Philippe Gref-Viau
1527456

RAPPORT DE PROJET INTÉGRATEUR

présenté

à

Olivier Gendreau

École Polytechnique de Montréal

1er septembre 2014

INTRODUCTION

Ce rapport aura pour but de décrire, d'expliquer et de documenter le processus de développement logiciel sur lequel moi-même, Philippe Gref-Viau, ai travaillé dans le but de produire une application dans le contexte de mon projet intégrateur de 4e année. Contrairement au cheminement habituel, ce projet c'est dans mon cas déroulé au terme d'un échange étudiant au Queensland University of Technology à Brisbane en Australie. Dans ce contexte, je décrirai ici les méthodes utilisées afin de mener à bout ce projet, un survol de l'historique de ce dernier ainsi que les concepts de génie logiciel utilisés et acquis au fil du processus de développement.

Le tout commence par l'équipe. Se joignant à moi pour ce projet étaient des étudiants en technologies de l'information à mon université d'accueil. Une sélection libre nous a menée à former un groupe originellement composé de 5 individus: Corey T., John W., Mikhaïl S. et Lachlan L. et moi-même. Corey et Lachlan étudiaient en programmation, Mikhaïl en architecture des réseaux et John en développement de processus. Ensemble, nous avons exploré les différents projets disponibles. Nous avions originellement choisi un projet qui concernait le traitement de données GPS, mais nous avons changé après s'être rendu compte que le projet demandais beaucoup de travail théorique et très peu d'interaction avec un client dans le but de construire un produit logiciel, ce qui ne semblait pas être compatible avec mes objectifs de projet intégrateur. Finalement, nous avons décidé de prendre un projet basé sur l'analyse des attaques contre les algorithmes de recommandations existants. Bien que le principe derrière le projet soit expliqué plus en détails dans les artéfacts du projet, je décrirai ici brièvement la nature du problème à résoudre.

Les systèmes de recommandation possèdent 3 constituants principaux: des utilisateurs, des items ainsi que des valeurs de classement qui établissent des relations entre certains utilisateurs et certains items. Ce genre de système a pour but de faire des recommander certains items pour un utilisateur en particulier basé sur un critère de relation que ce dernier possède avec l'ensemble des autres utilisateurs du système. Plus la corrélation entre deux utilisateurs sera forte, plus ils s'influenceront entre eux au niveau de leur choix d'items recommandés. On retrouve les systèmes de recommandation sous plusieurs de nos jours. De bons exemples incluent des sites de classement de films tels que IMDB ou bien la section de produits recommandés sur des sites de vente en ligne tels que Amazon.

Il faut aussi savoir que ces systèmes sont vulnérables à différentes attaques. Ces attaques peuvent par exemple prendre la forme d'une influence intentionnelle, positive ou négative, de la note globale d'un item quant à son degré de recommandation pour l'ensemble des utilisateur. Il existe plusieurs façons d'effectuer ces attaques; typiquement, une grande quantité de faux "profils", c'est-à-dire des faux utilisateurs ayant noté des items clés spécifiques à la nature attaque, seront créés et insérés dans l'ensemble des profils existants, tentant ainsi d'influencer l'ensemble du système. Le projet que nous avons choisi avait comme objectif final l'élaboration d'une application système servant de plateforme à partir de laquelle les membres de l'équipe de recherche pourraient plus facilement tester ces attaques

sur des systèmes de recommandations et de pouvoir évaluer les résultats et statistiques générés par l'attaque.

C'est donc dans ce projet que mon équipe et moi nous sommes lancés, sous la supervision de notre client, le Professeur Yue Xu. Cette dernière était responsable de la recherche sur les systèmes de recommandations dans le département d'informatique de QUT. La première étape de notre travail était d'analyser le volet mathématique du problème. En effet, bien que les besoins de notre client pouvaient en majorité être traduis par des concepts logiciels, une grande partie du travail concernait la compréhension des algorithmes et des théories mathématiques derrière les systèmes de recommandation. Il était nécessaire de bien comprendre la complexité et la nature des différentes formules de recommandation et des attaques qui leurs étaient associées. En ce sens, j'ai su appliquer des concepts mathématiques appris au cours de mon programme d'ingénierie afin de mieux comprendre le problème et de choisir plus judicieusement les solutions potentielles. Étant donnée la nature du projet, ces concepts se rattachaient beaucoup plus au domaine de la statistique qu'aux autres cours mathématiques vus à la Polytechnique. Ceux-ci incluaient entre autres la mesure de corrélation par la méthode de Pearson, l'extrapolation statistique et à l'application du TCC (Théorème Central Limite). Je comptais aussi sur la collaboration de mon collègue Mikhaïl, qui possède une mineure en mathématique, avec lequel nous avons pu entre autre appliquer nos connaissances afin de gérer des cas d'exceptions lors de la transition des formules mathématiques en leur implémentation programmatique, un point sur lequel je parlerai plus en détail plus tard dans ce texte.

