ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT DE PROJET PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE
EN GÉNIE LOGICIEL

PAR
Christophe COMMEYNE

ÉTABLISSEMENT D'UN MODÈLE D'ESTIMATION *A POSTERIORI* DE PROJETS DE MAINTENANCE DE LOGICIELS

MONTRÉAL, LE 17 AVRIL 2014

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Alain Abran, directeur de projet
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Alain April, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

REMERCIEMENTS

Je tiens à remercier mon directeur de projet Dr Alain Abran d'avoir accepté mon projet et de m'avoir ainsi permis de découvrir et approfondir un domaine du génie logiciel qui m'était inconnu au début du programme de maîtrise.

Ensuite, je remercie mon gestionnaire Simon-Charles Duberger de m'avoir accordé le temps nécessaire pour réaliser ce travail, ainsi que mes collègues avec qui j'ai pu échanger sur les découvertes que j'ai faites en travaillant sur ce projet.

ÉTABLISSEMENT D'UN MODÈLE D'ESTIMATION *A POSTERIORI* DE PROJETS DE MAINTENANCE DE LOGICIELS

Christophe COMMEYNE

RÉSUMÉ

L'analyse des estimations historiques de l'effort prévu pour implémenter des demandes de changement réalisées par une équipe de maintenance montre des écarts importants entre l'effort réel et l'effort estimé en utilisant une adaptation de la technique du Planning Poker. L'imprécision du processus d'estimation actuellement utilisé par l'équipe est de 58 %. Cette grande volatilité de ces estimations contribue donc à rendre le travail d'estimation de l'équipe presque inutile.

Dans ce projet, la norme COSMIC – ISO 19761 – a été utilisée pour mesurer la taille fonctionnelle des tâches liées aux demandes de changement. En utilisant le modèle d'estimation construit à partir de la taille fonctionnelle des tâches et de l'effort réel, l'imprécision du modèle d'estimation a été réduite à 16,5 %. Le nouveau modèle d'estimation a aussi permis d'observer une certaine constance dans la capacité de l'équipe à implémenter ces changements tout au long d'une année, itération après itération. Les résultats obtenus appellent donc à revoir le processus d'estimation actuel de l'équipe.

DESIGN OF AN *A POSTERIORI* ESTIMATION MODEL FOR SOFTWARE MAINTENANCE PROJECTS

Christophe COMMEYNE

<**ABSTRACT**>

A close look into the historical estimated efforts of change requests made by a maintenance team using a custom implementation of the Planning Poker technic reveals important differences between planned effort and the actual effort: the volatility of these estimations makes it as if the estimation work done by the team is almost useless since the inaccuracy of the actual estimation process is 58 %.

This project has used the COSMIC ISO standard 19761 to measure the functional size of the tasks. After designing a new estimation model using the functional size of the tasks and the actual effort, the inaccuracy of the estimation model has been lowered to 16.5 %. The new estimation model also shows that the team was able to implement the change requests at a constant pace all year long, one iteration after another. These findings call for a revision of the actual estimation process used by the team.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ISO	International Organization for Standardization
COSMIC	Common Software Measurement International Consortium
ISBSG	International Software Benchmarking Standards Group
IP	Internet Protocol
UX	User Experience
SDK	Software Development Kit
UI	User Interface
MMRE	Mean Magnitude of Relative Error

LISTE DES SYMBOLES ET UNITÉS DE MESURE

CFP	COSMIC Function Point

INTRODUCTION

Apparue au début des années 2000, l'approche agile est maintenant installée dans de nombreuses entreprises. Une équipe de maintenance d'une entreprise a choisi d'implémenter la méthodologie Scrum pour devenir plus agile dans ses processus de développement de logiciels. Parmi les activités réalisées par cette équipe de maintenance, il y a une rencontre au début de chaque itération qui offre l'occasion à l'équipe de découvrir les fonctionnalités à mettre en place dans l'application. C'est le moment choisi par l'équipe pour faire les estimations de l'effort nécessaire pour accomplir les tâches à l'aide d'une adaptation de la technique du Planning Poker. Cependant, il n'existe pas de processus mesurant la qualité des estimations ni de processus permettant d'améliorer la précision de ces estimations. En réalité, l'équipe n'a tout simplement aucune mesure reliée aux estimations : les précédentes estimations ne sont pas réutilisées. À chaque itération, l'équipe recommence donc à zéro, sans possibilité de s'améliorer.

L'objectif de cette activité de synthèse est d'établir un modèle d'estimation *a posteriori* des tâches de maintenance. Dans un premier temps, le travail donnera un aperçu de la qualité des estimations actuelles faites par l'équipe. Dans un second temps, la création du modèle d'estimation permettra d'améliorer la précision des estimations et leur reproductivité grâce à la norme COSMIC – ISO 19761 – : cela signifie que, d'une itération à une autre, une demande de changement d'une taille donnée donnera une estimation de l'effort du même ordre de grandeur et ce, peu importe les membres de l'équipe participant à l'estimation, leur expérience et leur connaissance du logiciel en question.

Le présent rapport est divisé en 4 chapitres. Le premier chapitre présente l'entreprise, son secteur d'activité, sa structure et le fonctionnement de l'équipe chargée de compléter les tâches de maintenance. Le second chapitre présente les tâches sélectionnées pour ce travail, les estimations de l'effort qui avaient été faites ainsi qu'une analyse de ces données historiques. Le troisième chapitre présente la mesure COSMIC, ses principes et sa mise en pratique sur les

tâches retenues. Ce chapitre présente également la transmission des mesures auprès de l'organisme ISBSG. Le quatrième et dernier chapitre présente la construction du modèle d'estimation *a posteriori* à partir des mesures recueillies ainsi que de son exploitation pour faire une estimation *a priori* de l'effort afin d'implémenter une demande de changement. Ce chapitre présente également l'évolution du modèle d'estimation en y intégrant une nouvelle donnée. Finalement, une conclusion résume ce rapport.

CHAPITRE 1

MISE EN CONTEXTE

1.1 L'entreprise

Le secteur de l'entreprise dans laquelle ce projet de synthèse a été réalisé est la sécurité physique et la surveillance. Sa spécialisation est le développement de solutions informatiques sur IP. Elle ne produit pas des logiciels sur mesure, mais une plateforme dont l'utilisation est licenciée à de nombreux clients provenant de différents secteurs d'activités. La plateforme développée répond aux besoins variés des forces de l'ordre, des gouvernements, des écoles, des aéroports, des casinos, des détaillants et bien d'autres encore. Elle est utilisée partout dans le monde, aussi bien sur le continent américain (Amérique du Nord, Amérique centrale, Amérique du Sud) que sur le continent européen, au Moyen-Orient, en Asie et dans l'Océanie.

Bien qu'elle ait été pionnière dans le domaine de la vidéosurveillance sur IP peu avant le début des années 2000, l'entreprise ne se trouve pas dans une situation de monopole; elle fait donc face à une concurrente féroce qui la contraint à être compétitive et, en conséquence, à être efficace dans le développement de nouvelles fonctionnalités tout comme dans ses processus organisationnels.

1.2 Structure interne de la compagnie (Recherche et développement)

Pour pouvoir maintenir une plateforme comprenant plusieurs millions de lignes de code, l'entreprise a réparti les responsabilités de différentes parties du logiciel à plusieurs équipes. L'application de vidéosurveillance est maintenue à la fois par une équipe chargée des demandes de changements au sein de la plateforme et par une autre équipe chargée d'intégrer de nouvelles caméras. L'application de contrôle d'accès comporte elle aussi deux équipes pour l'intégration des demandes de changement dans la plateforme et pour la gestion des lecteurs de cartes. Il y a trois équipes qui travaillent dans le module de reconnaissance de plaques

d’immatriculation : une équipe se charge des demandes de changement dans la plateforme, une autre s’occupe des algorithmes consacrés à la capture et à l’analyse des images et une autre pour le développement du micrologiciel des caméras. À cela s’ajoute une autre équipe dédiée à la maintenance de l’infrastructure de la plateforme permettant aux applications de communiquer entre elles. Il y a aussi deux autres équipes qui se consacrent spécifiquement à la maintenance des interfaces usagers, en accord avec 2 architectes UX pour assurer une certaine uniformité visuelle de l’ensemble des applications de la plateforme.

En parallèle à toutes ces équipes, il y a une équipe dédiée aux applications s’exécutant sur les terminaux mobiles et une autre équipe pour le développement de la version infonuagique de la plateforme. Finalement, il y a deux équipes destinées à la programmation de l’extensibilité de la plateforme (SDK et Custom Solutions).

La taille de ces équipes variant entre 5 et 10 personnes, cela donne une centaine d’employés dédiés à la programmation de la plateforme alors que l’entreprise compte un peu plus de 500 employés.

