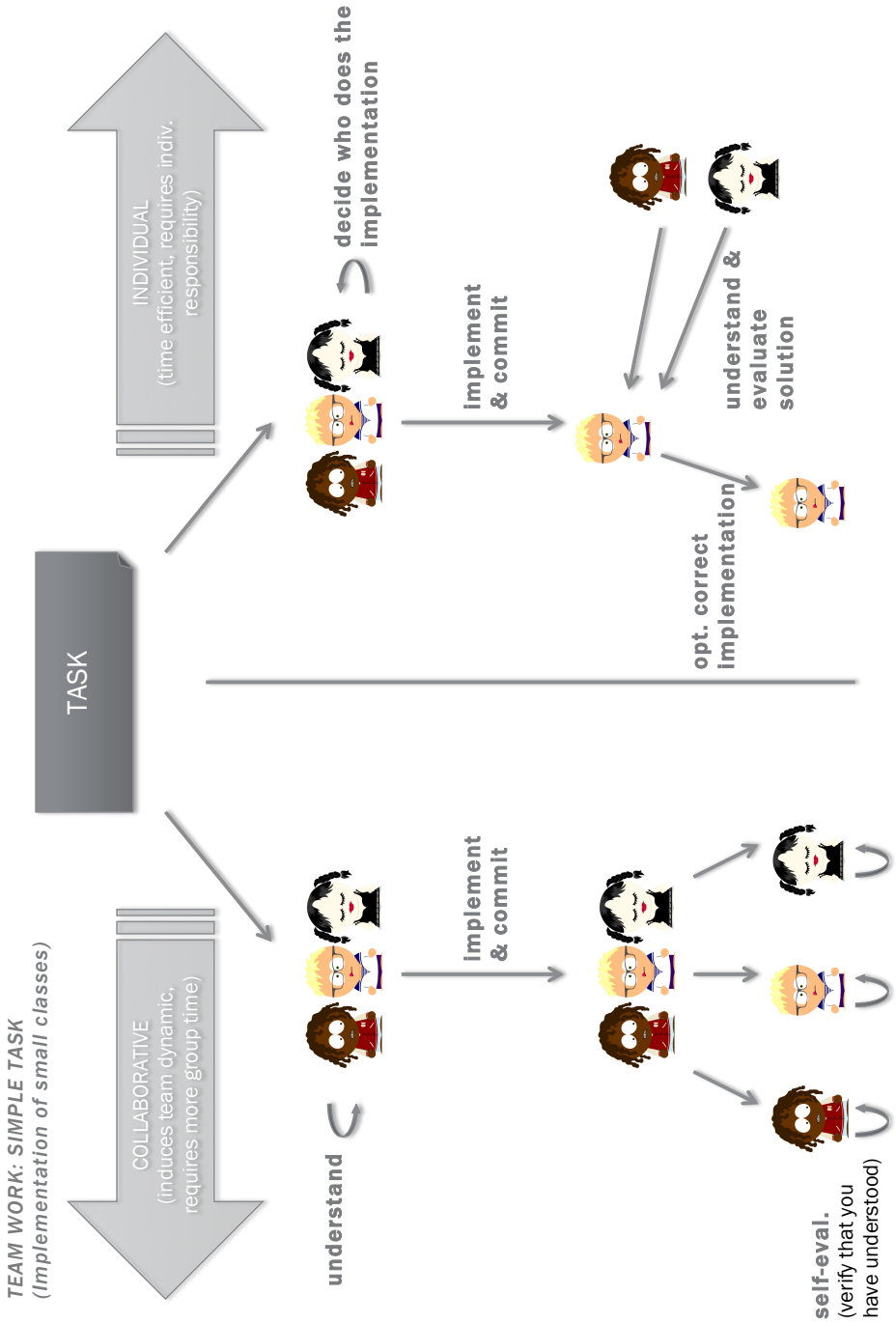
plusieurs extensions qu'ils veulent intégrer à leur projet. **Temps pour réponse aux questions individuelles et résolution de problèmes d'implémentation dans les groupes** **Temps de laboratoire pour les groupes**			
Semaine 8 **Cours sur transparents :** – Présentation de LaTeX, requis pour la mise en forme du rapport. **Temps pour réponse aux questions individuelles et résolution de problèmes d'implémentation dans les groupes** **Temps de laboratoire pour les groupes**	Indiquées sur Moodle : – réservation de plages horaires pour l'utilisation du laboratoire – Description du ou des extensions choisies par chaque groupe (à rendre via Moodle).		
Semaine 9 **Temps de laboratoire pour les groupes**	Indiquées sur Moodle : – réservation de plages horaires pour l'utilisation du laboratoire	collaboratif / individualiste	
Semaine 10 **Temps de laboratoire pour les groupes**	Indiquées sur Moodle : – réservation de plages horaires pour l'utilisation du laboratoire	collaboratif / individualiste	
Semaine 11 **Evaluation orale du cours, en classe** **Temps de laboratoire pour les groupes**	Indiquées sur Moodle : – Rendu du rapport	collaboratif / individualiste	
Semaine 12 **Présentation des projets par chaque groupe, à huis-clos avec les membres du groupe en question, le professeur, l'assistant et le sous-assistant.** Inclut des questions posées aux individuellement aux membres, pour vérifier que tout le monde a bien collaboré au projet de façon significative.		pas d'enseignement	

