DURAND SAINT OMER

Sacha

ING 3 – SIGL



568 Broadway 11th Floor, New York, NY 10012

**De meilleures Applications**

Maitre de stage : German Bejarano

Réferent : Ludovic Ollagnier

Mémoire de fin d’études

Mai 2013 – juin 2013

Année scolaire 2012-2013

# Remerciements

Sommaire

Remerciements 3

Introduction 5

État des lieux du marché 5

Environnement de stage 5

Etude de Cas : Cooli 5

Un projet comme les autres 5

Imprévus extérieurs 5

Un projet compromis 5

Vers une meilleure Organisation 6

La gestion du code source 6

La sauvegarde du code source 6

Besoin d’un système décentralisé 6

Laissons faire les experts 7

Optimiser le travail en equipe 7

Controle de qualité 9

Iterations 10

Feedback 10

Vers un meilleur code 10

Syntaxe 10

Architecture 10

Tests 10

Conclusion 10

Annexes 10

References 10

# Introduction

## État des lieux du marché

## Environnement de stage

# Etude de Cas : Cooli

## Un projet comme les autres

## Imprévus extérieurs

## Un projet compromis

Nous allons, dans un premier temps, voir les pistes d’amélioration concernant l’organisation du projet.

# Vers une meilleure Organisation

L’organisation du projet informatique et une composante indispensable de son succès. De nombreuses décisions prises en amont vont nous permettre de limiter les risques.

## La gestion du code source

La valeur première du logiciel, c’est son code source. Pour toutes les sociétés dont le fonds de commerce repose sur la production de solutions logicielles, SSII, éditeur logiciel ou startup informatique, la perte de code source correspond presque directement à la perte de chiffre d’affaires.

Il est donc indispensable de protéger celui-ci de tous les risques potentiels, par exemple éviter à tout prix la perte code source.

Mais le code source étant au centre de l’activité de l’entreprise, on veux, outre la perte, aussi optimiser au maximum son accès et contrôler la qualité de celui-ci.

### La Gestion du code source

#### Besoin d’un système décentralisé

Nous voulons protéger notre fonds de commerce et donc limiter les riques au maximum**.** Afin d’éviter quelquouque perte, chaque ligne de code produite par les développeurs doit être précieusement sauvegardée via un système fiable.Réfléchissons. Quel est le système d’information le plus robuste à ce jour ? L’internet!En effet, l’internet fut inventé originellement pour résister à une attaque nucléaire!Internet est robuste, car il est “décentralisé”. Le réseau vit par l’agrégation de tous les serveurs connectés entre eux et La perte d’un ou plusieurs serveur ne fait pas tomber le réseau!

SVN? Mercurial ? talk puis focus git.

Une solution décentralisée nous permettrait donc de nous protéger de tout incident pouvant arriver en local dans l’entreprise. Intéressons-nous à un système de gestion de code source décentralise populaire : Git.

Git fut inventé ….

Utilisé pour Linux : fais ses preuves.

L’avantage de Git, c’est que le code source est dupliqué sur chaque machine utilisant le projet. En effet, si le projet central tombe, celui-ci existe sur toutes les machines

#### Laissons faire les experts

L’entreprise peux donc héberger des serveurs de gestion de code source Git. Mais cela rajoute une tache de plus à l’entreprise, à savoir, la configuration et la maintenance des serveurs.

Rappelons-nous que le savoir-faire numéro un de l’entreprise est de produire du logiciel, pas de maintenir des serveurs!

On achète notre pain chez le boulanger, et on ne le fait pas nous même tous les matins pour une bonne raison : Le boulanger a toutes les installations adéquates, son pain est bon et à moindre cout.

De la meme facon qu’on fait appel à un hébergeur pour héberger son site internet, faisons donc appel à un profesionnel de git et déléguons cette tâche à une entreprise dont c’est le métier!

Il existe de nombreux services permmettant d’heberger du code source, Github, bitbucket **SVN explain ???AUTRES**

Leur fonctionnement est similaire.

DESC GITHUB

Sur internet : -> acces depuis partout avec les bons acces.

AU dela la protection contre la perte de lignes de code précieuse, nous allons voir que le système Git nous offer aussi un grande flexibilité dams la collaboration.