En général, les demandes de changements transmises aux développeurs sont analysées au préalable par l’équipe d’ingénieurs système qui s’assurent de comprendre les besoins et de les formuler de la manière la plus précise possible. Les modifications effectuées par les développeurs sont validées par deux équipes de testeurs. La plus grande des deux équipes se charge de tester le bon fonctionnement de la plateforme, à l’exception de ce qui touche à la reconnaissance de plaques : il s’agit de la responsabilité de la seconde équipe de testeurs de valider spécifiquement le matériel et le logiciel se rapportant à ce domaine.

1.3 Équipe étudiée

Le présent rapport de projet concerne directement l'équipe chargée de l'implémentation des demandes de changements liées à la partie reconnaissance de plaques. Cette équipe est composée de 8 personnes et elle interagit avec d'autres équipes, qui sont :

- Imagerie (5 personnes);
- Testeurs (7 personnes);
- Matériel (6 personnes);
- Production (4 personnes);
- Support à la clientèle (2 personnes).

À ces équipes s'ajoutent les analystes de marché, la gestionnaire de projets et les ingénieurs des ventes. Toutes les équipes mentionnées sont dépendantes les unes des autres et un retard dans une des tâches de ces équipes peut fortement impacter les autres équipes. En conséquence, une bonne estimation du temps nécessaire à la réalisation d'une tâche ne peut être qu'un apport positif.

Durant l'année 2012-2013, le développement des tâches de maintenance réalisées par l'équipe a suivi certains principes de l'approche agile, et plus précisément la méthodologie Scrum. Le développement s'est effectué à l'intérieur d'itérations variant de 3 à 6 semaines. Au début de chaque itération, tous les membres de l'équipe se sont réunis durant un certain nombre d'heures (de 4 à 8 heures, jamais plus) et ont passé en revue avec le gestionnaire de l'équipe qui avait le rôle de *Product Owner* la liste des demandes de changements à effectuer pour l'itération courante. Ces discussions comprennent les détails techniques concernant les modifications à effectuer, les potentiels points d'extension existants (sans avoir d'accès au code source) et les impacts de ces changements sur le reste de l'application. L'objectif est de s'assurer que tout le monde ait une connaissance des tâches à accomplir dans l'itération courante. Il s'agit là d'un

excellent moment pour partager les connaissances (à la fois celles du système et celles plus techniques).

L'estimation de l'effort nécessaire pour implémenter chacune des tâches est obtenue suite à un consensus entre les différents membres de l'équipe. Cette technique d'estimation est appelée le Planning poker [1] et comprend les étapes suivantes :

- Une tâche de référence doit être d'abord sélectionnée et sa taille estimée. Chaque membre de l'équipe propose une valeur. Cette valeur est choisie parmi un ensemble de tailles possible qui sont : 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100;
- Si de trop grands écarts sont observés entre les différentes propositions, une discussion a lieu et un autre tour est effectué jusqu'à l'obtention d'un consensus;
- Par la suite, toutes les tâches restantes sont estimées de la même manière. La taille estimée d'une nouvelle tâche est toujours relative à la tâche de référence;
- Finalement, ces tailles sont ensuite transformées en intervalles d'heures en se basant sur la vélocité historique de l'équipe.

En pratique dans cette entreprise, l'équipe de maintenance a choisi d'effectuer ses estimations de l'effort directement en termes d'heures, sans se baser sur une quelconque référence, ni même un historique. Lorsque l'itération est complétée, les estimations initiales sont comparées avec l'effort réel et elles sont ensuite consignées sur le site intranet de l'équipe. Ces valeurs ne sont pas réutilisées pour une itération ultérieure.

Durant l'itération, une rencontre de 15 minutes est effectuée chaque matin. Chaque membre d'équipe tente d'énoncer *brièvement* le travail de la veille, ce qui le bloque et le travail qu'il pense pouvoir accomplir durant la journée. Chaque jour, les membres d'équipe entrent le nombre d'heures passées sur les tâches. Le nombre d'heures allouées à une tâche peut être réajusté à la hausse si le besoin s'en fait ressentir. Le graphique représentant la progression est présenté chaque jour par le Scrum *Master*. Ce dernier a à sa charge (conjointement avec le

Product Owner) la responsabilité de déplacer des fonctionnalités qui ne pourront pas être développées durant une itération en cours à la prochaine.

Lors de la dernière journée, une démonstration des changements qui ont été implémentés durant l'itération est faite au *Product Owner*. En fonction de ce qu'il voit et des critères d'acceptation, il détermine si l'application peut être envoyée à l'équipe de testeurs pour une validation et une vérification plus approfondie.

CHAPITRE 2

LES TÂCHES DE MAINTENANCE

2.1 Sélection des tâches de maintenance

L'objectif de cette activité de synthèse est d'établir un modèle d'estimation *a posteriori* des tâches de maintenance réalisées par l'équipe étudiée. Pour cette activité, toutes les tâches sélectionnées proviennent seulement de cette équipe et s'étalent sur la période du 16 avril 2012 au 16 mai 2013. Chaque tâche est composée d'un ensemble de sous-tâches et est assignée à un ou plusieurs membres de l'équipe. Le Tableau 1 présente la liste de toutes les itérations complétées durant la période concernée. Il est à noter qu'aucune tâche n'a été complétée durant la cinquième itération : il s'agissait de stabiliser le code.

Tableau 1 – Liste des itérations des tâches étudiées

Numéro de l'itération	Période	
	Du	Au
1	16 avril 2012	4 mai 2012
2	4 juin 2012	22 juin 2012
3	16 juillet 2012	10 août 2012
4	13 août 2012	7 septembre 2012
5	17 septembre 2012	9 Novembre 2012
6	16 janvier 2013	15 février 2013
7	18 février 2013	22 mars 2013
8	25 mars 2013	26 avril 2013
9	30 avril 2013	16 mai 2013

Les tâches choisies ont des caractéristiques très similaires. Tout d'abord, elles apportent toutes au moins une fonctionnalité visible par l'utilisateur final; il ne s'agit pas de travaux

d'infrastructure. Ensuite, elles contiennent majoritairement des ajouts et des modifications au code de la logique d'affaire. Les modifications à l'interface utilisateur se limitent principalement à l'ajout de contrôles; les changements n'impliquent pas de modifications majeures nécessitant de nombreuses discussions sur l'ergonomie des changements.

Tableau 2 – Sélection des 24 tâches et écarts d'estimation entre l'effort estimé et réel

#	Itération	Tâche	Effort estimé (en heures, ajustées à 70 %)	Effort réel (en heures)	Erreur relative + : sous-estimation - : surestimation
1	1	#52546	97,25	173,25	44 %
2	1	#44323	178,88	96,25	-86 %
3	2	#58731	28,01	68	59 %
4	2	#79533	36,01	33	-9 %
5	3	#59710	22,66	22	-3 %
6	3	#77760	110,01	57,5	-91 %
7	4	#85738	75,72	39	-94 %
8	4	#85747	84,27	39	-116 %
9	4	#78934	61,45	25,9	-137 %
10	4	#78935	90,02	61	-48 %
11	6	#44289	34,3	63	46 %
12	6	#39148	10,1	5	-102 %
13	6	#93307	104,29	36	-190 %
14	6	#91764	15,72	6,5	-142 %
15	7	#95760	31,44	36	13 %
16	7	#102114	75,72	41	-85 %
17	8	#107205	48,57	37,75	-29 %
18	8	#107206	8,57	35,75	76 %
19	8	#107207	28,57	20,75	-38 %
20	8	#107209	12,87	28,7	55 %
21	8	#107208	34,29	32	-7 %
22	8	#107211	32,86	36	9 %
23	8	#107210	30	18	-67 %
24	9	#108775	77,14	40	-93 %

Toutes les tâches ont été complétées par des développeurs ayant des compétences très similaires dans l'environnement de développement adopté par l'entreprise (le langage de programmation, les outils de développement) et ils ont tous accès aux mêmes ressources en ce qui concerne les informations du système existant.

2.2 Analyse des estimations passées

Le Tableau 2 présente la liste des 24 tâches sélectionnées pour cette analyse. Pour chaque tâche, l'erreur relative a été calculée et est présentée dans la colonne de droite. Cette valeur donne une indication de la divergence entre la valeur estimée par l'équipe et la valeur réelle. Exprimée en pourcentage, la divergence – ou erreur relative – est calculée de la manière suivante [2] :

$$Erreur\ relative\ (\%) = \frac{(Effort\ réel - Effort\ estimé)}{Effort\ réel}$$

L'amplitude moyenne de l'erreur relative[1] des estimations peut être utilisée pour juger de la qualité du processus d'estimation ou du modèle d'estimation [2]. Elle est définie ainsi :

$$Amplitude\ moyenne\ de\ l'erreur\ relative = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{(Effort\ réel - Effort\ estimé)}{Effort\ réel} \right|$$

Un modèle capable de prédire des estimations exactes aura une amplitude moyenne de l'erreur relative égale à 0 %. L'amplitude moyenne de l'erreur relative du processus actuel d'estimation est de 58 %.