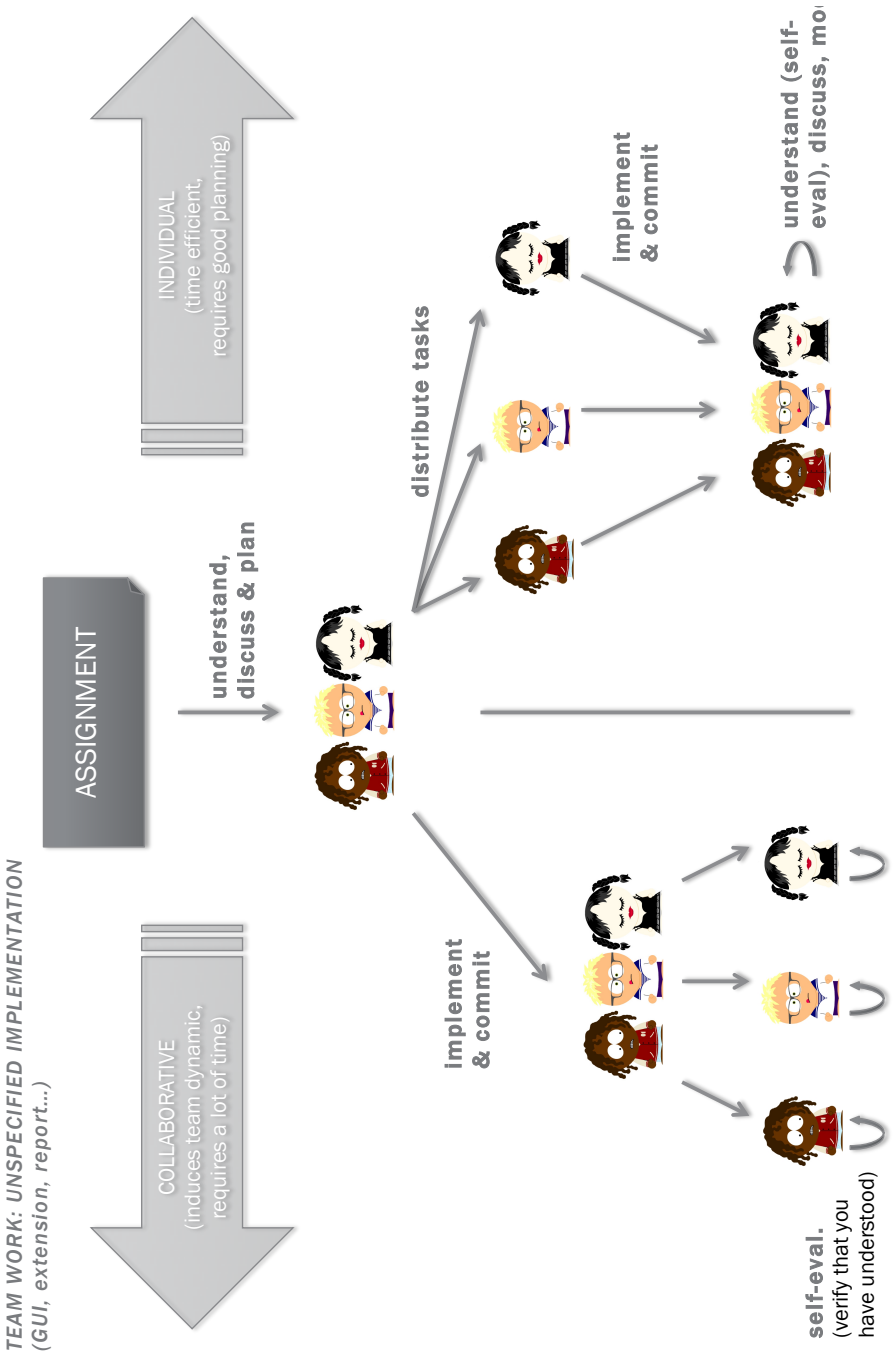
Vérification des caractéristiques d'un apprentissage en profondeur	Présent : o Absent : x	Commentaires (justification de ses choix)
Parcours négociés	X	Cours obligatoire de 1ère année BSc en informatique.
Unités de temps et de lieux diversifiées	O	La diffusion des concepts théoriques, ainsi que les réponses aux questions d'implémentation se fait en classe. La majeure partie du travail de programmation se fait à distance (librement, individuellement ou par groupe). Les tests sont effectués au laboratoire (à des plages horaires réservées par les étudiants).
Ressources en provenance des lieux de vie privés et professionnels	X	Il est difficile d'inclure des ressources externes au projet. Il est rare qu'un étudiant ait déjà travaillé sur un projet qui pourrait être intégré à celui-ci.
Évaluation	O	Beaucoup de feedback tout au long du projet, afin d'assurer l'avancement et la qualité du projet dans tous les groupes.
Tâche	O	Les tâches sont principalement des travaux de programmations. Etant de simples exercices de programmation bien définis au départ, les tâches deviennent plus intéressantes et personnelles au fil des semaines.