### Optimiser le travail en equipe

Ne pas perdre de code source est une chose, mais comment organiser un projet qui emploie, 5, 10, 15, 100 employés sur le même logiciel ?

Là encore, git viens à notre rescousse, grâce à git, un developpeur peux créer une “branche”, c’est à dire une copie du projet, sur laquelle il va pouvoir travailler tout seul, sans que les travail des autres vienne changer sa version.

Quelques minutes, jours ou semaines en suivant, la modification apportée est fonctionnelle et c’est à ce moment la que l’on va l’integrer au projet principal.

On appelle cela un “merge”.

Ci dessous un schema decrivant une utilisation typique de git.

**SCHEMA**

- Possibilité de revenir en arrière apres une erreur vers une version plus stable du projet. Mais cette notion de branche <privée> integrée plus tard dans le projet principal nous apporte plusieurs avantages :

* Possibilité de travailler sur une fonction bien précise du logiciel sans etre dérangé par les modifications des autres.
* Possibilité d’effectuer des tests “dangereux” sur des nouvelles foncionalités sans impacter le projet principal (EX XCODE 5 sur COOLI.)
* Maintenir plusieurs version du logiciel en parallele.

Ntons en plus que la création de branche peux se faire en local a 100pourcent sans internet., ce qui donne enormément de flexibilité.

Un developpeur peut travailler une semaine en local et integrer son travail plus tard.

Cela est extrèmement pratique aujourdh’ui, ave l’expansion du travail à distance.

Ex SVn sync?

-- SVN ++ git (decentralize (a fait ses preuves(ex Linux)

sans internet , branches dématérialise.

Bien, notre code est en lieu sur, hebergé sur internet et sauvegardé sur chaque machine participant au projet. Les developpeurs peuvent travailler de manière autonome sur une fonctionalité bien précise et si il arrive quoi que ce soit à la version principale du projet, nous avons désormais la possibilité de revenir en arrière de manière atomique. (PARLER commits??? + bonnes pratiques??)

### 

### Controle de qualité

C’est bien de pouvoir revenir en arrière à tout moment, mais cela prends du temps, et le temps, c’est de l’argent!. De plus plécouvert tot dans un logiciel, moins il coute cher à l’entreprise (|**CF** graphe MICROSOFT BUGS) .

Nous verrons plus tard qu’une bonne suite de tests sont la meilleure securié contre l’introduction de nouveaux bugs dans le logiciel, mais concentrons nous pour l’instant sur des techniques simple.

Pull request.

L’introduction d’un nouveau bug dans la version principale du projet peux arriver à chaque integration d’une branch privee.

Certaines entreprises **(Fueled ex)** l’integration du nouveau code est effectué par le développeur lui meme.

Cette pratique est peu recommandé car tous les developpeurs ont le “pouvoir” d’integrer de sbugs.

Nous voulons controller la qualité du code integrée au projet et nous allons ut

**Shema pull request**

Premierement, on veux donner ce “pouvoir” de pousser du code sur la version principale seulemt au developpeurs séniors.

Le developpeur sénior revoit le code, peut integrer la fonctionalité en local sur sa machine, verifier que tout fonctionne et ensuite pousser sur le projet principal.

SHEMA.

**REVUE de code**

**EN plus de permettre une double verification de qualité par une paire dyeux fraiche.**

**(car l’erruer est jumaine)**

Certaines boites(ex Fueled) organise des revues de code tout les mois, un senior se pose avec les developpeurs et transmet son savoir, releve les erreur des conceptions etc.

Nous allons profiter du fait que le senior effectue le controle de qualité pour effectuer en meme temps une revue du code.

Au lieu d’une grosse réunion une fois par mois de revue de code, celles ci vont etre effectués tous les 3 jours / semaines. A chaque integrations.

Nous allons voir que cette pratique a ne nombreux effets seconfdaires trés positifs sur le projet :

* Les erreurs de conceptions sont relevés très tot, ce qui permet de les corriger avant meme de les integrer au projet principal.
* Le partage de savoir et d’expérience entre le senior et les developpeurs juniors se fait plus souvent ce aui permet au junior d’apprendre petit peu par petit peu, mais régulièrement.
* Met en avant la collaboration entre les developpeurs, renforce l’esprit d’equipe.
* Au niveau humain, cela va motiver e developpeur junior qui va etre plus attentif au travail produit, sachant qu’il sera revu. Tout en lui permettent de “grandir professionellement.