[1] Mean Magnitude of Relative Error (MMRE) en anglais

L’effort réel par tâche varie entre un minimum de 5 heures et un maximum de 173 heures, avec la majorité des tâches (18) présentant un effort se situant en dessous de 50 heures. Originalement, 26 tâches avaient été choisies, mais 2 d’entre elles se sont retrouvées exclues pour les raisons suivantes :

- La première tâche a été développée par un stagiaire qui, malgré ses fortes aptitudes, n’avait pas la même expérience que le reste de l’équipe avec le langage utilisé;
- La seconde tâche était un cas extrême. La fonctionnalité demandée était déjà codée, mais elle avait été explicitement désactivée : l’implémentation a consisté littéralement à supprimer une ligne de code pour répondre aux nouvelles exigences.

Lorsqu’on compare l’effort réel avec l’effort originalement estimé par l’équipe (Figure 1 et Tableau 2), on peut constater qu’il y a de nombreux écarts qui dépassent 20 % en valeur absolue. Les plus grands écarts sont une sous-estimation de plus de 76 % (5 journées de développement au lieu de 1 journée estimée pour une itération d’une durée de 5 semaines) et une surestimation de 190 % (1 journée de développement au lieu des 2 journées estimées). L’écart le plus faible est de 3 %.

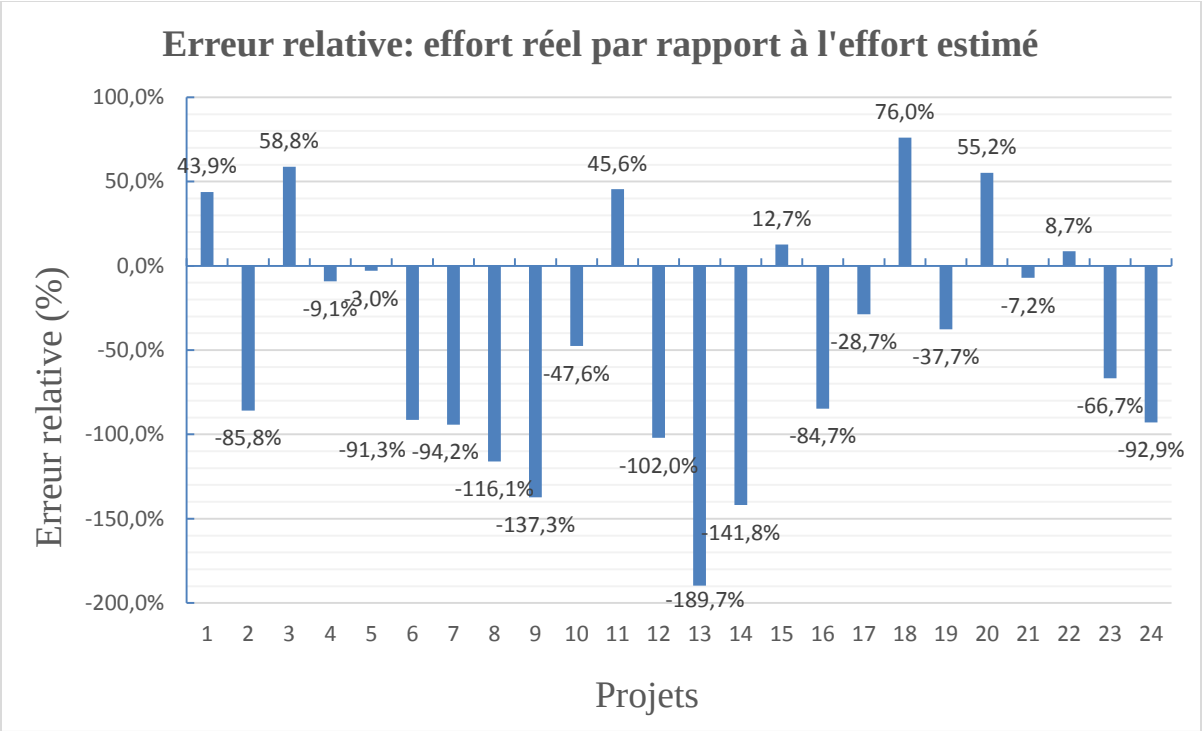

Figure 1 – Erreur relative : l'effort réel par rapport à l'effort estimé

Les chiffres montrent une forte tendance à surestimer l'effort nécessaire (17 tâches sur les 24 mesurées). Étant donné le niveau de granularité de la mesure, les pourcentages peuvent induire en erreur. Les 5 tâches les plus sous-estimées (#1, #3, #11, #18, #20) ont engendré un dépassement de 190 heures, soit presque une itération de 5 semaines à raison de 40 heures par semaine pour une personne. Et qu'en est-il des surestimations? Les 5 tâches les plus surestimées (#8, #9, #10, #13, #14) représentent un total de 195 heures.

En moyenne, lorsqu'on prend les 24 tâches sélectionnées, l'équipe de maintenance a surestimé l'effort global de près de 43 %. La valeur n'est pas négligeable sachant que l'ensemble des tâches a nécessité un effort réel total de 1056 heures. La moyenne n'a pas beaucoup de sens sans la mention de son écart-type. Dans le cas présent, l'écart-type a une valeur de 72 %, indiquant donc une grande variation par rapport à la moyenne.

Avec ces informations, la prochaine question à répondre est la suivante : existe-t-il une corrélation quelconque entre l'effort estimé et l'effort réel?

La Figure 2 présente la régression linéaire montrant la relation entre l'effort estimé et l'effort réel. Avec un coefficient de détermination R^2 de 0,326[2], la relation est faible : celle-ci n'est explicable qu'à 32 % seulement; le 68 % restant n'étant que du bruit. En d'autres termes, le processus d'estimation de l'effort utilisé par l'équipe de maintenance ne permet pas, à l'heure actuelle, de donner des estimations précises de l'effort nécessaire pour implémenter une tâche.

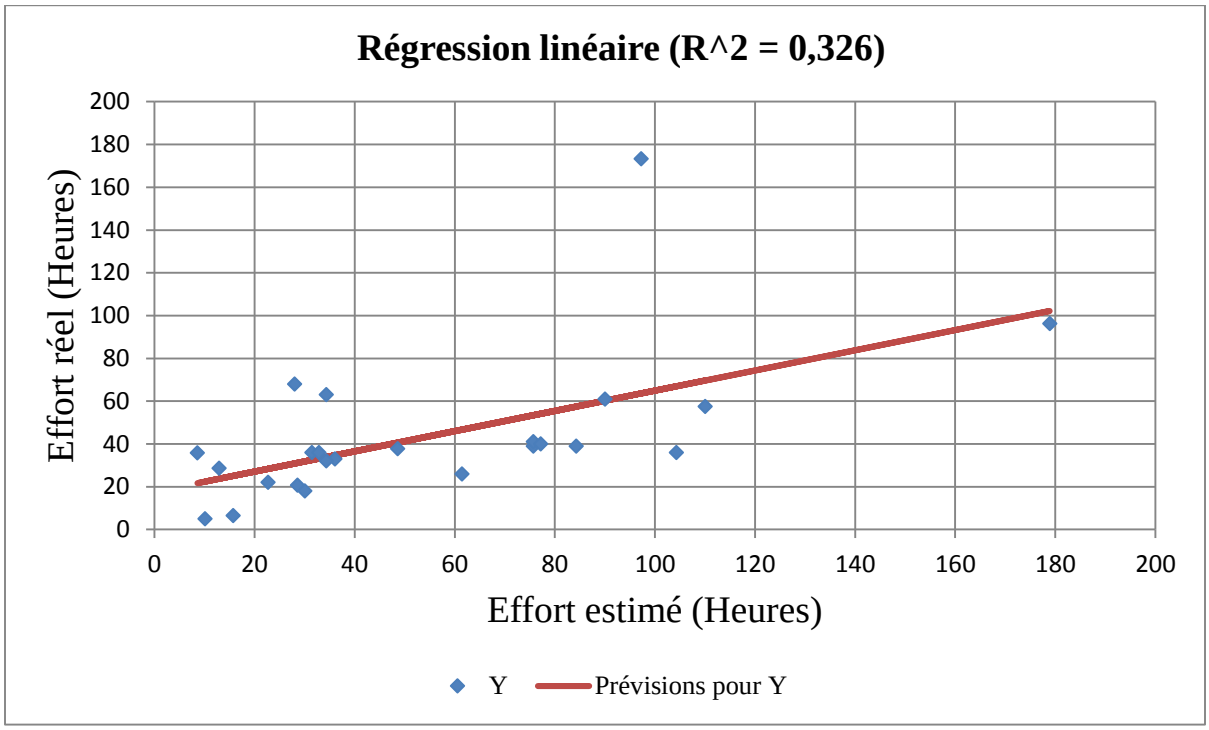

Figure 2 – Régression linéaire : Effort estimé par rapport à l'effort réel

Si le processus d'estimation a amené l'équipe à produire des estimations incorrectes, rien n'empêcherait ce processus d'amener l'équipe à sous-estimer les tâches de manière

[2] Le coefficient de détermination (R^2) a une valeur comprise entre 0 (aucune relation) et 1 (relation parfaite)

systématique. Étant donné le peu de précision du processus d'estimation privilégié par l'équipe, c'est une situation tout à fait plausible. La conséquence immédiate de cette situation sera un retard dans la livraison des applications à l'équipe de testeurs. Cette dernière recevant un flux de travail en provenance de trois équipes différentes, il est important que les équipes en amont respectent leurs estimations de temps le plus souvent possible. Ce retard pourrait soit engendrer le report de la publication de la nouvelle version de la plateforme, soit forcer l'entreprise à publier une version de moindre qualité si elle était tenue de respecter un contrat. Bien entendu, rien n'empêcherait non plus l'équipe de surestimer les efforts de manière systématique. Dans tous les cas, cette imprécision ne facilite pas le travail du gestionnaire de projet dont une de ses responsabilités est de pouvoir répondre en tout temps à ces deux questions importantes : quand est-ce que le projet va se terminer et combien il va coûter.