Cohérence objectifs – méthodes – évaluation	O	Le tout semble cohérent. Le projet est basé sur la réussite. Tous les outils sont donnés au cours, et les étudiant-e-s développent leur projet grâce à ces outils, et doivent parvenir à un résultat qui fonctionne au laboratoire. Les objectifs cités dans le scénario sont en adéquation avec l'organisation et l'évaluation du projet.
Collaboration	O	C'est un travail de groupe qui exige une collaboration entre les membres du groupe, ainsi qu'une certaine planification pour la répartition des différentes tâches.
Usage des TIC	O	Utilisation de Moodle pour la distribution des documents, pour la communication des tâches, et pour le feedback sur le travail hebdomadaire. Utilisation d'un SVN pour la collaboration dans l'implémentation.
Intègre des moments de régulation ou une recherche évaluation à propos du dispositif	X	Il n'y a que durant l'implémentation des tâches simples des premières semaines que les groupes doivent évoluer à la même vitesse. Si quelques groupes prennent du retard, les enseignant-e-s se focalisent sur ceux-ci et les aident à maintenir le rythme. Ensuite, le projet étant personnalisé, l'avancement des groupes diffère, et ceci sans conséquence. Certains auront un projet plus élaboré, d'autres plus simple, mais tous seront acceptés, pour peu qu'ils respectent les exigences minimales.