Etudes Motivation etc. (psychologie, l’aspect humain est trop souvent oublié.

Github (meme pour les autres) nous permet de réaliser ce controle de qualité/ revue de code meme à distance.

Le developeur junior <Pouse son code> demande a etre integré (Pull request)

Le senior peut laisser des commentaire et, en cas de problème majeur, demander des modifications. Celui-ci fusionnera la branche privée du developeur une fois tous les problèmes résolus.

Perte de temps? Non bug coute cher cf Microsoft article

**SCHEMA tYpique pull request ->> commentaire ->> amelio (transfert de connaissance --> MERge.**

## Gestion du Produit

Le produit est une autre étape du processus de création d’un application iPhone. C’est le résultat de la compilation du code source. Dans le cadre d’une application iPhone, cette compilation donne naissance à une application que l’on peut executer sur un telephone.

Comme dans l’étape précedente, on va tester le produit au maximum afin de déceler des eventuels disfonctionnement le plus tot possible. Rappelons nous que plus un bug est trouvé tard, plus il coute cher (**CF Sheme etude MSoft)**

Souvent, les personnes effectuant ces tests sont des Testeurs à plein temps.

En effet, les developpeurs sont les pires des testeurs! Ils connaissent bien le logiciel et savent incosciement les partie de l’application qui sont suseptibles de présenter des anomalies.

Les testeurs professionnels, eux sont souvent bien plus rapides, car c’est leur métier.

Ils peuvent par exemple mettre en place des tests automatiques qui vont tenter de s’authentifeir sur l’application avec 1000 adresses mail différentes.

Cela peut etre fait par les developpeurs, mais encore une fois, le but et de garder le precieux temps de développeurs à construire l’application.

Pour finir ,les testeurs sont plus “objectifs” et vont tester methodiquement chaque fonctionnalitéécrite dans le cahier des charges.

Bien que les tests soient effectués par les testeurs, il est interssant d’embarquer d’autres membres de l’équipe.

Par exemple les chef de projets et les managers connaissent bien le projet et le client et vont etre excellent à deceller les parties à modifier pour satisfaire le client.

D’ailleurs, c’est aussi un excellente idee d’inclure le client car cela permet d’etre certain de ne pas s’eloigner de la demande initiale.

Ce qui est interessant ici, c’est qu’au delà de la recherce de bugs pure et dure, on va utiliser cette campagne pour collecter d’autres informations dans le but d’ameliorer le produit.

Une fois l’application en main, les designers par exemple, vont vouloir aggrandir un bouton qui s’avère trop petit à l’utilisation, ajouter une animation pour rendre l’experience utilisateur plus vivante, ou meme repenser totalement l’interface.

Encore une fois, il ne faut pas oublier le coté humain et l’esprit d’equipe qui est au coeur de la création d’un produit de qualité. Chacun aura son mot à dire et va se sentir plus impliqué dans le produit. Un gain d’interet et donc de productivité et de bien etre au travail.

Voici la liste des retours collectés à ce étape :

- Disfonctionnement

- Suggestion d’expériencre utilisateurs

- Retours sur la performance, ralentissements

- suggestion d’addition/ suppression de fonctionalité

Ces avantages vont indéniablement cous permettre de tendre vers un meilleurs produit. Mais ce la demande du temps.

En effet, l’application en cours de developpemnet n’est pas toujours en mesure d’etre compilée. Par exemple dans le cas de l’integration d’une fonctionnalité en cours. Les developpeur peuvent perdre du temps à creer le “build”. Les campagnes de test ayant pour but d’etre effectués souvent, il faut réduire le temps perdu par le developpeur à creer l’application.

Une foi un bug fixé, il faut recommencer à nouoveau, compiler la version rectifiée et la re distribuer au testeurs pour etre validée.

**Création du produit automatique.**

L’idee ici est de pouvoir creer une version testable de l’application le plus vite possible apres une modification.

Nous allons voir ici que la mise en place du serveur de getion de code source dans le chapitere pécedent va nous etre d’une aide précieuse.