Est venu par la suite le choix des technologies à utiliser. Encore là, j'ai bien profité de mon expérience avec une multitude de différents langages afin de mieux choisir ce qui conviendrait le mieux à nos besoins et à celui de notre client. Notre focus original était beaucoup centré sur la rapidité inhérente du langage à effectuer les différents calculs liés aux algorithmes de recommandations et de génération d'ensembles de données d'attaque. Ainsi, le choix avait originellement été fait de faire le traitement grâce à Java, un compromis qui permettait de bénéficier de l'expérience des mes collègues dans ce langage en plus d'offrir un cadre d'application assez flexible pour répondre aux besoins fonctionnels du problème. Un langage de plus haut niveau permettait aussi d'utiliser des abstractions en ce qui attrait à l'entrée/sortie de données par l'entremise de librairies telles que Hibernate, lesquelles ont facilité de beaucoup l'interaction avec une base de donnée. Cependant, suite à de la rétroaction de la part de notre client au cours du cycle de développement, nous avons compris que l'interface utilisateur et l'automatisation du processus étaient aussi des parties essentielles du produit final. Ainsi, nous avons pris la décision d'effectuer une transition vers le langage C#, en accordance avec les besoins et désirs de notre client, afin de mieux profiter d'éléments tels qu'un système d'interface plus adapté et efficace (WPF) ou bien des fonctions lambda. Bien que, rétroactivement, ce changement semble avoir été plus que nécessaire, il a occasionné un gaspillage de temps non négligeable et aurait dû prendre place bien avant. Cependant, malgré ce délais, nous nous sommes relativement bien adaptés au changement, incluant l'utilisation d'un langage différent, le transfert de fonctionnalités ainsi que l'utilisation de librairies différentes. Ainsi, de façon , je juge que cet adaptation au changement était un de nos points positifs dans ce projet.

Une fois ces étapes préliminaires effectuées, nous avons commencé le processus d'analyse des besoins en collaboration avec notre superviseur, qui était par le fait même notre cliente. Pour ce faire, nous avons débuté en établissant de façon prioritaire les cas d'utilisations associés au projet. Il était bien important de comprendre avant tout quelles sortes d'utilisateurs allaient utiliser le système et surtout dans quel but afin de bien définir quelles sortes d'interactions allaient être priorisées pour ce produit. En ce sens, nos avons interviewé notre client et ciblant spécifiquement les besoins primaires derrière l'outil que nous allions créer. Ce faisant, nous avons compris qu'il s'agissait ici de pouvoir effectuer des calculs de recommandation (par l'entremise des algorithmes analysés précédemment), de pouvoir générer de façon paramétrable un ensemble de données selon un critère d'attaque spécifique (ou alternativement de se baser sur un ensemble existant), de fournir un moyen efficace de gérer l'entrée/sortie de données à partir de l'application, d'avoir un moyen efficace de présenter les données générées et finalement d'offrir une interface navigable et intuitive. Nous nous sommes basés sur ces 5 points pour bien définir nos critères de complétion, pour par la suite les décortiquer en plusieurs sous-besoins . Par exemple, notre cliente avait exprimée le désir de pouvoir aller chercher les profils d'attaque dans une base de donnée.

Par la suite, nous nous sommes lancé dans le processus d'élicitation des requis. Puisque les membres de mon équipe ne possédaient pas de structure spécifique quant à cette l'élaboration des requis, je leur ai proposé la structure hiérarchique et granulaire telle qu'explorée dans les projets de la Polytechnique. Bien que nous n'ayons éventuellement pas adopté ce modèle dès le début, nous avons quand même décomposé les besoins de notre client en une multitude de requis fonctionnels et non-fonctionnels. J'ai par la suite détaillé ces requis en leur donnant des indicateurs tels que les exceptions possibles, les chemins d'exécutions ainsi que les pré et post-conditions.