Annexe B

Schémas pour le travail de groupe


communication
& collaboration
investment
understanding
PROJECT
communication
& collaboration
communication
& collaboration
TEAM WORK: GENERAL CONTEXT


TEAM WORK: SIMPLE TASK
(Implementation of small classes)
COLLABORATIVE
(induces team dynamic,
requires more group time)
INDIVIDUAL
(time efficient, requires indiv.
responsibility)
TASK
understand
implement
& commit
decide who does the
implementation
implement
& commit
understand &
evaluate
solution
opt. correct
implementation
self-eval.
(verify that you
have understood)


TEAM WORK: UNSPECIFIED IMPLEMENTATION
(GUI, extension, report...)
ASSIGNMENT
INDIVIDUAL
(time efficient,
requires good planning)
COLLABORATIVE
(induces team dynamic,
requires a lot of time)
understand,
discuss & plan
distribute tasks
implement
& commit
implement
& commit
understand (self-
eval), discuss, mo
self-eval.
(verify that you
have understood)

Annexe C

Evaluation

DEVELOPPEMENT D'UN SCENARIO PEDAGOGIQUE : EVALUATION

Joel Allred, joel.allred@unifr.ch

Évaluation des apprentissages	
Type d'évaluation (formative, sommative)	**Tâches :** Evaluation hebdomadaire formative des tâches à effectuer sur le projet. **Rapport :** Evaluation sommative du rapport **Implémentation :** Evaluation sommative de l'implémentation finale du projet. **Personnelle :** Evaluation sommative de la contribution de chaque membre du groupe.
Fonctions (diagnostique, pronostique, certificative)	**Tâches :** diagnostique. Permet de mesurer l'avancement de chaque groupe dans le projet. Permet de détecter des retards dans l'implémentation, et des problèmes de compréhension. La surveillance continue de chaque projet permet aussi aux enseignants de connaître le projet de chaque groupe, ainsi que la contribution de chacun des membres, afin de pouvoir évaluer le rendu final en connaissance de cause **Rapport :** certificative. Permet de vérifier la compréhension des différents aspects du projet. **Implémentation :** certificative. Permet de vérifier que les objectifs du projet ont été atteints. **Personnelle :** certificative. Permet de vérifier que chaque membre du groupe a contribué de manière significative au projet.
Formes et outils (type de questions et d'échelle(s) d'évaluation)	**Tâches :** Chaque semaine (durant les premières semaines), les étudiant-e-s reçoivent des tâches d'implémentation à effectuer sur leur projet, avec une limite de temps. L'implémentation doit en continu être téléchargée sur le serveur, ce qui permet aux enseignants de contrôler l'évolution du projet, et de voir qui a téléchargé quoi. Pour chaque tâche, une note est attribuée pour le groupe (excellent, satisfaisant, insatisfaisant) via Moodle. Ces notes n'ont pas de valeur formelle pour l'évaluation finale, mais permet aux enseignants de mesurer la qualité de l'implémentation. **Rapport :** Les exigences concernant le rapport, ainsi qu'un modèle de mise en forme, sont données durant le cours. Il n'y a pas de note pour le rapport. Il entre en ligne de compte dans l'acceptation ou non du projet. **Implémentation :** L'implémentation en elle-même est évalué sous deux angles : premièrement via le contrôle continu effectué par les enseignants sur le serveur, ensuite via la présentation orale (+ démonstration) de chaque groupe. **Personnelle :** Deux méthodes d'évaluer les étudiant-e-s de manière personnalisée. La première est la surveillance continue du projet, qui permet de voir qui a écrit quoi. Si les enseignants se rendent compte qu'un-e étudiant-e ne soumet pas de code au serveur, une discussion a lieu avec le groupe pour en connaître la raison (il est tout à fait admissible qu'un groupe travaille en permanence ensemble, et qu'il n'y ait qu'une personne qui soumette son code au serveur). La deuxième méthode a lieu lors de la présention du projet. Des questions techniques sur les bases théoriques du projet, ainsi que sur l'implémentation sont posées individuellement à chaque membre du groupe. La validation ou non du projet final dépend du jugement des enseignants par rapport aux 4 aspect ci-dessus. Ceux-ci sont annoncés au début du projet.

Critères d'évaluation	**Tâches :** Fonctionnement correct de la méthode, élégance de la notation **Rapport :** Contenu (correspondant aux exigences), exactitude des explications, présentation. L'orthographe est corrigée, mais pas considérée, surtout si les étudiants n'écrivent pas dans leur langue maternelle. **Implémentation :** Fonctionnement correct, qualité du design, nombre et pertinance des fonctionnalités. **Personnelle :** Connaissances théoriques. Connaissance du projet propre.