Construire des version de l’application en continue durant le developpement porte un nom “**l’integration continue**”

Le principe est simple. Dès qu’un changement est detecté sur la branche “developpement” du projet

Une tache de fond sur le serveur d’integration continue va chercher les changement apportés et lancer une compilation.

**ScHEMA**

-Detection d’iun changement

- recuperation des nouvelles sources

- compilation--> build

Compile bien – envoie de mail a la persone qui pete le build

Compile sur une autre machine (probleme de librairie etc, pas de config locale)

Boite Fun pete le build bonnet d’ane, alarme etc. ==>> pousser a avoir un produit delivrable a tout moment.

Jenkins +testflight / Hockeyapp

Avantages : objectifs

Build one step.

Distribute

Automatise some tests.

considérables et sont

Retour des test d’integrations

Qa bots

Meme methodologie, Creation, tests et retour à la case depart.

Creation :

Dans le but the TESTER consciencieusement chaque étape de la création de notre application,

Nous auront des testeurs qui vont tester l’application en utilisation réelle.

Afin de pouvoir détecter un bug le plus tot possible, une campagne de test doit etre organisée

Optimiser : serveur creer automatiquement le build et le publie pour les testeurs

Test

Optimisation : lnach test automatique on push

Exemple de configuration Typique (expliquer

Automatisation:

Build auto

De la même manière que l’erreur est humaine, il est impossible de trouver la meilleure solution du premier coup. (**TROUVER EXEMPLE).**

En programmation, c’est la meme chose. L’exp donne des retours sur l’utilisation.

#### Controller la qualité régulièrement

Qa des le debut et Souvent pour affiner le software –

Puis ameliorer en continu

. (need for build in one step, cause multiple builds) (jole list (former Mic / stackoverflow/exchange)

build souvent, automatiser.

Integration continue

Ex Xcode (ou jenkinsbots) : push sur master - > pushjer version a la qa. AUTOMATIQUE :)

Aussi rajouter des tests par exemples

------- TRANS Amelioration doit etre conduite par les utilisateurs.

## Feedback

Encore une fois, on veux verifier au plus vite nos “Assumptions” .

Comme dans les autres étapes de la creation de l’application, on veux ici resserer au maximum le cercle des itérations pour avoir un retour des utilisateur le plus tot possible dans le processus de l’application.

Crerr

Test

Analytics

Rerun

**Analytics** - > features used> unused? Validate prediction.

CANOOT simply find thge truth, only educated guesses , MATHEMATICALLY VERIFY.

Exx TVShowzz.

Exx cooli

**LEAN MGMT – MVP (the lean startup Eric Reiss)**

**Crahs reports**

**No crash app used? Fine let’s optimize now -->> Ex facebook does it , ex Thibualt subject Ab testing**

# Vers un meilleur code

## Syntaxe

## Architecture

## Tests

# Conclusion

# Annexes

# References

Pourquoi vouloir construire de meilleures Apps ?

* 900 000 apps sur le store
* L’app doit être vraiment bonne pour sortir du lot

-

**+ Fiable** : Test unitaires, test d'integrations, campagne de tests, pull requests, code reviews, formation

**+ Maintenable** : Code propre, Design patterns, UML

**Amélioration continue** : Analytics, feedback utilisateur, heatmaps, tests AB, itérations, Minimum viable product, crash logger

**Automatisation** : Automatisation, Integration continue, Xcode BOTS, Jenkins

**Gestion du code source** : Git, Github

Objectifs :

preuve de sa rigueur et sa méthode dans l’analyse et la formulation de recommandations opérationnelles.

**Forme : 50 à 60 pages (hors annexes)**

**problématique** et de présenter les **éléments du contexte de travail** dans lequel elle apparaît ainsi que la manière dont le sujet sera traité pendant le stage et dans le mémoire.

**éléments nouveaux pour la profession**.

Il ne peut se contenter de relater des données que tout le monde connaît.

Il fait état de ces données dans un contexte particulier et leur donne ensuite une dimension plus générale au service de l’ensemble des professionnels intéressés par le sujet traité.