Deux défauts majeurs ont été constatés dans le processus d'estimation de l'équipe :

A. L'utilisation partielle de la méthode d'estimation avec le Planning Poker;
B. Les limites inhérentes à la méthode du Planning Poker.

Le processus d'estimation se base uniquement sur l'expertise des intervenants (« *educated guess* ») en faisant abstraction d'une référence pour faire les estimations. De plus, l'estimation étant réalisée par consensus directement en termes d'heures et non pas en termes de tailles relatives comme cela est recommandé [1], il ne suffit que d'une simple influence d'une ou plusieurs personnes pour forcer un résultat à la baisse ou à la hausse suite à un simple doute dans l'unique but d'obtenir le consensus. Le fait est que rares sont les situations où plusieurs personnes possèdent, au même moment, le même niveau de connaissance d'une section du programme, ce qui aurait permis de corroborer certains points de vue sur le système à modifier. Pire, la taille de la plateforme étant importante, il devient de plus en plus difficile de se remémorer des sections d'un programme qui n'ont pas été visitées depuis quelques mois, surtout en sachant que ces sections sont amenées à être modifiées au fur et à mesure que des

problèmes sont trouvés et corrigés. Une autre équipe avec le même niveau de connaissances de la plateforme pourrait fort bien ne pas aboutir aux mêmes estimations. Tous ces points contribuent au caractère, jusqu'à un certain point, presque aléatoire des estimations actuelles.

Si les efforts estimés sont aussi imprécis, qu'en est-il de l'effort réel? Est-ce que l'équipe est constante dans sa capacité à réaliser ses tâches de maintenance?

CHAPITRE 3

LES MESURES

3.1 La méthode de mesure COSMIC

Sachant que l'entreprise ne possède pas d'estimations de taille des tâches à accomplir, on doit avoir recours à un autre procédé pour déterminer la capacité de l'équipe à accomplir des tâches de maintenance. La mesure de la taille fonctionnelle à l'aide de la norme ISO 19761 (COSMIC) [2] nous donne cette possibilité. Grâce à cette norme, il est possible d'obtenir une mesure de la taille fonctionnelle d'une application qui soit objective et reproductible par différentes personnes. Une fois cette mesure de base obtenue, la vélocité de l'équipe (une mesure dérivée) pourra être calculée et permettra de répondre à la question posée précédemment, à savoir si l'équipe est capable d'implémenter des demandes de changements d'une manière constante.

La méthode de mesure COSMIC considère un logiciel de la manière suivante. Au minimum, un logiciel réalise une opération. Pour pouvoir accomplir cette opération, il a besoin au minimum d'une donnée (appelée groupe de données) en entrée (entrée) pour générer un résultat (sortie). Durant l'opération, le logiciel peut avoir besoin d'obtenir des données depuis un espace de stockage (lecture) et même d'inscrire des informations dans cet espace de stockage (écriture). Ces 4 opérations sont des mouvements de données et ils sont représentés dans la Figure 3. La mesure COSMIC consiste à identifier des processus fonctionnels, les groupes de données et finalement compter le nombre de mouvements de données d'un processus fonctionnel. L'unité de mesure du nombre de mouvements de données le Point de Fonction COSMIC (CFP) qui correspond à un mouvement d'un groupe de données, quel que soit le type de ces mouvements (c'est-à-dire entrée, sortie, lecture et écriture).

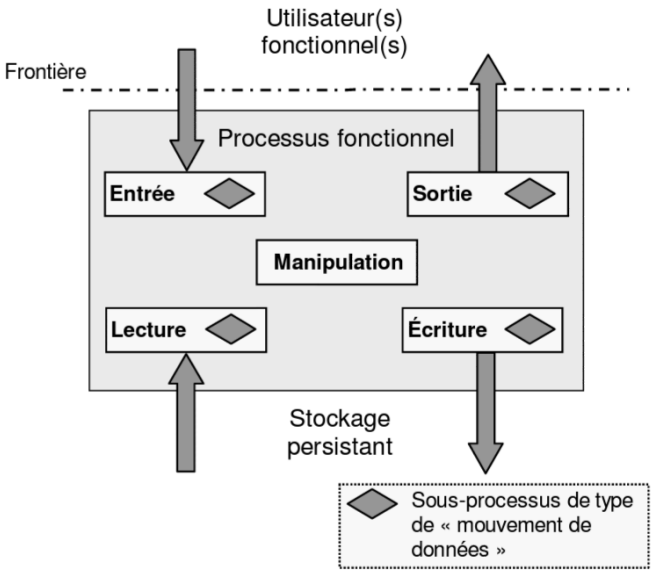

Figure 3 – Processus fonctionnel selon COSMIC[3]

3.2 Préparation à la mesure

Avant de pouvoir commencer la mesure de la taille fonctionnelle des tâches, le mesureur doit préparer le processus de mesure [3]. La première étape consiste à déterminer si la mesure COSMIC est applicable. Cela nécessite d'abord de présenter les différentes applications impliquées. Ensuite, le mesureur doit déterminer l'objectif de la mesure. Après ça, il faut arrimer les couches des applications sur lesquelles la mesure se fait au modèle défini par la norme. Finalement, la mesure est réalisée.

3.2.1 Description des applications

L'équipe de maintenance gère plusieurs applications au sein du domaine de la reconnaissance de plaques. L'application Patroller s'exécute sur un ordinateur portatif placé dans un véhicule.

[3] Figure tirée du manuel de mesure de la norme ISO 19761

Son but est de fournir un retour visuel sur le déroulement des opérations, c'est-à-dire la lecture de plaques d'immatriculation. L'application offre plusieurs modes de fonctionnement qui sont essentiellement liés au type de traitement effectué lorsqu'une plaque est lue. Pour pouvoir configurer l'application Patroller, l'utilisateur doit utiliser l'application Patroller Config Tool qui permet d'ajuster aussi bien des détails visuels que des options plus précises liées à la logique d'affaire.

L'application Plate Reader s'exécute dans chacune des caméras connectées à l'application Patroller et réalise la lecture de plaque. Elle se charge de récupérer les images et de déterminer les numéros des plaques d'immatriculation qui s'y trouvent. La configuration du Plate Reader est réalisée par une application internet appelée Portal. Elle permet de définir les paramètres du Plate Reader dans un simple navigateur.

Non seulement les caméras peuvent se connecter au Patroller (dans une configuration appelée mobile), mais elles peuvent également se connecter à un serveur dans une configuration fixe pour transmettre les plaques lues accompagnées de leurs images. Dans cette situation, la caméra peut être placée en hauteur près d'une autoroute par exemple et envoyer les informations au serveur.

La plateforme est divisée en Rôles ayant chacun leurs propres fonctionnalités (vidéosurveillance, contrôle d'accès, détection d'intrusion, etc.). Ces rôles peuvent être instanciés sur différentes machines. Pour les configurer, l'administrateur utilise l'application appelée Config Tool. Cet outil est commun à toutes les composantes de la plateforme, tout comme l'application Security Desk qui permet aux opérateurs de surveiller les lieux.

En se basant sur la description des applications présentées précédemment, nous pouvons définir que le domaine du type de logiciels est : Logiciel d'application d'affaires. L'application Plate Reader nécessite une librairie provenant de l'équipe d'imagerie. Cette librairie effectue le traitement d'analyse d'images permettant de découvrir la plaque d'immatriculation. Cette

librairie est exclue de notre travail, car l'équipe de maintenance concernée par la mesure n'y fait aucune modification.

Dans le cas présent, il ne s'agit pas de tâches faisant intervenir des algorithmes mathématiques complexes ou des systèmes de simulation, d'autoapprentissage, de prévisions météorologiques. La méthode de mesure COSMIC peut donc être appliquée pour mesurer la taille fonctionnelle des tâches sélectionnées.

3.2.2 Objectif de la mesure

La première chose à définir est la raison d'être de la mesure. Ici, la raison d'être de ce processus est la mesure de la performance de l'équipe sur des tâches de maintenance complétées. Les résultats seront utilisés par la suite pour établir un modèle d'estimation *a posteriori* de tâches de maintenance avec, comme objectif final, l'estimation *a priori* d'un intervalle de temps requis pour accomplir de futures tâches de maintenance.

3.2.3 Arrimage : définition des couches

Pour pouvoir effectuer la mesure, il faut faire le lien entre les différentes couches des applications dont on va mesurer les modifications et les couches définies par le modèle COSMIC. La Figure 4 présente l'arrimage réalisé entre la plateforme et le modèle COSMIC.

Grâce à cet arrimage, les frontières logicielles à travers desquelles les objets d'intérêts transiteront peuvent être déduites. Comme présentées à la Figure 5, des frontières ont été appliquées pour diviser certaines applications, car le travail pour mettre en forme les données est aussi grand que celui permettant de les obtenir et nécessite souvent un processus fonctionnel en lui-même.

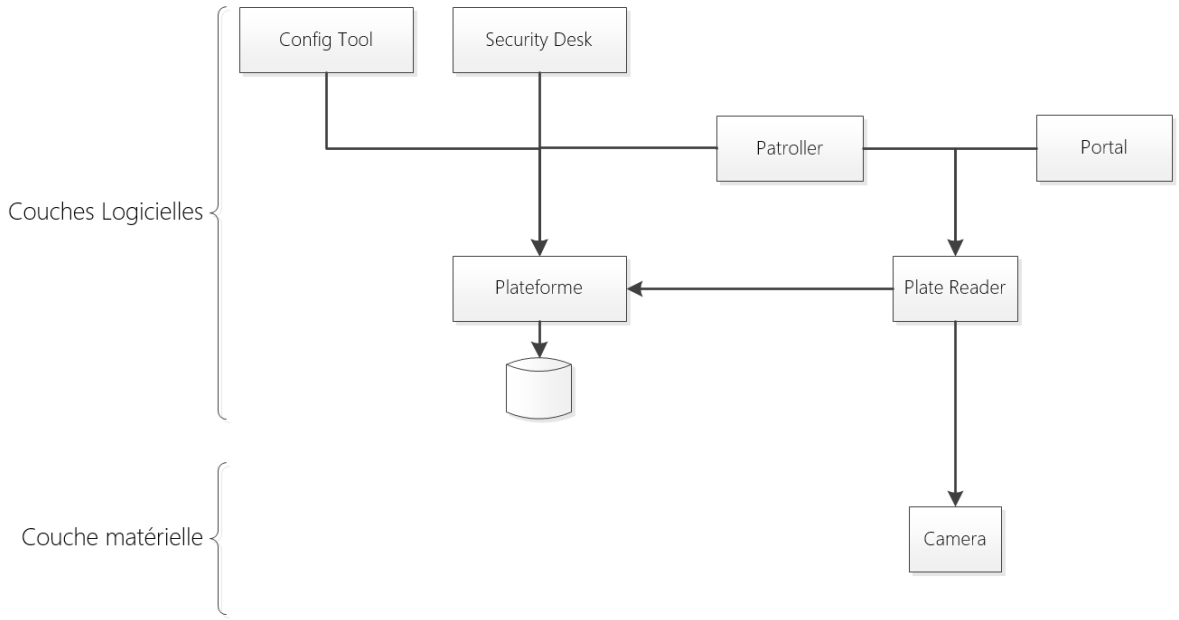

Figure 4 – Arrimage entre la plateforme et le modèle COSMIC

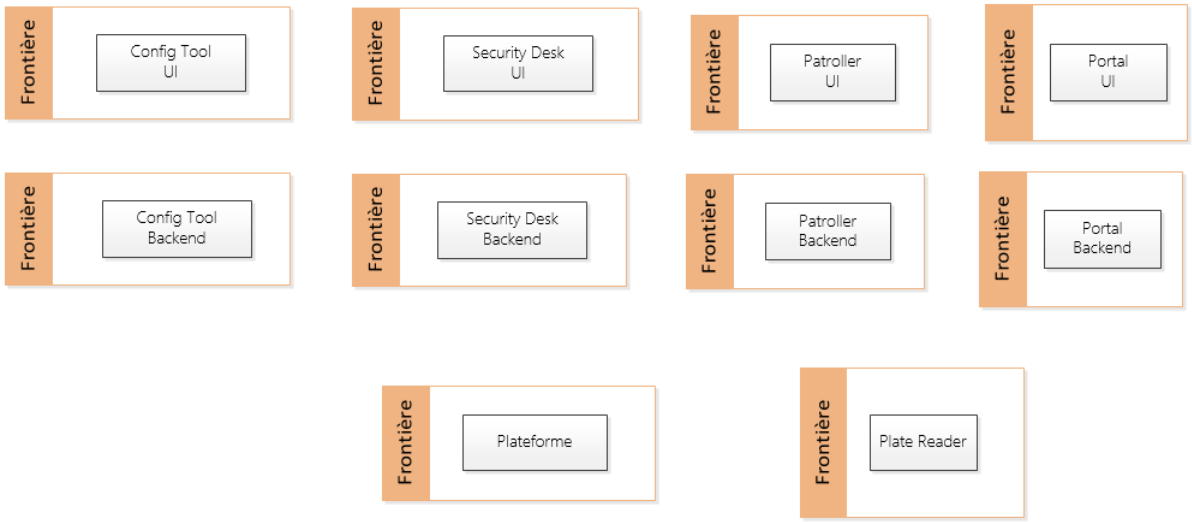

Figure 5 – Frontières logicielles

3.3 Mesures des tailles fonctionnelles des tâches

Les mesures des tailles fonctionnelles des tâches ont été réalisées du mois de mai 2013 jusqu'à la première semaine du mois de juillet 2013 par la seule et même personne. Pour chaque tâche, les composantes de la plateforme qui entraient en ligne de compte ont été définies, de même que les utilisateurs fonctionnels, les processus fonctionnels ainsi que les objets d'intérêts qui transitaient au travers des frontières. Le Tableau 3 présente les résultats de ces mesures de la taille fonctionnelle en unités de mesure COSMIC des 24 tâches.

Après que les mesures des tailles fonctionnelles des tâches de maintenance ont été obtenues, ainsi que les efforts réels correspondants, ces données ont été transmises auprès d'un organisme indépendant de collecte et d'étalonnage de projets logiciels.

3.4 Comparaisons avec des données de l'industrie

3.4.1 Présentation de l'ISBSG

L'ISBSG est un organisme indépendant basé en Australie et qui se charge de maintenir des bases de données de mesures réalisées partout dans le monde. Il permet ainsi de faire des comparaisons objectives de la productivité d'une entreprise (c.-à-d. le ratio entre la taille fonctionnelle d'un projet et l'effort nécessaire pour le compléter). Pour soumettre des données de projet à l'ISBSG, il est nécessaire de remplir un questionnaire comprenant plus d'une centaine de questions permettant à l'organisme d'avoir un aperçu de l'environnement dans lequel le projet a été complété.

Tableau 3 – Mesures COSMIC de la taille fonctionnelle des 24 tâches

Tâches	Processus Fonctionnels	Entrées (CFP)	Sorties (CFP)	Lectures (CFP)	Écritures (CFP)	Total (CFP)
1	15	27	9	28	8	72
2	9	19	10	7	1	37
3	13	13	10	5	3	31
4	4	4	4	3	0	11
5	4	4	2	3	1	10
6	8	8	7	3	3	21
7	6	6	5	7	1	19
8	6	12	4	11	3	30
9	4	4	4	2	2	12
10	7	9	5	7	3	24
11	12	12	12	35	0	59
12	1	1	0	0	2	2
13	4	4	3	5	1	13
14	1	1	1	1	0	3
15	4	4	1	5	3	13
16	3	3	2	9	1	15
17	5	5	2	4	3	14
18	3	3	3	9	0	15
19	2	2	2	1	0	5
20	7	7	3	4	4	18
21	5	5	4	6	1	16
22	6	9	3	3	3	18
23	3	3	3	3	0	9
24	6	7	5	13	1	26

3.4.2 Transmission des données à l'ISBSG

Dans le cadre de cette activité de synthèse, chaque tâche est considérée comme un projet du point de vue de l'ISBSG. La raison est que chaque tâche vient avec ses spécifications, sa phase de conception, d'implémentation, d'intégration et de tests. L'ensemble des 24 projets soumis à l'organisme a reçu la note B : cette note signifie que les informations principales sont présentes, mais que d'autres sont manquantes et donc, que des précautions doivent être prises lors de l'interprétation des données.