La prochaine étape majeure était de commencer à élaborer l'architecture logicielle du projet. À ce niveau, j'ai encore une fois utilisé une multitude de concepts utiles acquis lors de mes cours et de mes projets à la Polytechnique ainsi que lors de mes stages. Par exemple, un des requis non fonctionnels que nous avions établi en début de projet avec notre cliente était que, bien que nous avions pour but de livrer un projet complet à la fin de la session, l'application serait fort probablement reprise par d'autres étudiants dans le futur pour que ces derniers rajoutent des fonctionnalités supplémentaires. Plus précisément, les algorithmes de recommandations basés sur les items, c'est-à-dire la sorte que notre client avait suggéré que nous utilisions, pourraient être remplacés par des algorithmes basés sur des utilisateurs. Ces derniers utilisaient des équations mathématiques différentes qui, au lieu de calculer les items recommandés pour un utilisateur à partir des marques données par ce dernier comparativement à l'ensemble des autres utilisateurs, choisit plutôt de faire l'inverse et de calculer une liste des utilisateurs qui seront potentiellement les plus intéressés par un certain item, une méthode qui se révèle utile dans certains scénarios. Prenant cela en compte, il était de mon opinion que l'architecture logicielle devrait facilement supporter l'ajout de fonctionnalités correspondant à cette nouvelle méthode, tout en réutilisant au maximum les artéfacts et opérations communes aux deux approches. C'est donc dans cette optique que j'ai utilisé au maximum les principes de l'abstraction et du polymorphisme afin de m'assurer que les items et utilisateurs soient traités de façon presque interchangeable au niveau des

calculs respectifs leur étant associés. Le résultat final est une hiérarchie polymorphique dans laquelle il sera très facile d'intégrer de futurs types d'algorithmes de recommandation.

D'autres principes logiciels acquis grâce à mon expérience passée nous ont permit de faire des améliorations importantes au niveau de la performance de l'application. Par exemple, lors de l'implémentation de la partie de calcul intensif relié aux algorithmes de recommandations, la performance était sous le niveau acceptable que nous avions défini dans les requis non-fonctionnels. Le problème provenait du temps élevé associé aux requêtes dans la structure de donnée interne qui conservait les données. En ce sens, j'ai réglé le problème en implémentant un système de cache, qui assurait un temps de réponse adéquat en gardant des références sur les éléments les plus fréquemment demandés dans la table contenant les données importantes. J'ai aussi encouragé l'utilisation de traitement asynchrone et parallèle de données afin d'augmenter la rapidité générale de calcul. De plus, j'ai dû concevoir une architecture modulaire pour efficacement gérer les pages de l'application ainsi que les opérations de transition et de validation. Il fallait par exemple que les données soient rafraîchies et les entrées en conflit invalidées dans le reste de l'application si l'ensemble d'échantillon de données en entrée venait à changer. Bien que ces quelques exemples ne puissent pas montrer l'ensemble de la planification de conception utilisée dans ce projet, je juge que c'est malgré tout un bon aperçu des principaux concepts techniques

Le projet m'as aussi permis d'appliquer et de raffiner ma connaissance de certains patrons de conception, notamment le patron MVVC, que j'ai utilisé amplement pour bien décortiquer l'architecture de l'application et d'établir un lien efficace entre les couches de données et de présentation. J'ai aussi appliqué d'autres patrons variés, tel que le patron commande afin de gérer le passage d'information entre les pages de l'application tout en gardant un couplage faible, ou bien le patron proxy pour gérer efficacement l'entrée/sortie de données en lien avec différentes sources, parfois distantes. Comme les ensembles de données étaient particulièrement grands et que le traitement subséquent par les algorithmes de recommandation avait tendance à être assez coûteux en terme de temps de calcul et d'utilisation de ressources mémoires, j'ai aussi dû pour la première fois me plonger dans l'analyse d'optimisation. Pour ce faire, j'ai utilisé extensivement différents outils de profilage afin d'avoir une vue d'ensemble de la distribution du travail lors de l'exécution de mon programme

Comme je l'ai mentionné plus tôt, j'ai placé beaucoup d'importance sur la rétroaction de notre client. Cela venait du fait que, bien que nous comprenions bien le modèle théorique derrière les systèmes de recommandations, la seule personne qui pouvait vraiment valider notre travail, tout particulièrement dans les premières étapes du projet, était notre cliente. Nous avons donc établi un cycle de développement dans lequel nous devions livrer des prototypes à chaque 2 semaines, avec une rencontre pour faire part de notre progrès et clarifier les requis. En absence de prototype concret lors des premières semaines, nous avons plutôt décris notre processus d'investigation mathématique à la place. Ce système fonctionnait assez bien, permettant au client de voir le produit évoluer et nous donnait une fenêtre de 2 semaines pour bien implémenter la prochaine itération de prototype. Les rencontres d'équipes se faisaient approximativement 2 fois par semaine, traditionnellement juste après la rencontre avec notre client et une autre fois le Lundi de la semaine suivante (afin que nous puissions chacun travailler sur le projet pendant la fin de semaine). Nous avons aussi effectué quelques

rencontres au fil de la session plutôt centrées sur la gestion de conflit afin de régler des problèmes qui régnaient au sein de l'équipe.