Le sujet de mémoire doit :

* -  fournir des indications sur la technologie qui sera décrite ou sur le contexte particulier  d’utilisation d’une technologie ou sur l’association de deux technologies dans un projet,
* -  faire état de difficultés spécifiques de mise en œuvre d’une solution technique et/ou  mentionner des problèmes concrets d’utilisation de telle ou telle solution technologique,

**synthétiser les connaissances acquises durant ses cinq années d’études en les mettant au service des besoins concrets de l’entreprise.**

L’introduction

* **cadre du travail** et présente **le plan suivi**.
* **doit susciter l’intérêt du lecteur** et lui **donner envie de lire le mémoire.**

Arial 11

Titres et sous-titres : gras 12, un seul retour avant et après Interligne : 1,5 et Marges : 2.5 cm dans toutes les directions.

**Intro**

**pourquoi le sujet traité est-il intéressant** ?

* -  **L’actualité**
* **-  Les différents aspectss  (économiques, culturelles...) ;**
* -  **L’objectif du mémoire : la finalité de votre travail ;**
* **-  La méthodologie de travail : comment les informations ou données ont été collectées et  analysées ;**
* **-  La justification du plan : pourquoi vous avez choisi tel mode d’organisation des parties votre  mémoire plutôt qu’un autre ;**
* **-  L’annonce du plan.**

Le corps du mémoire est subdivisé selon un plan structuré et logique.

Selon les besoins on peut retrouver successivement des parties, chapitres, sections et paragraphes.

La **conclusion** résume le travail et en indique les résultats obtenus. Elle peut aussi mentionner d’autres méthodes possibles et s’ouvrir sur de nouvelles pistes, suggérer d’autres études possibles. Elle comporte :

* -  Le rappel des objectifs et de la méthodologie ;
* -  Les principaux résultats ;
* -  Les limites de votre travail ;
* -  Les prolongements possibles du mémoire.
* La bibliographie  C’est une partie importante du mémoire qui reprend l’ensemble des documents (livres, revues, rapports, site internet...) consultés pour la réalisation du mémoire. Ces documents doivent être classés par ordre alphabétique d’auteurs, tout document confondu. Elle se trouve avant les annexes du mémoire.
* Le glossaire  Si nécessaire, l’étudiant peut ajouter un glossaire à la fin de son mémoire. Celui-ci est une liste qui reprend et définit les termes techniques ou expressions en langue étrangère figurant dans le texte du mémoire (ils apparaissent en italique dans le texte). Ces termes ou expressions doivent être classés alphabétiquement.
* Les annexes  Les documents utiles à l’étude mais qui alourdiraient le texte plutôt que d’en favoriser la compréhension doivent figurer dans la rubrique Annexes. Les annexes doivent être complémentaires du corps du mémoire.

Les notes de bas de page

Placées, par définition en bas de page, ces notes remplacent les parenthèses qui alourdissent souvent le texte, allongent les phrases et nuisent à la compréhension.

Les références bibliographiques

Pour les références dans le corps du texte, ne citez que le nom de l’auteur du livre ou de l’article et l’année de parution ; le complément d’information (éditeur, nom de la revue...) figure en bibliographie. Pour les références bibliographiques, la norme est la suivante :

Ouvrage : Nom et prénom de l’auteur (date de publication), Titre de l’ouvrage, Éditeur, Lieu d’édition. Article : Nom et prénom de l’auteur, « Titre de l’article », Titre de la revue, Vol x, N° x, p. x-y, date de publication.

Les citations : entre guillemets et recopiées très exactement. Si vous ne citez pas la phrase entière, remplacez la partie manquante par des points de suspension

- pas de citations trop longues dans le corps du texte, car elles ralentirssent la lecture et nuisent à la compréhension au lieu de la faciliter.

- recherche théorique

-les réalisations en entreprise,

- l’analyse du contexte, etc.

Exposer son point de vue et l’exposer

montre que l’étudiant est capable de déterminer un problème

de chercher les informations et les données théoriques adéquates pour le

comprendre et le

solutionner,

de mettre en œuvre la solution et de l’évaluer,

de définir les perspectives pour lui et pour l’entreprise.

En d’autres termes, il s’agit de produire une réflexion approfondie sur un sujet en combinant les connaissances théoriques et les expérimentations pratiques.

Theorie et pratique s’appuyer sur des anecdote de stage.