En effet, les données transmises à l'ISBSG ne concernent que les activités réalisées par l'équipe de maintenance. L'effort passé à définir les exigences logicielles d'une tâche, l'effort passé par l'équipe de tests à définir les cas de tests et l'exécution de ces tests n'est pas inclus. De même, les données n'incluent pas le temps passé par l'équipe de maintenance pour corriger les défauts trouvés par l'équipe de tests. Le temps consacré au déploiement de la plateforme chez le client n'est également pas inclus. Pour ce dernier point, il sera difficile de calculer ce temps sachant que la même plateforme va être installée (ou mise à jour) pour différents clients; il ne s'agit pas d'une situation dans laquelle il existe un projet par client.

Lorsque les données transmises à l'ISBSG ne comprennent que certaines phases du cycle de développement, l'organisme tient compte des phases incluses et exclues pour normaliser les données pour qu'elles puissent être comparées avec le reste des projets de la base de données.

Dans ces circonstances, on peut en conclure que cette note est amplement suffisante puisque l'organisme s'intéresse au cycle de développement complet alors que l'objectif du travail effectué est d'avoir un aperçu de la productivité de l'équipe de maintenance, et plus précisément, à savoir si sa vélocité pourrait expliquer les variations observées initialement entre l'effort estimé et l'effort réel.

3.4.3 Comparaisons avec les données ISBSG de l'industrie

En retour des données transmises par une organisation, l'ISBSG compare ces données avec celles de sa propre base de données et prépare pour l'organisation un rapport la comparant avec les données pertinentes de l'industrie. Pour faire une comparaison pertinente, l'organisme sélectionne un certain nombre de critères reliés aux caractéristiques de l'organisation. Plus précisément, il en sélectionne deux ayant le plus grand impact sur la productivité. Dans le cas présent, les critères identifiés par ISBSG sont la plateforme de développement utilisée (PC) et le type de langage utilisé (3^{e} génération).

Le Tableau 4 présente les résultats des comparaisons avec l'industrie de toutes les tâches mesurées : la colonne de gauche présente la productivité de chacune des 24 tâches de l'organisation, et la colonne de droite présente le positionnement des projets par rapport aux données pertinentes – et normalisées – de l'ISBSG. Ce positionnement est présenté sous la forme des quartiles : par exemple, un percentile inférieur à 25 % indique que la productivité d'une tâche se situe dans le premier quart des organisations les plus performantes. Pour presque l'ensemble des 24 tâches sélectionnées du Tableau 4, cette productivité se situe en dessous du premier quartile : on peut donc considérer que, généralement, l'équipe de maintenance a de grandes facilités à étendre les fonctionnalités de la plateforme et, plus précisément, les applications dont l'équipe est responsable.

Tel que mentionné précédemment, ces informations brutes peuvent être trompeuses si l'on souhaite tirer des conclusions sur la performance de l'équipe, car, tel que mentionné précédemment, nous n'avons qu'une partie de la réalité. Certes, l'équipe de maintenance semble être capable de répondre aux demandes de changement dans des temps record, mais qu'en est-il de la qualité de cette livraison? Rien n'a été fait dans l'infrastructure actuelle de l'entreprise pour faire ressortir cette information : un défaut particulier trouvé par l'équipe de tests ne peut pas être relié à une tâche complétée par le développeur. En conséquence, il n'est pas possible, pour le moment, de connaître la qualité de ce qui est produit par l'équipe.

Tableau 4 – Productivités normalisées de l'équipe de maintenance

#	Productivité normalisée (heures par CFP)	Percentile
1	3,1	< 25 %
2	3,4	= 25 %
3	2,8	< 25 %
4	3,9	< 50 %
5	2,9	< 25 %
6	3,5	< 50 %
7	2,7	< 25 %
8	1,7	< 25 %
9	2,8	< 25 %
10	3,3	< 25 %
11	1,4	< 10 %
12	3,0	< 25 %
13	3,6	< 50 %
14	3,0	< 25 %
15	3,6	< 50 %
16	3,5	< 50 %
17	3,5	< 50 %
18	3,1	< 25 %
19	5,4	< 50 %
20	2,1	< 25 %
21	2,6	< 25 %
22	2,6	< 25 %
23	2,6	< 25 %
24	2,0	< 25 %

Finalement, à la lumière de ces résultats rien n'indique que la vitesse à laquelle l'équipe implémente les changements soit une source de problème en soi. En effet, ce qui ressort des rapports fournis par l'ISBSG, c'est que la productivité de l'équipe est, comparée aux données d'autres entreprises, excellente.

CHAPITRE 4

MODÈLE D'ESTIMATION DES PROJETS *A POSTERIORI*

4.1 Construction du modèle d'estimation avec une régression linéaire

La Figure 6 présente la régression linéaire entre la taille fonctionnelle mesurée et l'effort réel rapporté par les différents membres de l'équipe – cette régression linéaire correspond alors à un modèle d'estimation entre la taille fonctionnelle et l'effort de maintenance qui lui est associé. On peut constater que le coefficient de détermination R^2 est bien meilleur que celui présenté à la Figure 2 entre l'effort estimé et l'effort réel : la relation est désormais explicable à 78 % au lieu de 32 % initialement. De même, l'amplitude moyenne de l'erreur relative a été améliorée positivement : nous avons maintenant une valeur de 28 % au lieu de 58 %.

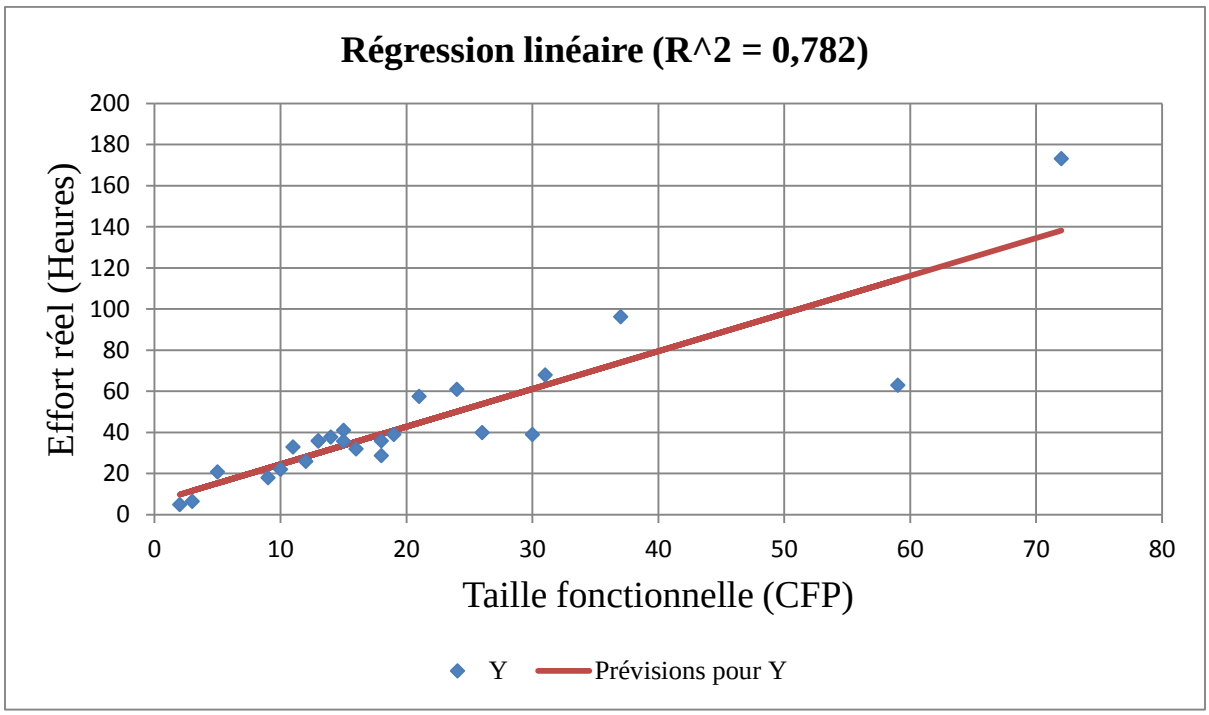

Figure 6 – Régression linéaire entre la taille fonctionnelle et l'effort réel

Avec cette régression linéaire, la qualité prédictive du premier modèle d'estimation basé sur la taille fonctionnelle des demandes de changement peut être calculée. Ainsi, on obtient :

$$Pred(0{,}25) = 67\ \%$$

Cela signifie que l'erreur relative est de moins de 25 % pour 67 % des projets. Le pourcentage monte à 75 % si l'on accepte une erreur relative de moins de 27 %.

$$Pred(0{,}27) = 75\ \%$$

Par comparaison, le modèle défini originalement (Figure 2) présentait une erreur relative de moins de 55 % pour 75 % des projets.

$$Pred(0{,}55) = 75\ \%$$

On a donc amélioré l'estimation en doublant la précision de la prédiction de l'effort nécessaire pour accomplir une tâche de maintenance.