Je peux affirmer avec certitude que de tous les concepts reliés au développement logiciel, ce projet m'a permit d'en apprendre le plus sur la gestion d'équipe. En effet, le plus gros défi associé à ce projet en particulier provenait principalement des différences existantes entres les méthodologies de travail propres aux différentes cultures universitaires. De plus, il était plus difficile de faire la planification de projet avec des gens avec lesquels je n'avais jamais travaillé auparavant. Cela a cependant offert des opportunités uniques d'en apprendre plus sur la dynamique d'équipe.

La première surprise que j'ai rencontré était un manque semblable d'habiletés techniques et de rigueur en gestion de projet apparent de la part de mes collègues Australiens. En effet, même si ces derniers étaient en 3e année universitaire, ils avaient tendance à ne pas avoir vu ou appliqué certains concepts clés du génie logiciel (du moins relativement à la Polytechnique). En ce sens, je me suis donc retrouvé en position de leadership car j'étais le seul étant préparé à appliquer une méthodologie de travail quelconque. C'est donc moi qui a poussé pour que les besoins du client soient documentés, que les requis élaborés soient structurés, que l'architecture développée soit flexible et que l'emphase soit mise sur des cycles de développement avec du prototypage et de la rétroaction client comme éléments principaux.

Je ne m'étais jamais retrouvé dans une position de leadership, alors il était difficile pour moi de savoir comment réagir au début. Un problème majeur était que certains de mes coéquipiers semblaient ne pas vouloir ou pouvoir compléter des tâches techniques que je jugeais simples. J'ai originellement réagi en appliquant une politique qui tendait plutôt vers l'accommodation. Ainsi, lorsque les tâches que j'avais assigné n'étaient pas ou mal faites, j'essayais de ne pas trop critiquer en essayant d'encourager mes coéquipiers à mettre un peu plus d'effort et en leur fournissant des objectifs plus précis. Je comptais surtout sur la connaissance des forces et des faiblesses de chacun afin de mieux rediriger la charge de travail du côté technique. Cela s'est avéré inefficace, et j'ai graduellement dû sévir afin d'assurer que le travail était complété de façon efficace. Cela a formé la base d'un conflit relationnel implicite basé sur l'accumulation graduelle d'un manque d'effort de la part de mes coéquipiers. Bien que je n'aie pas pu régler l'ensemble des problèmes d'équipe au fil du projet, j'ai cependant eu le sentiment que mes connaissances en gestion de conflit ont aidé à désamorcer l'évolution du conflit, et que la situation aurait pu devenir nocive si je n'avais pas su comment gérer ce genre de situation. J'ai finalement atteint un équilibre dans lequel j'ai cessé de pousser des tâches techniques sur les membres de l'équipe, choisissant plutôt de leur déléguer des tâches connexes. Cela a non seulement restauré quelque peu l'équilibre au sein de l'équipe, mais a aussi permit de se pencher sur des problèmes non techniques qui auraient pu affecter la qualité du résultat final du produit. Un bon exemple serait la présence d'ambiguïtés et d'erreurs dans les formules mathématiques de référence que nous utilisions, en plus de cas spécifiques pouvant affecter la version algorithmique de la formule et qui généraient des résultats erronés. Sans avoir assigné du temps et des ressources à cette assurance de qualité, nous aurions eu beaucoup plus de misère en fin de projet.

CONCLUSION

Pour résumer le tout, je pense qu'il est important d'admettre que ce projet n'a pas été parfait. Cependant, cette occasion m'as aussi permit d'en apprendre beaucoup sur le travail d'équipe en mettant dans une situation où rien ne m'était familier et que je devais en plus redoubler d'effort pour m'assurer de maintenir les standards de qualité et d'appliquer les artéfacts propres aux standards de la Polytechnique. Il ne fait aucun doute que ce projet m'as permit de mettre en pratique de nombreux concepts que j'ai appris au fil de mon baccalauréat. J'ai aussi pu apprendre à interagir avec les demandes d'un client et de gérer les ressources que je possédait afin de respecter les besoins de ce dernier. Finalement, j'ai pu me plonger dans un problème unique et intéressant qui m'as permit d'avoir une bonne expérience par rapport au développement logiciel sans trop d'encadrement scolaire, ce qui est selon moi un des buts principaux du projet intégrateur.

Le tableau suivant contient un résumé des différents concepts que j'ai appliqué lors de ce projet, ainsi que le degré d'efficacité que je juge avoir affecté à chacun d'entre eux, tant au niveau personnel qu'en tant qu'équipe.

Concepts	Bien appliqué	À améliorer	Pas assez présent
Investigation mathématique	X		
Gestion de la technologie		X	
Adaptation aux changements	X		
Analyse des besoins	X		
Élicitation des requis	X		
Architecture logicielle	X		
Patrons de conception/structures de données	X		
Méthodologie de travail		X	
Gestion de projet			X
Cycle de développement - rétroaction client	X		
Gestion d'équipe		X	
Gestion de conflits	X		