Feedback aux étudiant-e-s (sous quelle forme ? à quel moment ?)	**Contrôle continu (tâches) :** Chaque semaine, un feedback est donné pour chaque tâche via Moodle. **Contrôle continu (implémentation)** : Lors des tests en laboratoire, les enseignants peuvent aider les groupes, répondre aux questions, et donner leur avis sur la qualité de l'implémentation. **Rapport :** Le rapport est rendu aux étudiant-e-s avec les corrections de l'enseignant. **Evaluation finale :** Un feedback oral est donné à l'issu de la présentation de projet. Une éventuelle décision d'échec est discutée entre les enseignants après la fin du rapport et est communiquée à l'étudiant par voie électronique.

Évaluation de l'enseignement	
Questionnaire d'évaluation (quel type de questions ? à quels moments ?)	Il n'y a pas de questionnaire d'évalutation prévu. Il est possible qu'on envisage un questionnaire du bureau d'évaluation, mais il n'existe pas d'évaluation spécifique au projet. Nous réfléchissons s'il est préférable de faire cette séance après les présentations, plutôt qu'avant, pour les étudiant-e-s osent plus dire les choses. Mais il sera probablement difficile de faire venir les étudiants en classe après la fin du projet, si c'est juste pour une discussion d'évaluation.
Séance d'évaluation (quelle tâche ? en groupe ? individuelle ? mise en commun ?)	La séance d'évaluation aura lieu le semaine avant les présentations. Il s'agira d'aborder plusieurs thèmes (qualité du cours, transparents, documentation, travail au laboratoire, assistance…) sous forme de discussion.
Résultats (feedback aux étudiant-e-s ? régulation ? quelle prise en compte ?)	Les différents commentaires seront recueillis par écrit et gardés pour future consultation, mais pas redistribués aux étudiant-e-s. Ces commentaires seront naturellement pris en compte dans l'organisation du cours au semestre suivant.

Vérification des caractéristiques d'un apprentissage en profondeur	Présent : o Absent : x	Commentaires (justification de ses choix)
Parcours négociés	X	Cours obligatoire de 1ère année BSc en informatique.
Unités de temps et de lieux diversifiées	O	La diffusion des concepts théoriques, ainsi que les réponses aux questions d'implémentation se fait en classe. La majeure partie du travail de programmation se fait à distance (librement, individuellement ou par groupe). Les tests sont effectués au laboratoire (à des plages horaires réservées par les étudiants).
Ressources en provenance des lieux de vie privés et professionnels	X	Il est difficile d'inclure des ressources externes au projet. Il est rare qu'un étudiant ait déjà travaillé sur un projet qui pourrait être intégré à celui-ci.
Évaluation	O	Beaucoup de feedback tout au long du projet, afin d'assurer l'avancement et la qualité du projet dans tous les groupes.
Tâche	O	Les tâches sont principalement des travaux de programmations. Etant de simples exercices de programmation bien définis au départ, les tâches deviennent plus intéressantes et personnelles au fil des semaines.
Cohérence objectifs – méthodes – évaluation	O	Le tout semble cohérent. Le projet est basé sur la réussite. Tous les outils sont donnés au cours, et les étudiant-e-s développent leur projet grâce à ces outils, et doivent parvenir à un résultat qui fonctionne au laboratoire. Les objectifs cités dans le scénario sont en adéquation avec l'organisation et l'évaluation du projet.
Collaboration	O	C'est un travail de groupe qui exige une collaboration entre les membres du groupe, ainsi qu'une certaine planification pour la répartition des différentes tâches.
Usage des TIC	O	Utilisation de Moodle pour la distribution des documents, pour la communication des tâches, et pour le feedback sur le travail hebdomadaire. Utilisation d'un SVN pour la collaboration dans l'implémentation.
Intègre des moments de régulation ou une recherche évaluation à propos du dispositif	X	Il n'y a que durant l'implémentation des tâches simples des premières semaines que les groupes doivent évoluer à la même vitesse. Si quelques groupes prennent du retard, les enseignant-e-s se focalisent sur ceux-ci et les aident à maintenir le rythme. Ensuite, le projet étant personnalisé, l'avancement des groupes diffère, et ceci sans conséquence. Certains auront un projet plus élaboré, d'autres plus simple, mais tous seront acceptés, pour peu qu'ils respectent les exigences minimales.