Avec ces résultats, tels quels, il est envisageable d'établir un modèle d'estimation *a posteriori* des tâches de maintenance à accomplir par l'équipe. Ce modèle permettra de faire une estimation pour des tâches de maintenance ayant une taille comprise entre 2 et 72 CFP.

La formule calculant l'effort de programmation en fonction de la taille fonctionnelle est :

$$Effort\ Programmation = 6{,}11 + 1{,}84 \times Taille\ Fonctionnelle$$

Où

$$Coûts\ fixes\ =\ 6{,}11\ \text{heures}$$

Et

$$Coûts\ variables\ =\ 1{,}84\ \text{heures / CFP}$$

4.2 Amélioration du modèle d'estimation

La mesure ayant été réalisée en regardant la description de la fonctionnalité à ajouter et en analysant les modifications effectuées dans le code source, plusieurs observations ont pu être faites durant la mesure. Regardons tout d'abord les productivités de chacune des tâches présentées dans le Tableau 5 afin de voir si le modèle peut être affiné.

Outre le fait que les productivités (estimées et réelles) proches sont plutôt rares (comme c'est le cas pour les projets #4, #5, #15, #21, #22), on peut constater que certaines de ces tâches ont des productivités particulièrement élevées : il s'agit des tâches #8, #11, #20 et #24. Voici quelques commentaires sur certaines de leurs particularités liées à leurs implémentations.

La tâche #8 affiche la deuxième meilleure productivité avec 1,30 heure par CFP. Il s'agissait ici de reproduire une partie de la fonctionnalité développée dans la tâche #1, mais sur un autre type d'entité de la plateforme. Lors de l'implémentation, le développeur a pu profiter d'une logique existante qui a pu être réappliquée sur cette nouvelle entité.

La tâche #11 avec 1,07 heure par CFP possède la meilleure productivité de l'ensemble des tâches mesurées. Les données enregistrées dans la base de données de la plateforme sont toutes sujettes à une expiration pour répondre à des contraintes de confidentialité. La tâche consistait à prendre en compte une valeur indiquant la durée de rétention de certaines de ces données lorsque des requêtes étaient effectuées. Si, d'un point de vue utilisateur, prendre en compte la valeur de rétention pour chacune des requêtes est une nouvelle fonctionnalité en soi, le programmeur n'a eu qu'à insérer le même traitement additionnel pour filtrer les résultats de ces nombreuses requêtes. Les tâches #8 et #11 partagent donc un facteur commun : la duplication de code. Nous pouvons retirer ces deux tâches de notre ensemble puisque les autres

tâches ne partagent pas ce même facteur avantageux et aisément identifiable avant le début d'un projet.

Tableau 5 – Productivités (estimées et réelles) des 24 tâches

#	CFP	Productivité estimée (h/CFP)	Productivité réelle (h/CFP)	Notes
1	72	1,35	2,41	
2	37	4,83	2,60	
3	31	0,90	2,19	
4	11	3,27	3,00	L'équipe a « correctement » estimé un écart de productivité
5	10	2,27	2,20	
6	21	5,24	2,74	
7	19	3,99	2,05	
8	30	2,81	1,30	Beaucoup de copier-coller de logique à partir de #1
9	12	5,12	2,16	
10	24	3,75	2,54	
11	59	0,58	1,07	Beaucoup de copier-coller de logique.
12	2	5,05	2,50	
13	13	8,02	2,77	
14	3	5,24	2,17	
15	13	2,42	2,77	
16	15	5,05	2,73	
17	14	3,47	2,70	
18	15	0,57	2,38	
19	5	5,71	4,15	Cette fonctionnalité implique le développement d'un nouveau contrôle UI
20	18	0,72	1,59	

21	16	2,14	2,00	
22	18	1,83	2,00	
23	9	3,33	2,00	
24	26	2,97	1,54	Cette tâche profite de l'infrastructure existante

La tâche #20 présente une productivité bien meilleure que la majorité des autres tâches, mais les circonstances qui ont permis cette bonne performance ne sont pas connues.

La plateforme possède une gestion des utilisateurs. Pour un certain nombre d'actions pouvant être réalisées par un utilisateur, il existe un privilège associé indiquant si l'utilisateur a le droit ou non d'effectuer l'action en question. Dans le cadre de la tâche #24, il s'agissait de rajouter un privilège sur une action. L'infrastructure de gestion de privilèges était déjà en place et permettait relativement aisément l'ajout de nouveaux privilèges.

Les tâches #21 et #24 ont été gardées, car il apparaît difficile de prévoir le niveau d'extensibilité de la plateforme pour une fonctionnalité donnée. Certaines sont évidentes, d'autres non.

La tâche #19 à 4,19 heures par CFP a été gardée même si elle présente une productivité plus basse. À plus de 4 heures par CFP, il s'agit de la tâche la moins productive parmi celles mesurées. Cette tâche a impliqué le développement d'un nouveau contrôle pour l'interface utilisateur. Malgré tout, comme présenté dans le rapport de l'ISBSG, il s'agit encore d'une tâche avec une excellente productivité par rapport aux données pertinentes de l'industrie.

La Figure 7 présente la nouvelle régression linéaire avec les 22 tâches restantes. On peut constater que le coefficient de détermination R^2 s'en est retrouvé grandement amélioré passant de 0,782 à 0,977. On peut en déduire que le facteur *Duplication de code* était une des raisons contribuant au bruit dans la régression présentée dans la Figure 6. Avec cette nouvelle courbe, on est maintenant en mesure d'expliquer un peu plus de 97 % de la relation. Seulement 3 %

est considéré comme du bruit. L'amplitude moyenne de l'erreur relative s'est améliorée et est passée sous la barre des 20 %, à 16,5 %.

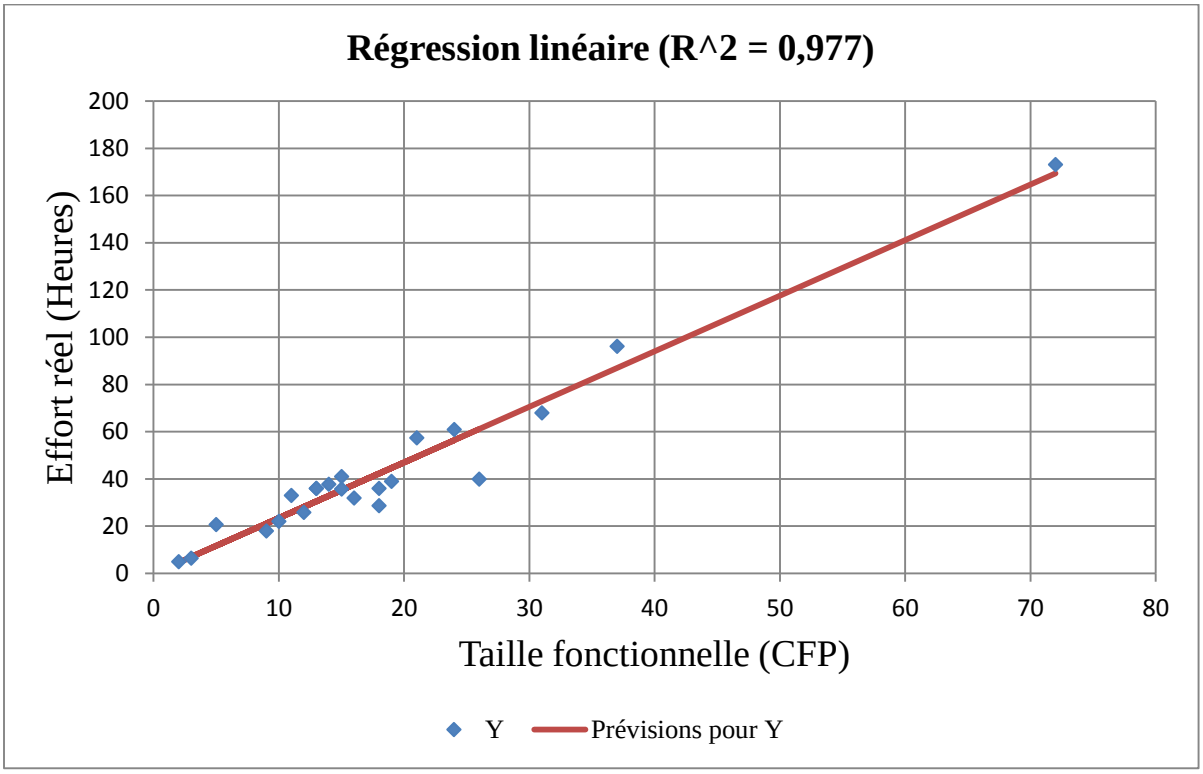

Figure 7 – Régression linéaire des 22 tâches

Bien entendu, la prédictibilité du modèle a aussi été améliorée : on obtient maintenant une erreur relative de moins de 18 % pour 82 % des tâches :

$$Pred(0{,}18) = 82\ \%$$

La formule du modèle d'estimation devient donc :

$$Effort\ Programmation = -0{,}08 + 2{,}35\ \times\ Taille\ Fonctionnelle$$

Où

$$Coûts\ fixes\ =\ -0{,}08\ \text{heure}$$

Et

$$Coûts\ variables\ =\ 2{,}35\ \text{heures / CFP}$$

On constate que le modèle présente un coût fixe négatif or, celui-ci est tellement proche de 0 qu'il peut être considéré comme nul.

4.3 Application *a priori* du modèle d'estimation

Le dernier point de ce chapitre concerne l'utilisation de ce modèle dans un contexte d'estimation *a priori*, c'est-à-dire de faire la mesure d'une tâche de maintenance avant son implémentation et de donner une estimation de l'effort de développement en fonction de la mesure et du modèle d'estimation.

La mesure a été faite au début de l'itération par la même personne qui a fait les mesures précédentes. Cette mesure a été faite en une journée, en même temps que la phase de conception. La tâche à faire correspondait à une amélioration du système existant.

Un avantage supplémentaire de la mesure a été de permettre de faire une vérification manuelle des requis fournis et de constater quelques manques aussi bien dans la demande de changement que dans la proposition de solution. La taille du projet mesuré est de 76 CFP. Puisque la taille fonctionnelle dépasse la borne supérieure du modèle d'estimation, les résultats devront être traités avec précaution.

Avec une taille fonctionnelle de 76 CFP, le modèle d'estimation donne un effort estimé de 178 heures avec une précision du modèle de ±18 %, soit un effort compris en 145 heures et 210 heures. L'estimation qui a été communiquée au début de l'itération était la borne inférieure, soit 145 heures. Normalement, il s'agit de la responsabilité du gestionnaire de choisir la valeur appropriée selon le risque de chaque tâche et de prévoir des plans de contingence. Dans

l'entreprise qui nous intéresse, les développeurs ont la responsabilité de déterminer l'effort nécessaire.

Le développement de la tâche a nécessité un effort réel de 131 heures, soit une productivité de 1,73 heure par CFP.

Historiquement, la tâche qui se rapproche le plus de celle en question est la tâche #1 qui comportait 72 CFP. À l'époque, l'équipe avait estimé 97 heures pour compléter la tâche #1, mais avait finalement eu besoin de 173 heures pour réellement terminer le travail. Il s'agissait d'une erreur de près de 44 %.

Ici, l'effort estimé avec le modèle d'estimation a été surestimé de seulement 10 %. Sachant qu'il s'agit là de la plus grosse tâche estimée et que peu d'estimations ont montré un écart aussi faible (4 tâches ayant des tailles comprises entre 10 et 20 CFP), on peut considérer l'estimation comme particulièrement bonne.

4.4 Mise à jour du modèle d'estimation

Cette nouvelle donnée peut être introduite dans le modèle d'estimation. On obtient ainsi la Figure 8 qui comprend dorénavant 23 points. Le coefficient de détermination R^2 a été impacté à la baisse, passant de 0,977 à 0,962. L'amplitude moyenne de l'erreur relative est passée de 16,5 % à 16,6 %.

La prédictibilité du modèle a également été modifiée en conséquence : on a maintenant une erreur relative de moins de 20 % pour 83 % des projets :

$$Pred(0{,}20) = 83\ \%$$

La formule du modèle devient :

$$Effort\ Programmation = 4{,}93 + 2{,}01 \times Taille\ Fonctionnelle$$

Où

$$Coûts\ fixes = 4{,}93 \text{ heures}$$

Et

$$Coûts\ variables = 2{,}01 \text{ heures / CFP}$$

Les futures estimations devront se baser sur ce nouveau modèle et ainsi, itération après itération, le modèle sera réajusté pour prendre en compte de nouveaux résultats.

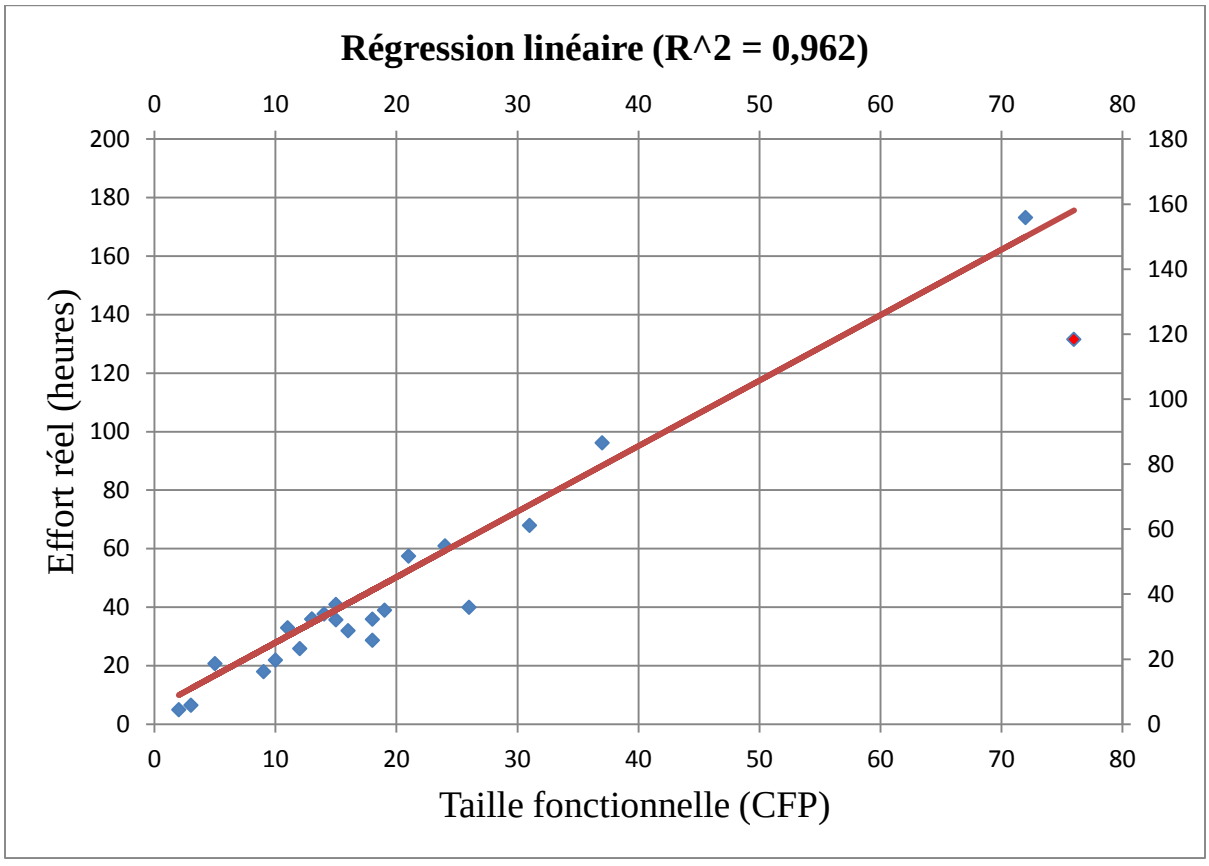

Figure 8 – Intégration du résultat dans le modèle

CONCLUSION

Ce rapport a mis en avant un problème important concernant le processus d'estimation de l'effort requis pour accomplir des tâches dans l'équipe de maintenance. La méthode actuelle ne permet pas de faire des estimations appropriées et ce problème a été constaté avec de nombreuses tâches. Par chance, ces déficiences ne sont pas visibles directement puisqu'il n'existe pas de processus pleinement implémenté pour identifier et contrôler ces problèmes. D'autre part, l'application de la norme COSMIC – ISO 19761 – pour mesurer la taille fonctionnelle des tâches de maintenance a permis de constater que l'équipe de maintenance semble relativement constante dans sa productivité, bien qu'on n'ait aucune information concernant la qualité de ce qui est produit. Avec ces données, on a pu générer un modèle d'estimation avec une régression linéaire offrant une perspective positive et intéressante quant à la possibilité de faire des estimations d'effort plus précises.

L'équipe devrait essayer de s'éloigner du processus actuel d'estimation et de se rapprocher de ce que la méthodologie recommande : on n'estime pas des heures, mais des tailles relatives à la taille d'une tâche de référence. Ensuite, l'effort peut être défini de manière plus précise grâce à la vélocité historique de l'équipe.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

[1] M. Cohn, Agile Estimating and Planning, Prentice Hall, 2005.

[2] A. Abran, Software Estimation Models: Rigor and Innovations, École de Technologie Supérieure, 2012.

[3] ISO, «Software engineering -- COSMIC: a functional size measurement method,» 2011. [En ligne]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=54849. [Accès le 26 03 2014].

[4] COSMIC, «COSMIC Method Measurement Manual,» 2013. [En ligne]. Available: http://www.cosmicon.com/dl_manager4.asp?id=73. [Accès le 05 2013].