C’est donc cette recherche d’originalité dans l’analyse, dans la variété des connaissances mises en œuvre, dans une recherche de nouvelles données, dans les solutions proposées pour résoudre un problème identifié ou améliorer une situation, dans l’application de ces solutions et dans l’évaluation des résultats obtenus que va se trouver la nouveauté.

capacité à travailler dans un contexte professionnel,

à adapter son travail aux réalités de terrain, à chercher des solutions répondant à un besoin précis.

Se positionner en tant qu’ingénieur en informatique et transposer sa capacité de travail dans des contextes diversifiés.

**Conclusion**

Rappeleridées principales exposées tout au long du mémoire.

mettre en valeur l’idée centrale et répéter le fil conducteur du mémoire.

* possible d’ouvrir le débat vers un autre sujet qui pourrait utilement prolonger celui traité mais attention à ne pas laisser penser que l’étudiant n’a que très partiellement abordé la question qu’il était censé exposer de manière professionnelle.

Des considérations sur les progrès que le stage a permis de faire du point de vue professionnel et personnel sont toujours intéressantes en mentionnant brièvement les problèmes que le stagiaire a dû résoudre par lui-même (car non appris à l’école) et les acquis du point de vue de l’organisation du travail en équipe dans l’entreprise.

**Introduction**

Tout a la fin

fournir au lecteur les indications dont il a besoin pour comprendre la construction du mémoire et lui indiquer les idées principales qu’il va trouver dans le document.

Intro en accord avec le developpement qui va suivre.

**Prez’**

montre qu’il est un **professionnel** du milieu de l’informatique et du numérique, **compétent** et **sérieux**, **à l’aise devant un auditoire**, capable de **défendre son point de vue** et de **répondre aux questions qui lui sont posées**, **ouvert aux idées novatrices** et ayant la capacité à **s’investir dans des projets d’envergure**.

BUILDING BETTER SOFTWARE Title Everything to know to found a successful It company

Create libraries

Github revolution

Before / Now

Joel on software checklist

Not reinventing the wheel

Analytics (don’t build the best product in the world if nobady uses it!)

Continuous dev

Easy build

Tests

Scrum

Remote workers

Meetings are toxic

Manage dematerialized team

TDD

AGILE

MVP -> Iterate

UML datamodel

A quest for better software ( code reviews, git, feature brachs history)

Always ask why we do what we do?

Some are cost killers, we are time killers, automate evry automatable tasks and save time

FORMATION TRAIN Employees

Always use latest tools and techo, Time spent on training is gained on debugging!!!

Cleaning by graphing dependencies (python script)

Joels list

Bug are expensive, find them day 1 (continuous tests)

Annexes:

Cooa design patterns

iO5 5 ray wenderlich

TDD in iOS

Joel on software

XCODE 5 continous integ demo!!!

Convetion over onfiguration (Obama quote about suits -> lets think more about features than implementation problems! Tools, frameworks, design patterns etc ☺

REF https://cleancoders.com/

vocabulaire, grammaire, syntaxe, présentation générale, qualité des arguments développés et la construction de la démonstration manière dont l’étudiant expose les recherches qu’il a effectuées, les moyens qu’il a utilisés pour recueillir des données théoriques ou techniques, les démarches entreprises envers des personnes ressources, la sélection des données pour bâtir la démonstration, la pertinence de ces données dans le cadre du problème identifié et des objectifs de résolution annoncés.

degré de reproduction ou d’innovation de la solution proposée par rapport au contexte du stage et la connaissance de l’étudiant des différentes actions réalisées habituellement dans cette entreprise ou dans d’autres entreprises pour résoudre des difficultés similaires à celles qui sont à l’origine du projet de stage.

capacité de distanciation du stagiaire avec les événements immédiats pour produire un exposé clair et efficace. compétences mises en œuvre sur le plan technique, les aptitudes du candidat à élaborer un plan d’action tenant compte de l’ensemble des paramètres présents dans l’environnement : concurrents éventuels, respect des dates limites, conformité avec la commande de travail, etc.

expose sa maîtrise de compétences techniques et dont il rend compte de son travail.

respecté la nomenclature en vigueur pour citer les sources documentaires auxquelles il a fait appel.

AUCUN plagiat (recopie de données sans citation de la source d’information) n’est présent dans le document.

Dev sont bon a faire... puis automatiser, laisser les diriger! ;P

Voir notation