Annexe D

Programme du cours

Process control 2014

Planning of the semester

14 weeks

1. February 20th

Course : Introduction (Ulrich), System overview (Joel), SX Bus (Ulrich)

Tasks : install Netbeans and the VPN, Create groups

2. February 27th

Course : Team Work (Joel), Subversion (Joel), RailwayFactory (railway.xml) (Samuel)

Tasks : checkout, implement sector charging in RailwayFactory.

3. March 6th

Course : Locomotive (Ulrich), manual train control GUI (Cedric)

Tasks : complete Sector, Switch and Locomotive classes, test with unit tests, design a GUI that manually controls a train.

4. March 13th

Course : Blocks, Yblocks and routing (one presentation), GUI

Tasks : implement Block, SwitchPosition and Route classes, implement the manual train control GUI.

Lab time

5. March 20th

Course : observers I

Tasks : implement OnEnterLastSector, OnLeaveBlock, OnHalt classes, design complete GUI

6. March 27th

Course : observers II, extensions

continue implementing GUI

7. April 3rd

Course : LATEX, report, explanations of the parts which seem not well understood,

Tasks : continue implementation

8. April 10th

no course, lab time

Tasks : continue implementation

9. April 17th

Short presentations (8 minutes).

10. April 24th BANK HOLIDAY!

No course.

11. May 1st

No course, instead more lab time.

12. May 8th

No course, instead more lab time.

13. May 15th

Tasks : report submission on Monday 12th May 2014.

Presentations

14. May 22nd

Presentations

Annexe E

Scénario hybride : implémentation d'une classe Java

Titre du cours : Projet : Process Control

Enseignants : U. Ultes-Nitsche, M. Seuret, J. Allred

Objectifs d'apprentissage : Implémentation d'objets Java

	Activité 1 : Cours frontal	Activité 2 : Implémentation	Activité 3 Vérification	Activité 4 : Feedback	Activité 5 : Feedback personnel	Eval. d'appr.
En présence	**Intentions :** *Décrire les objets physiques à implémenter* **Médias :** *Powerpoint, photos, schémas, visite labo, projecteur* **Acteurs :** *Enseignants*			**Intentions :** *Feedback sur les implémentations, cours frontal* **Médias :** *projecteur* **Acteurs :**, *Enseignants*	**Intentions :** *Feedback sur les implémentations, par groupe, et réponses aux questions.* **Médias :** *ordinateurs des étudiants* **Acteurs :** *Etudiants, enseignants*	**Intentions :** *Présentation des projets.* **Médias :** *labo* **Acteurs :** *Etudiants, enseignants*
A distance		**Intentions :** *Implémentation des objets en groupe* **Médias :** *Moodle, SVN* **Acteurs :** *Etudiants en groupe*	**Intentions :** *Evaluation formative des implémentations* **Médias :** *Moodle, SVN* **Acteurs :** *Enseignants*			**Intentions :** *Documentation des projets.* **Médias :** *rapport, Moodle* **Acteurs :** *Etudiants*
Processus d'apprentissage	Transmissif	Pratique, Collaboratif, Exploratif Expérimental, Créatif		Transmissif	Collaboratif	

Annexe F

Grille d'Evaluation

Process Control 2014

Evaluation Grid

Name :	Team :

Presentation

Criteria	points
Structure	/2
Content	/2
Presentation total (P)	**/4**

Basic implementation

Criteria	points
Trains circulate along a line	/8
Trains take YBlocks	/2
Basic implementation total (BI)	**/10**

Graphical User Interface (basic functions)

Criteria	points
Graphical User Interface: Concept (manual control + lines)	/3
Graphical User Interface: Effectiveness	/3
GUI total (GUI)	**/6**

Report

Criteria	points
Structure	/4
Content	/6
Report total (R)	**/10**

Personal contribution (PC)
(based on analysis of personal work and questions at the final presentation)

Ratio is used to the group work total

Comments	decision
	%

Group work total (P + BI + GUI + R) * PC

Sum	points
	/30

Extension

Criteria	points
Description:	
Extension: Concept	/10
Extension: Implementation	/10
Extension total (BI)	**/20**

Points: /50	Mark:

Signatures: