h e g

Haute école de gestion
Genève

AGILITE ET MAINTENABILITE

« Les méthodes agiles sont-elles favorables ou non pour la maintenabilité des systèmes ? »

Travail de bachelor réalisé en vue de l'obtention du bachelor HES

par :

Katia MOTA STROPPOLO

Conseiller au travail de bachelor :

Philippe DUGERDIL, Professeur HES, Responsable de la recherche

Genève, le 29 août 2015

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de gestion

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre < Bachelor of Science HES-SO en Informatique de gestion >.

L'étudiante atteste que son travail a été vérifié par un logiciel de détection de plagiat.

L'étudiante accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seule le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 29 août 2015

Katia Mota Stroppolo

Remerciements

J'aimerais commencer par remercier M. Philippe Dugerdil, mon directeur de travail de bachelor. C'était pour moi une immense chance d'avoir partagé ce sujet avec lui : ses conseils, son enthousiasme, ses encouragements, ses critiques toujours bienveillantes et l'apport de son expérience et connaissances sont en grande partie responsables de l'essence même de ce document.

Ensuite, j'aimerais remercier les personnes qui ont participé activement à cette étude en remplissant le questionnaire en ligne et particulièrement à celles qui m'ont ouvert leurs portes pour une interview. Parmi ces personnes, je tiens à remercier fortement M. Frédéric Trémeau, coach agile/Lean chez Altran Suisse. Ma rencontre avec lui m'a ouvert des nouvelles perspectives pour ce travail. De plus, son expérience de l'agilité, du Lean et des méthodologies traditionnelles tout comme sa curiosité à chercher constamment des nouvelles solutions lui confèrent une ouverture d'esprit et une vision de l'agilité qu'il a su généreusement partager avec moi.

Ce qui me mène à remercier Mme Christine Aïdonidis-Flückiger, responsable de l'Urbanisation des SI à l'Etat de Genève et chargée d'enseignement à la HEG. Sans Mme Aïdonidis je n'aurais jamais eu les précieux contacts avec l'entreprise Altran Suisse et la DGSI de l'Etat de Genève. Or, les professionnels contactés dans ces deux entreprises ont sans doute été les plus grands collaborateurs pratiques de cette étude.

Tout au long de cette année j'ai pu compter sur le soutien sans faille de mes proches. Plus que les remercier, j'aimerais exprimer ici que sans eux ce travail n'existerait simplement pas. Du moins pas sous cette forme. Donc merci à mon amie Véronique pour les corrections finales, à mes enfants pour les encouragements et pour la patience qu'ils ont eue envers moi lorsque j'étais prise par la rédaction de ce document.

Pour finir, j'aimerais remercier profondément mon époux, M. Vincent Stroppolo. Sans son soutien inconditionnel, sans son appui logistique, sans ses relectures, sans sa patience, sans les heures incomptables qu'il a passé à s'occuper de nos enfants pendant que je travaillais… bref, sans lui, ce n'est pas simplement ce travail qui n'existerait pas mais tout ce projet de formation sur 4 ans : un vrai projet d'équipe dans notre vie de couple et de famille.

Résumé

Née d'un cours de génie logiciel et tout particulièrement lors de cours sur les méthodologies agiles, l'idée de base de ce travail est de pouvoir analyser la maintenabilité des systèmes ayant été conçus d'après les méthodes agiles et de pouvoir évaluer l'influence que celles-ci peuvent avoir sur l'amélioration ou la détérioration de ladite maintenabilité.

Or, il s'est avéré que répondre à la question « Les méthodes agiles sont-elles favorables ou non pour la maintenabilité des systèmes ? » relevait plus d'un parcours au travers de concepts, expériences et savoir-faire que d'une expérimentation tendant à apporter une simple réponse binaire « oui/non » à une problématique bien plus complexe.

Le long de ce parcours, il s'est avéré que mesurer le temps destiné à maintenir les solutions agiles n'était pas une étape praticable. Les directions ont alors été changées et les mesures ont été remplacées par des interviews, de l'observation et des échanges menant à une réflexion globale sur l'évolution et l'amélioration de la maintenabilité.

Ce travail de recherche fait donc état du parcours bien plus que d'une affirmation ou d'une trouvaille.

Keywords : agile, maintenance, maintenabilité, mesures, périmètre de la maintenance logicielle.

Contexte : ce document a été produit dans le cadre du travail de bachelor « Agilité et maintenabilité », encadré par M. Philippe Dugerdil, professeur à la Haute Ecole de Gestion de Genève, Suisse (HEG).

Il s'agit d'un travail de recherche empirique découpé en trois axes principaux :

A. Revue de littérature et définition du périmètre de l'étude

B. Recueil de la vision du terrain

C. Analyses et conclusion

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

Depuis plusieurs années il est avéré que la maintenance des systèmes informatiques est la plus coûteuse des phases du cycle de vie d'un logiciel : entre 60 et 80% de son coût global.

Dès lors, s'intéresser à la diminution des coûts de maintenance représente un enjeu économique essentiel.

Or, pour citer une enquête du cabinet de conseil Forester, seules 5% des entreprises utilisant une méthode agile s'attendent à une amélioration de la maintenabilité du logiciel développé.

Ces quelques observations nous mènent à nous questionner sur l'évolution de maintenabilité dans le temps, pour les systèmes conçus à partir de méthodes agiles.

Pour tenter de répondre à cette interrogation, ce travail de diplôme est découpé selon les phases suivantes :

- Trouver une définition pratique de la maintenance et ainsi parvenir à l'objectivation d'une mesure qui soit non seulement applicable, mais également utile au business.
- Effectuer une enquête en ligne et recueillir le point de vue du terrain.
- A partir des connaissances acquises dans les phases précédentes, synthétiser les informations, établir des constats et des recommandations.

2. Revue de littérature et définition du périmètre de l'étude

Résumé

Ce chapitre porte, plus précisément, sur le premier axe du travail de recherche - Revue de littérature et définition du périmètre de l'étude - et doit permettre d'atteindre les buts suivants :

1. Dresser un état des lieux sur la recherche dans le domaine de la maintenabilité des systèmes informatiques développés à partir des méthodologies agiles.
2. Sélectionner et analyser des sources littéraires pertinentes.

Guider la proposition d'un Framework méthodologique destiné à mesurer dans le temps, dans un contexte de production en entreprise, la maintenabilité des softwares développés avec des méthodologies agiles.

Il s'agissant d'une étude empirique, cette revue de littérature et définition du périmètre ne sont ni une fin en soi, ni exhaustives. Cette première phase sert à appuyer la recherche et à approfondir les connaissances qui tendent à aboutir sur l'analyse, la conclusion et la défense du travail de bachelor.

2.1 Introduction

La maintenabilité des systèmes informatiques a souvent fait l'objet d'études, d'analyses et de théories. Plusieurs d'entre elles mettent en avant le fait que la maintenabilité peut représenter un pourcentage allant de 60 à 80% des coûts de propriété d'un produit software.

Les lois de Lehman permettent de bien expliciter cette force d'influence de la maintenance dans le cycle de vie des systèmes [1]. Etablies entre 1974 et 1996, ces lois servent de socle pour plusieurs théories qui abordent la maintenabilité d'un logiciel et son évolution.

Parmi celles-ci, une qui touche particulièrement l'évolution de l'effort de maintenance dans le temps est la suivante :

"Increasing Complexity — as an system evolves, its complexity increases unless work is done to maintain or reduce it" [1].

Or, si l'on admet la justesse de cette loi, on peut aisément comprendre l'association entre l'évolution et l'augmentation de la complexité. Il n'en est pas si simple pour sa contrebalance « à moins que du travail soit fait pour maintenir ou réduire cette

complexité » : Quel travail ? En amont ? A la genèse du système ? En urgence ? En priorité ? - Probablement un peu de tout ça et encore d'avantage.

Ces quelques mots d'introduction peuvent nous permettent de dresser les observations suivantes :

- la maintenance d'un système a un coût élevé ;
- pour qu'il soit viable, un système doit évoluer et, ce faisant, sa complexité augmente.

Ces observations peuvent à elles seules expliquer un intérêt certain, du moins économique, pour le développement de solutions maintenables dans le temps et qui suivraient les bonnes pratiques, normes et bon sens adoptés par la communauté informatique au sens large (et dont certaines seront probablement reprises à la fin de ce travail de recherche).

L'avènement des méthodologies agiles et leur essor dans le début des années 2000 ont quelque peu bousculé et mis en cause certaines de ces bonnes pratiques. En effet, de par son manifeste publié en février 2001 [2], le ton est donné pour privilégier – *« donner sa préférence »* – aux individus et à leurs interactions, au logiciel fonctionnel, à la collaboration avec le client et à l'adaptation au changement. Besoins légitimes si l'on tient compte des nombreux échecs des projets logiciels liés à la non-satisfaction du client.

Toutefois, ce manifeste relègue au deuxième plan certaines pratiques pouvant être perçues comme favorisant la maintenabilité future des systèmes tels les processus et les outils, la documentation formelle des caractéristiques fonctionnelles, techniques et de l'architecture du système.

Pour contrer cette mise à mal, de nombreuses recommandations d'application du manifeste ont été développées et sont, plus ou moins, attachées à une ou autre méthodologie agile. Ce calibrage jugé nécessaire par plusieurs auteurs mène à une application « à la carte » dudit manifeste et à des nombreuses variantes de l'agilité. Interprétations qui peuvent favoriser la naissance de solutions peu rigoureuses du point de vue de la qualité puisque l'on peut piocher sur les « bénéfices agiles» tout en s'affranchissant des contraintes nécessaires à fournir un produit possédant une qualité intrinsèque.

Dans le but de guider l'analyse de l'influence de l'agilité sur la maintenabilité, les sections suivants tenteront de répondre aux questions ci-dessous :

Question 1 : Qu'est-ce qu'est la maintenabilité software ? (Q1)

Question 2 : Comment peut-on mesurer la maintenabilité dans le temps et dans un contexte de production en entreprise ? (Q2)

2.2 Comprendre, définir et mesurer la maintenabilité des systèmes informatiques

Définir la maintenabilité constitue un grand défi en soi, dans le sens où un grand flou existe entre sa définition et son périmètre d'application.

C'est pourquoi, nous essayerons de définir la maintenabilité à partir de trois différents points de vue : la littérature, les normes et le business. Cette séparation permettra d'arriver à une définition et un périmètre qui répondront, dans le contexte de cette étude, aux Q1 et Q2, énoncées ci-dessus.

2.2.1 La maintenabilité et la littérature

Cette section présente le détail des démarches effectuées afin de sélectionner et d'analyser les travaux de littérature jugés les plus pertinents pour apporter un éclairage permettant de répondre aux Q1et Q2 d'un point de vue de la littérature.

Pour ce faire, les deux phases de recherche littéraire expliquées ci-dessous ont été employées.

2.2.2 Phase 1 – la sélection des articles

Lors de cette premières phase, des requêtes ont été effectuées sur les sources mises à disposition par la HEG // HES-SO et, en particulier, sur les "Bases de données essentielles en informatique"[1] qui regroupent des sources provenant des bases de données documentaires externes suivantes : ACM Digital library, IEEE Xplore, Inspec, Science Direct (Elsevier), Springer LINK / Kluwer.

Les principaux mots-clés et termes recherchés ont été : agile, maintainability, metrics, empirics, system quality attributes, case study.

Parfois, ces termes ont été combinés pour une plus grande précision dans le retour des résultats de la requête.

[1] http ://www.hesge.ch/heg/infotheque/collections/bases-donnees-documentaires/informatique/bases-donnees-essentielles-en

Après une lecture superficielle des résultats des différentes requêtes, une trentaine d'études a été sélectionnée. Celles-ci ont été lues de manière plus approfondie lors de la phase 2.

> *Premier constat* : il n'existe que très peu[2] d'études de cas concernant la maintenabilité de systèmes développés avec les méthodes agiles. Au mieux, quelques revues de littérature tournant en boucle dans des ensembles fermés et peu nombreux et qui finissent par s'ouvrir sur des références bien plus généralistes concernant l'ingénierie et la qualité logicielle.

2.2.3 Phase 2 – lecture approfondie et classification des articles

Lors de cette deuxième phase, une lecture approfondie et pragmatique des articles sélectionnés précédemment a été effectuée.

Chaque article a été classifié selon les critères suivants :

Excellent : article empirique qui traite l'évaluation de la maintenabilité des solutions conçues avec des méthodes agiles.

Très bien : article sur les mesures qui traitent l'évaluation des attributs qualités en général pour les méthodes agiles, y compris la maintenabilité.

Bien : article qui traite l'évaluation des attributs qualités des méthodes agiles mais ne traite pas directement la maintenabilité. Toutefois, peut y être étendu / appliqué à la maintenabilité.

Suffisant : article sur les mesures des attributs logiciels, ne traite pas directement l'agilité ni parfois la maintenabilité (ou ne traite pas le sujet de manière suffisamment structurée pour être repris dans le TB). Toutefois, celui-ci apporte quelques points de vue intéressants sur le sujet et peut servir de source de réflexion pour le TB.

Insuffisant : article sur les mesures, ne contient pas de points de vue intéressants liés à la maintenabilité des solutions agiles, ni peut y être étendu.

Aucun : article n'ayant aucun intérêt pour le domaine étudié, même s'il traite parfois les mesures agiles (ou OO).

En plus de la classification des articles, d'autres critères ont été enregistrés dans un fichier Excel, il s'agit de :

[2] Moins de 10% des résultats des requêtes.

Numéro : le numéro d'enregistrement de l'article ; *Titre ; Auteur ; Résumé* : abstract de l'article ; *Classification* : cf. classification ci-dessus ; *Remarques* : remarques, passages et chapitres pertinents ; *Biblio* : les références intéressantes mentionnées dans l'étude ; "*+*" : les points positifs ; "*-*" : les points négatifs.

Cette phase d'exploration approfondie de la littérature a permis de sélectionner une vingtaine d'articles supplémentaires.

Au moment de l'écriture de ce rapport, 31 articles ont été examinés et répartis selon le tableau 1.

Excellent	Très bien	Bien	Suffisant	Insuffisant	Aucun
2	5	4	12	5	3

Tableau 1. Classification de la littérature sélectionnée

Deuxième constat : seuls deux articles ont été classés avec le critère d'excellence[3]. Ce qui confirme le premier constat.

Ce deuxième constat confirme la difficulté de pouvoir se référer à des études tournées vers l'expérimentation en entreprise des différents modèles théoriques présentés.

Quelques études ont abordé certaines de ces difficultés et mis en évidence, d'une part, la difficulté de définir la maintenabilité en elle-même et, d'autre part, la difficulté de pouvoir relier ces définitions aux pratiques préconisées par les méthodologies agiles.

D'autres tirent des conclusions sur la maintenabilité basées sur des analyses théoriques et revues de littérature [4] sans pour autant tester leurs conclusions dans le « monde réel » des entreprises.

Malgré tout l'intérêt et la qualité de certaines de ces études, force est de constater que vouloir aborder l'évaluation de la maintenabilité dans une seule optique théorique ou littéraire, rend difficile le lien de ces mesures avec le business et crée de nombreuses mesures qui, dénuées d'un contexte réel, sont rarement utilisables ou productives.

Dans ce sens, ce sont donc des dizaines de mesures abordées, cataloguées, maintes fois expliquées qui se trouvent ainsi noyées dans la théorie. Ces approches sont donc considérées insuffisantes pour cette étude qui vise à investiguer l'évolution de maintenabilité dans de réelles situations d'exploitation de systèmes développés avec les méthodologies agiles.

[3] Excellent : article empirique qui traite l'évaluation de la maintenabilité des solutions conçues avec des méthodes agiles.

Toutefois, certains bons recueils de métriques en font une source d'information utile, à l'instar de [5] qui contient une feuille de route sur les mesures de la maintenabilité y compris un fichier Excel contenant une liste très complète et de [10].

Les deux seuls articles classés « Excellent » sont commentés ci-après.

2.2.3.1 Martin Oberscheven, "Software Quality Assessment in an Agile Environment"

Dans son travail de thèse [9], Oberscheven présente une approche de mesure de la qualité dans un environnement de développement agile basée sur le Framework « Goal-Question-Metric » (GQM). Ce Framework a été proposé par Basili en 1994 [11] et offre l'avantage de relier la mesure à un but de haut niveau directement rattaché au business.

Dans son étude, le modèle est expérimenté dans un contexte réel de développement et un questionnaire permet de recueillir une évaluation subjective de la part du Product Owner et du Scrum Master du projet testé.

Quelques faiblesses de l'étude sont pointées dans le chapitre 6. Parmi elles, la faible quantité de données qui ont servi à calibrer et à réaliser un Benchmark pour interpréter les mesures ainsi que la difficulté à prendre en compte les facteurs influençant la métrique (particulièrement les facteurs RH).

Nous pouvons rajouter la difficulté à généraliser le modèle puisque celui-ci demande, comme prérequis, un ensemble de mesures historiques obtenues en amont avec l'outil « Issue Tracking System » (ITS).

De plus, Oberscheven adopte la définition de la maintenabilité selon ISO et l'applique ensuite dans le cadre d'un développement Agile. En entreprise, cette standardisation selon ISO pourrait conduire à un cadre d'évaluation peu précis et parfois difficile à relier aux influences des méthodologies Agiles dans le projet. Par ailleurs, ce point ressort dans les réponses au questionnaire du Product Owner [9, Appendix B, page 77].

Toutefois, la souplesse de ce modèle, le fait que l'on puisse l'appliquer dans un environnement réel, rattacher les résultats à des buts de l'entreprise et l'adapter à des différentes situations, font de celui-ci une référence de choix dans le cadre de ce travail de bachelor.

2.2.3.2 Dirk Wallerstorfer, « Improving Maintainability with Scrum »

Dans son travail de diplôme [8], Wallerstorfer analyse l'évolution de la maintenance d'un système en production sur une période de transition de sa stratégie de développement. Cette transition se fait par le passage de la méthodologie Waterfall à la méthodologie Scrum. L'étude a mesuré la maintenabilité sur une période de temps allant de 6 mois avant et 12 mois après le changement méthodologique.

Le but étant de pouvoir mesurer et comparer la maintenabilité de ce même système sous Waterfall et Scrum.

Pour ce faire, l'auteur utilise iPlasma, un outil d'analyse qualitative de systèmes orientés objet. Cet outil mesure la qualité du système à l'aide de plusieurs métriques OO connues telles CBO, RFC, WMC et TCC. Une explication succincte de ces métriques est donnée en page 8 du document [8]. D'autres études abordent ces mesures d'une manière plus approfondie [10] [12].

Dans le travail de recherche ici commenté, ces métriques sont comparées d'une méthodologie à l'autre et ces comparaisons, la plupart du temps statistiques, guident les conclusions de l'auteur.

Dans le cadre de ce travail de recherche, cette référence représente une très bonne source de compréhension de la comparaison de méthodologies et ce au travers de la capture de métriques obtenues à partir de l'analyse du code et de l'exécution du système.

Cependant, celui-ci présente la faiblesse de ne produire que des statistiques et des mesures qui peuvent, suivant le contexte, ne pas être représentatives de la maintenabilité d'un point de vue business (car difficilement traduites en business-value) et ainsi être reléguées à un niveau purement indicatif ou prédictif.

De plus, celui-ci compare Waterfall, une méthode pouvant être considérée comme produisant les solutions les plus lourdes et difficiles à maintenir à Scrum. Si l'on remplace Scrum par « quelque chose d'autre » et que l'on compare le « pire » à « quelque chose d'autre » ne peut-on pas obtenir de meilleurs résultats pour le « quelque chose d'autre » ?

Ces réflexions justifient le fait que, malgré sa classification, cette étude ne sera pas reprise dans la suite de ce travail de recherche.

2.3 La maintenabilité et les normes

Dans le cadre difficile de l'interprétation de la maintenabilité, plusieurs études se sont référées aux définitions fournies par des organismes normatifs. Parmi ces définitions, la plus utilisée est celle des normes ISO et standards IEEE, regroupées dans le standard international présenté ci-après.

2.3.1 Définition ISO / IEEE

Selon ISO, la maintenabilité est *« la capacité du logiciel à être modifié après sa livraison. Les modifications peuvent inclure des corrections, améliorations ou adaptations du logiciel à des changements dans l'environnement, et des besoins et spécifications fonctionnelles »* [6] [13].

Le document ISO/IEEE [6] est un standard international qui définit la maintenance et ses activités. La maintenance du logiciel y est décrite comme étant la partie finale du processus de développement de logiciels ou de son «cycle de vie».

Selon ce standard, il existe 4 types de maintenance : corrective, adaptative, perfective et urgente.

La maintenance corrective fixe les bugs découverts après la livraison alors que la maintenance adaptative maintient l'utilisabilité du logiciel dans un environnement modifié ou en changement. Les modifications apportées à un logiciel pour améliorer les performances et la maintenabilité sont classées comme « entretien perfectible ». La maintenance d'urgence consiste à la réalisation de tâches imprévues servant à faire que le système fonctionne correctement et à tout moment.

Tel que défini par la norme IEEE 1219-1998, le processus de maintenance des logiciels comprend 6 phases :

a) Process Implementation.
b) Problem and Modification Analysis.
c) Modification Implementation.
d) Maintenance Review/Acceptance.
e) Migration.
f) Retirement.

Les liens entre ces phases sont illustrés en figure 1

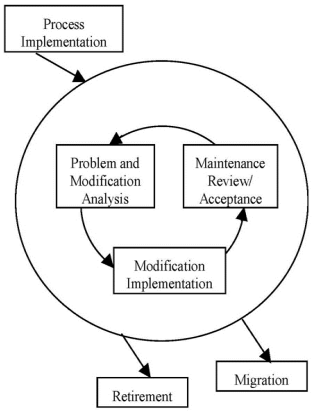


Figure 1 : maintenance process by ISO/IEC 14764

Source : http ://cnx.org/contents/625ae955-8b6a-4070-8174-0369e6a759c8@1/Software-Maintenance

La difficulté à appliquer ces normes dans une situation de développement agile pourrait être le fait qu'agilité et cadre normatif formel ne font pas souvent un mélange facile à appliquer. Cela pourrait même devenir parfois contradictoire dans le sens où les normes et processus viendraient contrer la place donnée à l'individu et ses interactions.

Dès lors, il semblerait opportun de pouvoir arriver à une définition du périmètre de la maintenabilité qui ne soit pas celle d'un cadre stricte normatif.

2.3.2 La maintenabilité et le business

Les études observées relatent une pléiade de métriques vouées à prédire ou à évaluer la qualité d'un système. Toutefois, comme il a été dit dans la section précédente, un nombre minimal de ces études se base sur des expérimentations faites sur le terrain.

Dans ce sens, la littérature présente une lacune en application d'études pratiques pouvant démontrer l'efficacité des modèles et recommandations proposés.

D'autant plus que des menaces bien identifiées dans ces études font état de risques survenant régulièrement dans un environnement réel, tels les facteurs liés aux RH, aux héritages du passé (techniques ou culturelles) et à l'environnement technologique qui ne cesse de changer.

Sans compter que « mesurer pour mesurer » n'apporte rien à la business-value de l'entreprise. Au contraire, une grande quantité de mesures et de termes techniques peut perturber l'aide à la décision managériale et, dans le pire des cas, mener tout l'effort de mesures à l'échec.

A ce sujet, Fenton et Neil [10] présentent la raison principale de cet échec comme étant le fait que la plupart des métriques logicielles ne remplissent pas leur rôle le plus

important qui est celui de fournir des informations qualitatives et quantitatives sur l'état du système. Informations qui permettront ensuite de guider la prise de décisions visant à gérer et à réduire les risques propres au projet mesuré.

Pour ces auteurs, « *l'avenir de la métrique logicielle réside dans l'utilisation de mesures existantes relativement simples servant à construire des outils de gestion et de soutien décisionnel qui combinent différents aspects du développement de logiciels et de tests et qui permettent aux gestionnaires de faire toutes sortes de prédictions, d'évaluations et des compromis pendant le cycle de vie du logiciel* ».

Ce qui nous mène au

> *Troisième constat :* pour qu'il soit utile, un modèle de métrique de la maintenabilité doit être à la fois simple, orienté but business et contribuant à la prise de décisions.

Vient alors la question de définir, au niveau du business, ce que c'est la maintenabilité et le moyen de la mesurer.

Parce que la maintenabilité d'un système est en soi un domaine complexe à définir, comportant plusieurs tâches, voire plusieurs différents métiers au sein d'une équipe IT, les entreprises tendent à mettre dans un même panier tout ce qui touche de loin ou de près à la maintenance et à l'évaluer en termes de pourcentage de budget ou de temps en ressources consommées. La maintenabilité d'un système devient alors un enjeu économique important pour les entreprises.

Capers [3] conclut son étude de 2006 de la manière suivante : « *Dans tous les secteurs la maintenance a tendance à exiger plus de personnel que celles de construction de nouveaux produits. Pour l'industrie du logiciel le nombre de personnel requis pour effectuer la maintenance est inhabituellement grand et attendra bientôt 75% de tous les travailleurs techniques du développement logiciel* ».

Cette conclusion confirme les chiffres avancés dans l'introduction de ce rapport : 60 à 80% des budgets informatiques sont absorbés par des activités de maintenance, ce qui nous permet de dresser le constat suivant :

> *Quatrième constat* : in fine, la maintenabilité d'un point de vue de l'entreprise pourrait revenir à une mesure de ressources consommées en temps pouvant être ensuite traduite en pourcentage consommé de budget.

Ce constat a le mérite d'être simple et de fournir une vue de haut niveau de la mesure de la maintenabilité.

Toutefois, dès que l'on souhaite aborder le thème des coûts de maintenance des logiciels, on s'attaque à un domaine complexe où plusieurs paramètres et difficultés peuvent influencer les résultats.

Une de ces difficultés réside dans le fait que les professionnels impliqués dans les tâches de maintenance le sont souvent aussi impliqués (et en même temps) dans les tâches de développement. Cette difficulté est davantage présente dans certaines pratiques des méthodes agiles. Prenons l'exemple de Scrum où les tâches de maintenance peuvent se trouver mélangées à d'autres dans un backlogg où tout (ou presque) est question d'User Story. Dès lors, il devient difficile de mesurer le temps passé à ne faire « *que* » de la maintenance.

De plus, dans l'urgence, les équipes développant le système et s'occupant également de la maintenance se retrouvent souvent à devoir faire des compromis entre la qualité interne du système et la satisfaction immédiate du client. Ce trade-off dicté par le temps fait que les équipes agiles pourraient être tentées de favoriser systématiquement les « livraisons » en sacrifiant une partie de la qualité. A moins d'avoir des directives bien établies permettant la gestion et la priorisation des tâches liées à la construction d'un système maintenable et qui vise la livraison continue d'un produit de qualité - quitte à y introduire une sous-couche qualité dédiée à l'atteindre ce but qualitatif [7].

Une difficulté supplémentaire est le fait que la maintenance est souvent exprimée en termes d'améliorations ou d'extensions des fonctionnalités métier. Dès lors, pour réaliser ses tâches de maintenance, le développeur doit disposer des informations à l'intérieur et à l'extérieur du système qui le renseigneraient sur l'architecture, le comportement, les fonctionnalités et l'implémentation de celui-ci. Ces informations devraient permettre au développeur d'être capable de faire le lien entre les fonctionnalités métier et le code qui doit être changé ainsi que de prévoir le comportement du système suite au changement.

On se retrouve alors confronté à devoir faire des changements non pas sur un simple ensemble codé d'instructions mais à l'intérieur d'un écosystème influencé par différents facteurs, internes et externes.

Capers montre, dans son article, un nombre de facteurs influençant positivement et négativement la maintenance [3 ; pages 11 et 13]. Ces deux listes proposent non

seulement les facteurs d'amélioration et de dégradation de la maintenabilité du système mais également le pourcentage moyen de l'influence de chacun d'entre eux (figures 2 et 3). Cette classification pourrait servir de socle pour une pondération de la mesure du temps obtenue lors d'expérimentations sur le terrain.

Table 6: Impact of Key Adjustment Factors on Maintenance (Sorted in order of maximum positive impact)

Maintenance Factors	Plus Range
Maintenance specialists	35%
High staff experience	34%
Table-driven variables and data	33%
Low complexity of base code	32%
Y2K and special search engines	30%
Code restructuring tools	29%
Reengineering tools	27%
High level programming languages	25%
Reverse engineering tools	23%
Complexity analysis tools	20%
Defect tracking tools	20%
Y2K "mass update" specialists	20%
Automated change control tools	18%
Unpaid overtime	18%
Quality measurements	16%
Formal base code inspections	15%
Regression test libraries	15%
Excellent response time	12%
Annual training of > 10 days	12%
High management experience	12%
HELP desk automation	12%
No error prone modules	10%
On-line defect reporting	10%
Productivity measurements	8%
Excellent ease of use	7%
User satisfaction measurements	5%
High team morale	5%
Sum	503%

Figure 2 : facteurs d'influence positive

Table 7: Impact of Key Adjustment Factors on Maintenance (Sorted in order of maximum negative impact)

Maintenance Factors	Minus Range
Error prone modules	-50%
Embedded variables and data	-45%
Staff inexperience	-40%
High complexity of base code	-30%
No Y2K of special search engines	-28%
Manual change control methods	-27%
Low level programming languages	-25%
No defect tracking tools	-24%
No Y2K "mass update" specialists	-22%
Poor ease of use	-18%
No quality measurements	-18%
No maintenance specialists	-18%
Poor response time	-16%
Management inexperience	-15%
No base code inspections	-15%
No regression test libraries	-15%
No HELP desk automation	-15%
No on-line defect reporting	-12%
No annual training	-10%
No code restructuring tools	-10%
No reengineering tools	-10%
No reverse engineering tools	-10%
No complexity analysis tools	-10%
No productivity measurements	-7%
Poor team morale	-6%
No user satisfaction measurements	-4%
No unpaid overtime	0%
Sum	-500%

Figure 3 : facteurs d'influence négative

2.4 Maintenabilité et agilité en entreprise : réflexions et propositions

Suite à cette immersion dans la littérature et à la recherche d'une définition de la maintenabilité et de son périmètre d'application, les constats qui ont été faits nous montrent que les questions visées par cette phase ne sont pas simples à répondre.

D'une part, la maintenabilité peut avoir différentes définitions selon le contexte et le point de vue que l'on utilise.

D'autre part, les méthodes agiles, à la différence de méthodes plus conventionnelles, demandent une collaboration intense et une grande proximité des acteurs impliqués dans le projet, y compris le client. D'où une influence certaine et encore plus marquée des différentes cultures d'entreprise et facteurs RH. Sans compter le fait que, seul à partir du manifeste, l'on peut faire plusieurs interprétations, mélanges et applications de différentes méthodologies Agiles.

On se retrouve ainsi avec un cadre méthodologique agile dont les frontières ne sont pas clairement définies et à vouloir mesurer un attribut qualité qui se trouve être également sujet à des grandes interprétations.

L'impact des difficultés est donc doublé et un soin particulier doit être donné à y trouver une définition et un périmètre qui collent au plus près de la réalité du business.

Si l'on admet que la définition de la qualité d'un système est conceptuelle et que sa mesure est opérationnelle, il semblerait nécessaire de garder deux dimensions : une dimension de haut niveau - conceptuelle et qui peut être davantage subjective - et une dimension opérationnelle - plus facilement scénarisable et mesurable.

Reste à savoir comment définir, relier et explorer ces deux dimensions. Dans le cadre de cette étude, une manière de définir ces deux dimensions pourrait être la répartition proposée ci-dessous :

A un haut niveau : la maintenabilité est la propriété que possède un système en production de pouvoir être modifié afin de fournir une plus-value pour l'entreprise et ce à un moindre coût (en temps et en argent) – que ces modifications soient de l'ordre de l'ajout de fonctionnalité, correction de bug, correction préventive ou évolution technologique.

A un niveau opérationnel : la maintenabilité est la propriété que possède un système en production de pouvoir être partiellement modifié par un développeur dans les délais impartis et sans effet de bord sur le reste du système.

La conjonction de ces deux dimensions fournirait à l'entreprise, propriétaire de la solution analysée, des mesures qui lui permettraient de consolider et d'évaluer la moyenne de temps dépensé.

La maintenabilité étant ainsi définie du point de vue de l'entreprise, il resterait à trouver un cadre permettant de la mesurer et d'évaluer son évolution dans le temps. Ce cadre sera théoriquement présenté dans la suite de ce rapport et fera l'objet d'une étude plus approfondie lors de la prochaine étape de ce travail de recherche : « Mise en place de l'étude empirique ».

2.4.1 Les Framework GQM et QAS

Cette section présente deux modèles méthodologiques qui pourraient répondre aux besoins de cette étude de mesure de l'évolution de la maintenabilité : Goal-Question-Metrics (GQM) pour le haut niveau et Quality Attributes Scénario (QAS) pour le niveau opérationnel.

2.4.1.1 GQM : Goal Question Metrics[4]

GQM est un modèle de mesure défini par Basili en 1994 [11]. L'objectif de ce modèle est d'établir un cadre de mesures à trois niveaux permettant de relier les mesures du système à un but business spécifiquement préétabli.

Cet objectif permet d'avoir une approche top-down visant non seulement à sélectionner des mesures ayant un sens pour le business mais aussi de pouvoir retracer le chemin opposé qui va de la mesure jusqu'au goal. Simple à appliquer, ce modèle possède en plus le bénéfice d'éviter tout type de mesure qui resterait dans une bulle, isolée de son contexte business et n'y apportant aucune plus-value.

Les trois niveaux du modèle se définissent ainsi :

Niveau conceptuel : le but (goal)

Le but représente un objectif de management propre au contexte dans lequel il est appliqué et gardant son interprétation à l'intérieur de celui-ci.

Niveau opérationnel : la question (question)

Pour chaque but, un ensemble de questions est articulé afin de spécifier comment ces buts peuvent être atteints.

Niveau quantitatif : la métrique (metric)

Un ensemble de métriques est associé à chaque question avec pour objectif d'y répondre de manière quantitative.

Au premier niveau, Basili décrit l'établissement du but comme une étape critique du succès de son approche et propose un processus en quatre étapes visant à faciliter sa construction :

1. Objectif
2. Problème
3. Objet (processus)
4. Point de vue

Suit alors le deuxième niveau qui consiste à trouver la ou les questions pertinentes qui serviraient à caractériser ce but d'une manière quantifiable. Puis le troisième, les mesures qui permettront de répondre quantitativement à chacune des questions formulées. La figure 4 montre un exemple d'application directe du modèle.

[4] Définitions basées sur [11], [9] et wikipédia (http ://fr.wikipedia.org/wiki/GQM)

Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question	What is the current change request processing speed?	
Metrics	Average cycle time Standard deviation % cases outside of the upper limit	
Question	Is the performance of the process improving?	
Metrics	$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$ Subjective rating of manager's satisfaction	

Figure 4 : Goal/Question/Metric, source [11, page 3]

Une fois ce modèle choisi, il nous faudra sélectionner un moyen approprié de collecter les données des mesures. Celles-ci seront ensuite mappées au modèle puis interprétées d'un point de vue du business. La sélection du modèle de collecte de données est décrite ci-après.

2.4.1.2 QAS : Quality Atributs Scenarios[5]

Dans le cadre de ce travail de recherche, un moyen approprié pour collecter les données au niveau opérationnel pourrait être l'approche basée sur les scénarios du Software Engineering Institute (SEI) [14].

Parce que les attributs qualité sont souvent orthogonaux et invisibles pour l'utilisateur, ceux-ci sont difficiles à exprimer et à mesurer. La maintenabilité est un exemple parfait de cette difficulté.

Ces attributs qualité sont aussi importants à satisfaire que les fonctionnalités car ils peuvent aussi mener à l'échec du projet. Exemple : une application qui satisfait le moindre désir fonctionnel présent de l'utilisateur mais qui ne sera pas maintenable dans le temps, restera figée et ne résistera pas au poids des changements qui ne manqueront pas d'arriver.

Reste le fait que ces attributs qualité sont difficiles à évaluer et à mesurer. Comment dire d'un système qu'il est facilement maintenable ? Comment dire d'un système qu'il est facilement modifiable ?

En 2003, pour pallier à cette difficulté, Bass et al. [14] ont introduit le concept QAS. Le but de cette méthode est de tester et d'exprimer, à l'aide de scénarios, la qualité des attributs non-fonctionnels d'un système.

[5] Définitions basées sur [14] et sur les enseignements du cours « Organisation du développement logiciel », HEG, Professeur Philippe Dugerdil

L'QAS, exposé en figure 5, propose six éléments à spécifier :

1. L'artefact : la partie du système que l'on doit tester
2. Environnement : l'état dans lequel se trouve le système au moment du test
3. Stimulus : l'élément que l'on inflige à artefact par rapport au QA que l'on veut mesurer
4. Source : celui qui inflige le stimulus : un programmeur, un autre système, etc.
5. La réponse : la réaction de l'artefact au stimulus
6. La mesure : la mesure quantifiée du déroulement ou du résultat du scénario

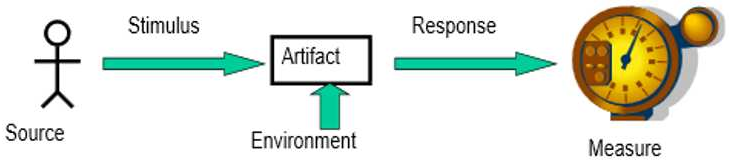


Figure 5 : Les 6 éléments de spécification de l'analyse par scénarios AQS

L'approche par scénarios permet de disposer d'un outil facilement applicable au niveau opérationnel- De plus, celui-ci fournit des réponses *quantifiables* selon un procédé bien structuré (figure 6).

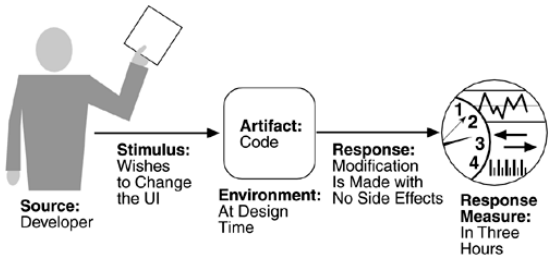


Figure 6 : Exemple d'un scénario « modifiabilité »

Le QAS choisi, Il nous restera l'étape de réinjection de ces mesures dans le modèle GQM et le chemin de retour (botton-up) pour lier et interpréter ces mesures aux questions et buts préétablis.

Pour plus de précision, ces mesures pourraient être pondérées par les facteurs positifs et négatifs proposés en figures 2 et 3. Particulièrement lorsqu'il s'agirait de mesurer différents systèmes.

2.5 Conclusion

La revue de littérature a permis d'appréhender un thème difficile qui est celui de la définition de la maintenabilité et d'une prise de mesures praticables en entreprise.

Compte tenu des nombreuses approches, variations et interprétations que l'on peut trouver à ce sujet, il semblerait nécessaire de ramener celui-ci à un niveau terre-à-terre proche des entreprises et facilement mis en œuvre par celles-ci.

C'est pourquoi, dans le cadre de cette étude, nous proposons deux points de vue business de la maintenabilité : l'un de haut niveau directement lié à un but de l'entreprise, l'autre de niveau opérationnel destiné à mesurer effectivement le temps voué à la maintenabilité à partir de scénarios.

A partir de cette vision de la maintenabilité à deux dimensions, directement liées au point de vue du business, on peut dresser les réponses suivantes aux Q1 et Q2 :

Question 1 : Qu'est-ce qu'est la maintenabilité software?

A un haut niveau *: la maintenabilité est la propriété que possède un système en production de pouvoir être modifié afin de fournir une plus-value pour l'entreprise et, ce, à un moindre coût (en temps et en argent) – que ces modifications soient de l'ordre de l'ajout de fonctionnalité, correction de bug, correction préventive ou évolution technologique.*

A un niveau opérationnel *: la maintenabilité est la propriété que possède un système en production de pouvoir être partiellement modifié par un développeur dans les délais impartis et sans effet de bord sur le reste du système.*

Question 2 : Comment peut-on mesurer la maintenabilité dans le temps et dans un contexte de production en entreprise ? A partir de scénarios QAS au niveau opérationnel qui fourniraient des mesures du temps de maintenance pouvant être reliées à un but de l'entreprise au travers du Framework GQM, tout en tenant compte des facteurs pouvant influencer la mesure (figure 7).

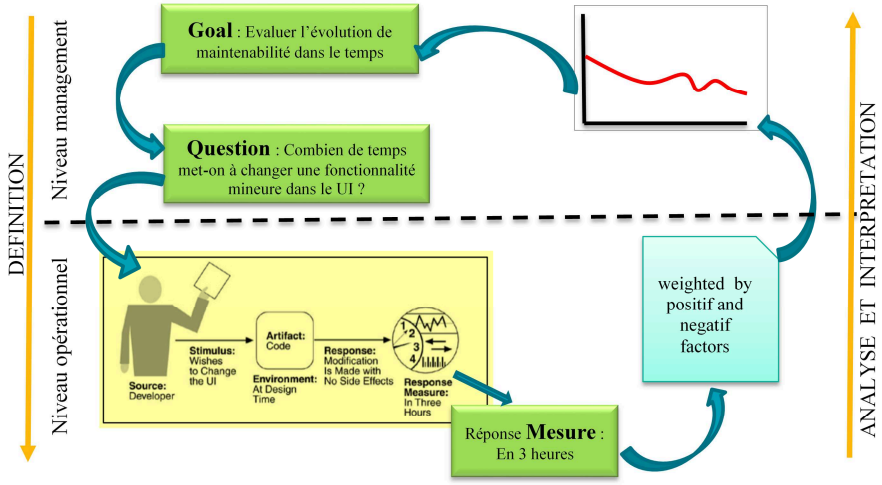


Figure 7 : Mesure de la maintenabilité dans un contexte d'entreprise

2.6 Prochaine étape

Le prochain chapitre comprend la suite de l'étude qui consiste à mener :

- Diffuser et traiter les réponses d'un questionnaire en ligne dont le but est d'établir une analyse subjective du sujet de l'étude.
- Interviewer des acteurs ayant participé à cette enquête afin de mieux comprendre leurs contextes de développement et leurs perceptions de l'influence de l'agilité sur la maintenabilité des systèmes.

Pour finir, la synthèse de ces enquêtes nous permettra d'établir les constats, recommandations et conclusions qui boucleront cette étude.

3. Recueil de la vision du terrain

Résumé

Ce chapitre porte plus précisément sur le deuxième axe du travail de recherche – recueil de la vision du terrain - et doit permettre d'atteindre les buts suivants :

- Créer les outils de diffusion du travail de recherche auprès des sociétés genevoises
- Trouver des entreprises souhaitant collaborer à l'étude
- Recueillir la vision du terrain sur le sujet de recherche

Il s'agit d'une deuxième phase servant à consolider les apports théoriques acquis lors de la première étape, la revue de littérature, et à les confronter aux réalités du terrain.

Pour atteindre ces buts, un flyer [annexe 1] a été envoyé à 353 entreprises du canton de Genève. Cette plaquette présente une vision résumée du problème aperçu pendant les cours de génie logiciel et approfondi lors de la première phase « Revue de littérature et définition du périmètre de l'étude ». De plus, les destinataires étaient invités à remplir une enquête en ligne [annexe 2].

Ensuite, des entretiens ont été menés ainsi qu'une matinée d'observation dans un service d'exploitation.

Ces différentes actions : l'enquête en ligne, les entretiens et la matinée d'observation forment la base de la vision de terrain qui est présentée à la suite de cette étude.

3.1 Introduction

Le lien avec le terrain permet de confronter les notions théoriques apprises lors de la revue de littérature et d'établir la vision pratique qui sera utilisée lors de la conclusion et les recommandations qui bouclent ce travail de recherche.

Le but principal de ce chapitre est de décrire les démarches entreprises pour sortir du contexte purement théorique du chapitre précédent et enrichir ce travail de recherche avec la vision subjective du terrain sur la thématique principale de l'étude : « agilité et maintenabilité ».

C'est pourquoi, nous décrivons ci-après chacune des démarches entreprises pour intégrer le terrain à ce travail de recherche.

3.2 La recherche de partenariat

Une des étapes critiques de ce travail de diplôme a consisté à trouver des entreprises intéressées à collaborer à l'étude.

Il nous a fallu réfléchir à une manière séduisante et concise pour leur présenter le sujet en quelques mots. Pour y parvenir nous avons créé un flyer de présentation. Les raisons de ce choix sont développées dans l'annexe 1.

Une fois les flyers envoyés, nous avons mis en ligne une enquête qui était le point d'entrée pour établir le contact avec quelques 353 entreprises du canton de Genève.

La démarche de réalisation de cette enquête en ligne, les questions et les statistiques globales du sondage sont disponibles à l'annexe 2.

La collecte et le traitement des réponses nous ont permis d'obtenir les contacts de six entreprises intéressées à participer à l'étude.

De plus, nous avons utilisé les informations ainsi obtenues pour établir un profil personnalisé qui a ensuite été repris et developpé dans la première partie des interviews (annexe 3, 1ère partie).

3.2.1 La réalité du terrain et l'impraticabilité de la prise de mesures

Lors de la phase I, la revue de littérature, nous avons été confrontés à la difficulté de définir la maintenabilité des systèmes informatiques tout comme à la difficulté de mesurer cette maintenabilité sur le terrain.

Ces difficultés nous ont menées à chercher et à établir un cadre méthodologique uniformisé et facilement praticable en entreprise. Il s'agit de la combinaison des Framework GQM et QAS présentés au chapitre précédant.

Or, il s'est avéré que la prise de mesures sur le terrain n'était pas praticable. Parmi les causes principales :

- Une des limites de la recherche : « *mesurer le temps, sans avoir le temps* ». En effet, le flyer et l'enquête en ligne ont été finalisés au mois de mars 2015. Il nous restait alors, au mieux, 3 mois de prise de mesures. Ce qui est à l'évidence insuffisant pour mesurer une évolution dans le temps qui solide et pouvant être utilisée pour appuyer des conclusions sur le sujet de l'étude.
- Les entreprises ayant répondu au questionnaire pratiquaient l'agilité depuis peu et estimaient ne pas avoir assez de recul pour pouvoir mesurer, de manière objective, la maintenabilité des solutions ainsi construites.
- La recherche de partenaires était laborieuse et le retour peu conséquent (~8% de taux de retour, soit 27 réponses pour 353 flyers envoyés).

C'est pourquoi, nous avons adapté le schéma préétabli et avons remplacé les prises de mesures par des interviews et une demi-journée d'observation. Ce changement de cap nous aura permis d'établir les constats et analyses présentés au chapitre 4.

3.3 Les interviews et la matinée d'observation

L'étape principale de cette phase de récolte d'information sur le terrain a été la réalisation de 5 entretiens réalisés auprès d'acteurs professionnels concernés par l'agilité. Ces entretiens ont eu lieu entre mai et juin 2015, dans des contextes variés tels qu'une multinationale, des services publics, des prestataires de services, des formateurs et d'analystes. L'annexe 4 contient les résumés de chacun des entretiens menés dans le cadre de ce travail de recherche.

De plus, nous avons réalisé une matinée d'observation dans le service d'exploitation de la Direction générale des systèmes d'information de l'Etat de Genève. Ce service introduit une démarche Lean pour optimiser la visibilité et le flux de leurs tâches. Les principaux bénéfices espérés étant de rendre l'équipe le plus autonome possible, de partager les informations de manière visuelle et de responsabiliser les collaborateurs.

Ces différents contacts avec le terrain nous ont permis de récolter plusieurs messages, expériences et avis au sujet de la maintenabilité des systèmes agiles. Apportant ainsi la vision du terrain qui vient compléter l'aspect théorique emmené par la revue de littérature.

3.4 Prochaine étape

Le prochain chapitre présentera une synthèse des informations obtenues grâce à la revue de littérature et aux informations récoltées sur le terrain. Sous forme d'analyses et constats, cette synthèse nous permettra d'aboutir à la conclusion finale de cette étude.

4. Analyses et constats

Résumé

Les revues de littérature, entretiens et observation relatés dans les chapitres précédents nous permettent d'avoir un point de vue pratique et théorique sur la maintenabilité des systèmes réalisés avec des méthodologies agiles. Les constats et analyses issus de ce rapprochement entre la théorie et le terrain sont à la base des propos avancés dans ce chapitre.

4.1 Introduction

Opter pour des méthodes agiles afin de dynamiser les processus de développement, de faire participer l'utilisateur à la construction et de diminuer le time-to-market semble être un choix qui apporte les bénéfices escomptés aux personnes ayant fait le pas de se lancer dans l'agilité.

Dans le cadre de cette étude, nous avons pu interviewer et observer des équipes évoluant autour de Scrum et Kanban (Lean). Dans chaque cas étudié, l'augmentation de la vitesse, l'adéquation des fonctionnalités produites avec les fonctionnalités attendues par les clients ainsi que la visibilité du travail effectué semblent être des atouts majeurs de l'application des méthodes agiles dans les contextes étudiés.

Ces bénéfices ont souvent été évoqués comme étant des facteurs favorisant la maintenabilité des systèmes. Des avis favorables qui ont également été nuancés par des avis de neutralité par rapport aux méthodologies traditionnelles – c'est-à-dire ni pire ni meilleure – tout comme par quelques craintes. Toutefois, les personnes rencontrées restent subjectivement convaincues que l'agilité, lorsque celle-ci est bien appliquée, aura soit un impact favorable sur la maintenabilité, soit un impact neutre. Aucune de ces personnes n'a fait état de l'observation d'une dégradation de la maintenabilité par rapport aux systèmes construits et maintenus à partir de méthodologies non agiles.

Ce chapitre devrait permettre de mettre en relation les visions subjectives et théoriques du sujet avec des principes, rôles et pratiques agiles favorisant la maintenance. Pour finir, nous aimerions aborder un point de vue économique de l'influence de l'agilité sur la maintenabilité.

Avant d'entamer une telle démarche, il conviendrait de se questionner au sujet de la pertinence de s'interroger sur les effets de l'agilité sur la maintenabilité des systèmes.

4.2 La maintenabilité des systèmes : une affaire de tous ?

Plusieurs enquêtes agiles ont démontré que l'amélioration de la maintenabilité ne se trouve pas dans les principales attentes et bénéfices espérés par les personnes appliquant les méthodologies issues de l'agilité.

Pour ne citer qu'une de ces statistiques, Tom Grant montre dans son article « *Navigate the Future of Agile and Lean* » [21] que seules 5% des entreprises utilisant l'agilité perçoivent une amélioration de la maintenabilité. Les résultats de ce sondage sont reconstitués par la figure 8.

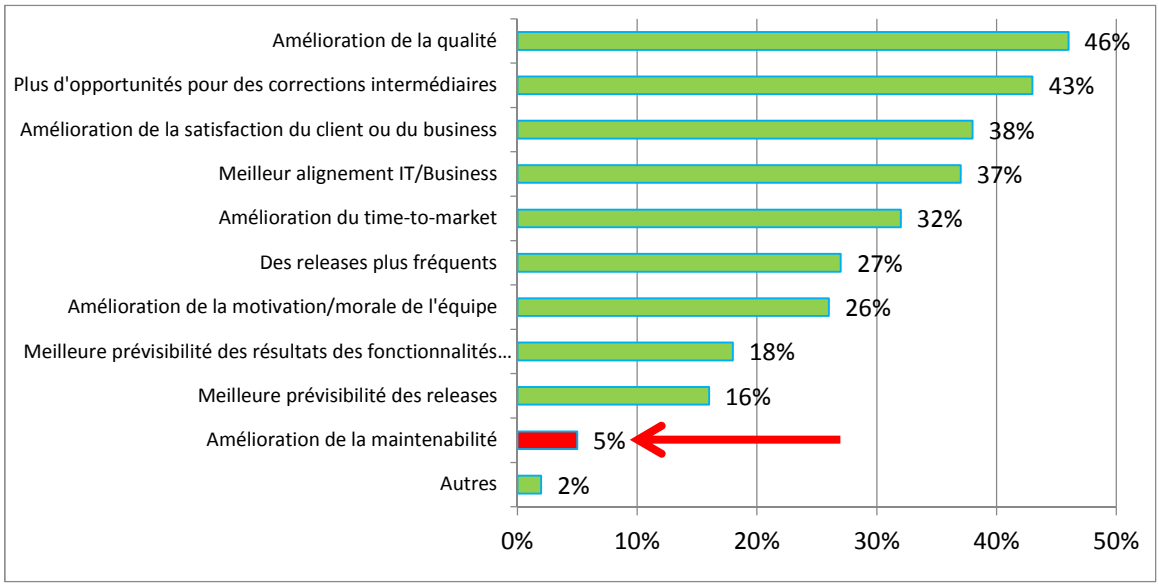


Figure 8 : Bénéfices perçus de l'agilité dans l'organisation.
Figure reconstruite à partir des données publiées par
T. Grant. Forrester Research, 10 Janvier 2012 [21]

Le cas d'utilisation de Japan Tobacco International (JTI) illustre bien une de ces applications de l'agilité où la maintenabilité ne constitue ni une préoccupation ni une attente lorsqu'une entreprise utilise l'agilité.

4.2.1 Le cas pratique de Japan Tobacco International (JTI)

Dans son contexte de développement agile, JTI a choisi de miser sur une seule technologie (Apple) et de développer des applications mobiles non-critiques, uniquement à usage interne. Ce choix place les systèmes développés avec Scrum dans un contexte très peu risqué, tout en permettant de profiter amplement des bénéfices liés à l'accélération du processus de développement et aux interactions avec l'utilisateur final.

Par analogie, les solutions agiles ainsi développées ont été associées à des solutions de type « kleenex » : si les coûts en maintenance devenaient trop élevés, la solution

pourrait être facilement arrêtée et remplacée par une autre, sans que cela n'ait de conséquences importantes pour le business.

Dans ce contexte métier, JTI est un exemple de l'efficacité de Scrum lorsque celui-ci est déployé sur un périmètre bien établi, dans un contexte professionnel et environnemental peu complexe.

Ce cas d'étude nous permet de constater que la maintenabilité des systèmes n'est pas une préoccupation innée de chaque projet de développement informatique. En effet, il existe des solutions – agiles ou non-agiles – qui ne sont tout simplement pas, ou très peu, concernées par une grande charge de tâches de maintenabilité. Des solutions à faible complexité, avec peu (ou pas) de changement de logique métier et peu critiques pour le business.

Ce constat ayant été établi, nous allons nous intéresser, dans les sections suivantes, à l'influence de l'agilité sur des systèmes où les besoins en maintenabilité sont existants.

4.3 Agilité locale et évolution des systèmes d'information

L'interview avec Qim info apporte une vue fortement basée sur l'expérimentation et la pratique de la maintenabilité des solutions agiles. Au fait, malgré la jeunesse de leur application, Qim info utilise l'agilité (Scrum en particulier) d'une manière bien plus large que les autres entreprises interviewées, c'est-à-dire, en interne, en externe, sur des projets agiles et non-agiles.

Cette vision élargie permet à Qim info de se confronter plus rapidement à quelques limites de l'application de la méthodologie Scrum. Une de ces limites apparaîtrait lorsque les personnes se trouvent confrontées à des besoins importants de maintenance destinés à assurer la conformité du produit développé avec le fonctionnement global du système d'information de l'entreprise.

Confrontés à cette limite, on constate alors que certains principes et pratiques agiles, particulièrement présents dans Scrum, sont trop restreints car localisés uniquement au niveau du projet et des équipes de développement. De ce fait, l'application de ces pratiques ne favorise ni la cohérence ni l'amélioration globale du système. Au contraire, cette optimisation locale pourrait favoriser l'apparition de solutions formant un patchwork d'applications. Cette construction des systèmes d'information en silos d'applications pourrait avoir une influence négative sur l'évolution et la maintenabilité du système d'information de l'entreprise (SI).

Dans ce contexte peu favorable à la maintenance, seule une gouvernance globale des SI, de sa trajectoire présente et future ainsi que de son architecture semblerait pouvoir éviter les effets négatifs issus d'une application méthodologique qui ne viserait que la performance locale, au niveau du projet et du développement, au détriment de la performance globale.

4.4 La maintenabilité et l'agilité lorsque celle-ci est *« appliquée correctement »*

Plusieurs autres personnes interviewées ont exprimé le fait qu'appliquer l'agilité dans un contexte plus complexe, ayant un fort besoin en maintenabilité, ne peut pas être réduit à appliquer Scrum, méthodologie perçue comme étant principalement focalisée sur la gestion du projet et de l'équipe agile.

Lors de son interview, M. Lo Giudice, analyste à Forrester Research, a fortement insisté sur l'expression *« lorsque l'agilité est appliquée correctement ».* Ce besoin de renforcer cette expression nous permet de constater le fait que l'agilité, telle que celle-ci a été formalisée par les créateurs du manifeste agile, est un ensemble de principes et non une méthodologie unique.

Dans ce sens, « devenir agile » reviendrait à adopter un ensemble de principes, de pratiques et d'outils tirés de différentes méthodologies portant le label agile. Ce parcours multi-méthodologique pourrait être illustré par la figure 9 qui représente la roadmap des différentes pratiques associées à l'agilité.

Cette diversité de méthodologies, de pratiques et d'outils peut conduire à différents choix qui vont influencer fortement la qualité de la maintenabilité du système.

En effet, le fait de pouvoir choisir parmi un panel de pratiques, tel celui présenté en figure 9, peut conduire à réaliser des solutions agiles qui ne parcourront jamais les embranchements de la roadmap menant à la construction de la qualité interne nécessaire à garantir une maintenabilité dans le temps. Dès lors, même si des pratiques favorisant la maintenabilité y figurent, rien ne garantit que ces pratiques soient appliquées.

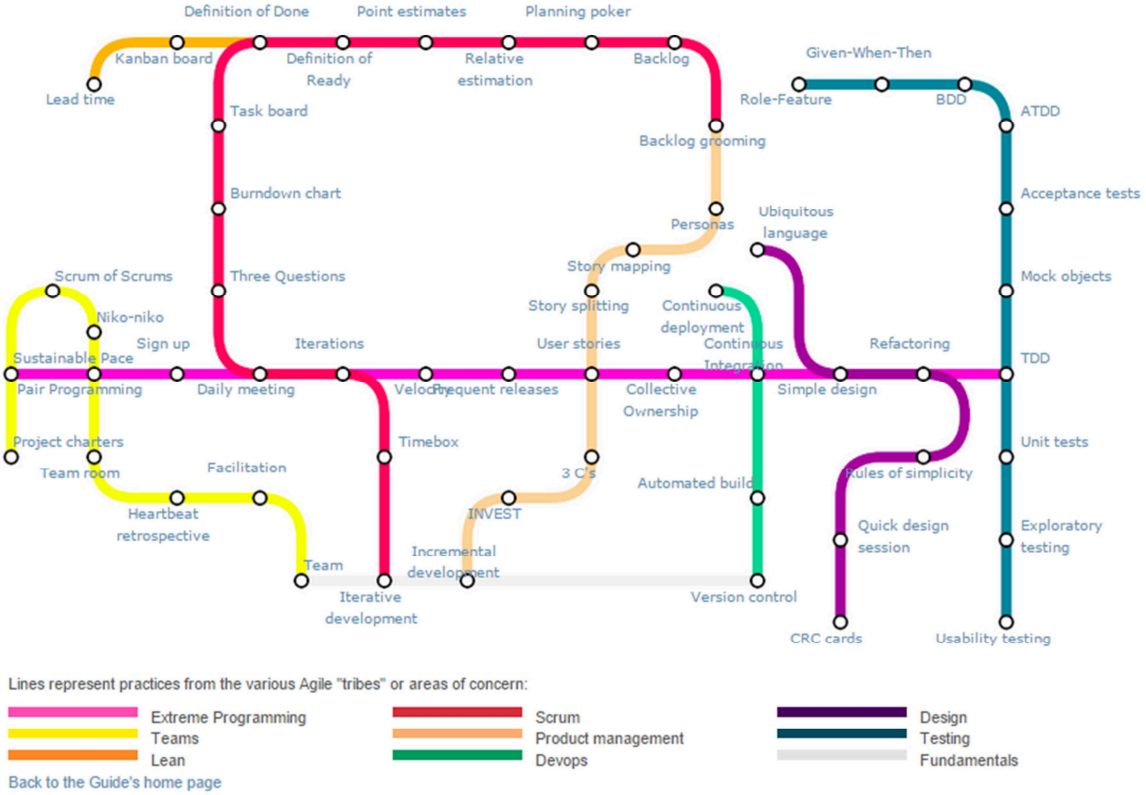


Figure 9 : Roadmap des pratiques agiles
Source : 2013 Agile Alliance and Institut Agile.
http://guide.agilealliance.org/subway.html#sthash.0n2SrqZo.dpuf

C'est pourquoi, dès lors que l'on est concerné par la maintenabilité dans le temps et que l'on développe en agile, il faudrait tenir compte de pratiques proposées par différentes méthodologies.

Pour ce faire, nous proposons, au tableau 2, une correspondance entre les principaux facteurs favorisant la maintenabilité des systèmes et les principes, pratiques et rôles agiles proposés par les méthodologies suivantes : XP, Scrum, Lean SD, DAD et SAFe®. L'annexe 5 décrit le détail des étapes et sources utilisées pour la création de ce tableau.

Tableau 2 : mapping entre les facteurs favorisant la maintenabilité et les principes, rôles et pratiques agiles

Facteurs	Mapping avec les pratique, principes ou rôles agiles	Introduit en agile par la méthodologie
Architecture et design favorisant la maintenabilité	Architecture basée sur des métaphores	XP
	CRC cards	
	Spikes pour les issues complexes	
	Refactoring chaque fois que possible	
	Epiques pour les stories complexes	Scrum
	Intégrité conceptuelle	Lean SD
	Implémenter d'abord les interfaces des systèmes complexes.	
	Décider le plus tard possible	
	Outils de retardement de décision :	
	- L'utilisation de modules	
	- Utilisation d'interfaces	
	- Utilisation de paramètres	
	- Utilisation d'abstraction	
	- Eviter la programmation séquentielle	
	- Choisir les outils sous la base d'expérimentations	
	- DRY	
	- Séparer les responsabilités	
	- Encapsuler les variations : mettre ensemble ce que changera ensemble	
	Utilisation du framework craftsmanship.	
	L'intégrité conceptuelle vue comme condition préalable à l'intégrité perçue.	
	Model-Driven Design pour les systèmes complexes. Collection de modèles :	
	- Conceptual domain model	
	- Glossary.	
	- Use case model.	
	- Qualifiers.	
	Architecture en couche	
	Construire une vision de haut niveau dès le départ, laisser émerger les détails	
	Refactoring régulier pour garder l'architecture saine.	
	Quand un système commence à perdre ces caractéristiques, il est temps de revoir la conception :	
	- Simplicité	
	- Clarté	
	- Aptitude à l'emploi	
	- Pas de répétition	
	Pas de fonctionnalités supplémentaires	
	NFRs à chacun des trois niveaux du Framework	SAFe®
	NFRs vues comme des contraintes sur le backlog	
	Une architectural runway au niveau des programs	
	System Architect	
	Architecture Owner	DAD
	NFRs liste crée pendant la phase d'Inception	

Facteurs	Pratique, principe ou rôle	Introduit en agile par la méthodologie
Faible complexité du code de base	Simplicité	XP
	Éliminer les gaspillages : Passez du temps uniquement sur ce qui ajoute de la valeur pour le client	Lean SD
	Essayer d'anticiper le futur est une source de gaspillage	
	Extra-fonctionnalités : ne coder que ce dont on a besoin « maintenant »	
	Différer la mise en œuvre des capacités futures	
Contrôle et suivi des changements	TDD : Ecriture des codes des Unit test en premier	XP
	Product Backlog listant toutes les fonctionnalités, besoins, améliorations et correctifs correspondant aux changements devant être appliqués au produit lors de livraisons futures	Scrum
	Version Control Tool	Lean SD
Reporting, suivi de l'occurrence et élimination des défauts	Corriger quand s'est cassé	XP
	Quand un bug est trouvé, des tests doivent être créés	
	Jira	Scrum
	Eliminer les défauts par :	Lean SD
	- Tester immédiatement	
	- Intégrer souvent	
	- Livrer dès que possible	
	Feedback-loop plus fréquent en cas de problème constaté	
	Automated build process	
	Build journalier suivi par une batterie de tests automatisés	
	Stop the line	
Intégration continue	Intégration régulière	XP
Partage et acquisition continue de connaissance	Pair programming	XP
	Amplifier l'apprentissage	Lean SD
	Lorsque vous avez des problèmes difficiles, augmentez les feedbacks	
	« Voir grand; Agir petit ; Échouer rapidement ; Apprendre vite »	
Documentation	Tests d'acceptation fonctionnelle et tests unitaires	XP
	N'écrire que de la documentation qui est attendue/utilisée par quelqu'un pour produire de la valeur.	Lean SD
	Documenter les détails juste avant l'itération qui va les implémenter	
	Pour la maintenance : rédiger des documents de synthèse de la conception seulement après avoir terminé le codage - sinon il faudra les écrire à deux reprises	
	Spécialistes au niveau programme et agissant entant que « ressource partagée » (dans une forme d'intervention matricielle)	SAFe®
	Documentation continue (pendant l'itération ou juste après)	DAD
	Liste de risques	

Facteurs	Pratique, principe ou rôle	Introduit en agile par la méthodologie
Vision globale des buts de l'entreprise	Voir l'ensemble	Lean SD
	Se méfier de la tentation d'optimiser les parties au détriment de l'ensemble.	
	Élaborer un modèle économique de chaque application du point de vue du client	
	Master développeur	
	Liste épurée des bonnes et de mauvaises pratiques	
	Mesurer le système à un haut niveau pour optimiser le tout et non les parties	
	Enterprise Architect	SAFe®
	Adoption de normes de codage documentées	XP
Mesure de satisfaction des clients comme indicateur pour évolution / refactoring.	Tests d'acceptation par les utilisateurs (ou leurs représentants) exécutés souvent et leurs score publié	XP
	Revue de Sprint avec les principales parties prenantes	Scrum
	Intégrité perçue	Lean SD
Pratiques d'amélioration continue	code review par pair	XP
	Rétrospective de Sprint : inspection et plan d'amélioration pour le prochain sprint	Scrum
	Map Value Stream	Lean SD
	Liste de bonnes et mauvaises pratiques affinée à la fin de chaque itération	
	Five Whys	
	Identifier votre plus grande contrainte et diriger tous les efforts vers son élimination	
	Déplacez votre attention d'un cran vers le haut et optimisez l'ensemble du système	
Stratégie globale d'amélioration et évolution continue de la qualité du SI	Construire l'intégrité de l'intérieur :	Lean SD
	- Ne pas essayer de traiter l'intégrité lorsque un problème arrive : construisez-là de l'intérieur.	
	Communauté d'experts intégrés au projet, sous forme matricielle	
	GQM	DAD
Management visuel du flux de travail en cours	Visualisation du travail en cours	Scrum
	Kanban	Lean SD
	visualisation des goulots d'étranglement (WIPL)	
Espace de travail adéquat	Espace de travail informatif	XP
	Espace de travail organisé pour éviter/diminuer les déplacements	Lean SD
Critères d'acceptation métier clairs	Notion de « terminé » partagée, affinement du niveau de granularité des items fait en communauté, avant le sprint	Scrum
	Tests fonctionnels comme outil de communication client-développeur	Lean SD

Facteurs	Pratique, principe ou rôle	Introduit en agile par la méthodologie
Comprendre le métier dans sa globalité et en avoir une vision partagée	Un langage commun décrivant le processus doit être partagé par tous les participants	Scrum
	Un système complexe devrait être représenté en utilisant un langage et un ensemble de modèles ayant au minimum les caractéristiques suivantes : - les clients les comprennent - les programmeurs peuvent les utiliser sans avoir à les raffiner	Lean SD
	Document de vision au niveau 1	SAFe®
	Inception phase	DAD
Compréhension du besoin métier	User stories écrites	XP
	Le client est toujours disponible	
	Product backlog prioritisés par le product owner, évolutif et suivant le produit pendant toute sa durée de vie	Scrum
	Map Value Stream	Lean SD
	Boucle de génération de connaissance en présence d'utilisateurs ou leurs représentants	
	Regular prototype milestones	
	Membres du team expérimentés dans le domaine	
	Document de vision au niveau 2	
Expérience élevé de l'équipe	Pair programming	XP
	Membres du team spécialisés dans les technologies critiques	Lean SD
	Master Developpeurs	
	S'assurer que des développeurs expérimentés sont impliqués dans tous les domaines critiques	
	Ne mobiliser les ressources que sur un seul projet à la fois (livrer les plus rapidement possible au travers d'un pipeline)	
	Diminuer les déplacements des artéfacts d'un groupe à un autre	
Tests automatisés	Tout le code doit avoir des Units Tests.	XP
	Tout le code doit passer tous les units tests avant d'être livré.	
	Tests automatisés et exécutés chaque jour comme partie du build	Lean SD
	Construction d'un échafaudage de tests en même temps que l'on construit le programme lui-même	
	Tests de régression automatisés	
	Suivi des tests d'acceptation réussis comme signe de convergence des itérations	
Transition développement / maintenance	Principal - Offrir une suite de tests automatisés avec le code - Modèle de présentation de haut niveau créé à la fin de l'effort de développement initial - Après son développement, création d'un modèle du système « tel que construit » Optionnels - Rendre les développeurs responsables des mises à jour continues et ceci comme une façon de maintenir la mémoire d'un système. - Alternativement, les développeurs et les programmeurs de maintenance peuvent travailler conjointement sur une période de temps pour transférer les connaissances tacites.	Lean SD
	Présence de membres « opérationnels » dans le team pour assurer que l'intégration produit le résultat espéré en environnement de production	SAFe®
	Pratiques et application de principes du framework DevOps	
	End-of-lifecycle testing exécutés sur la suite des tests de régression, idéalement faits par l'équipe de développement et par une équipe de test indépendante	
	Disaster recovery documentation : documentation pour installer le système à partir de zéro	DAD

Cette mise en correspondance nous permet d'observer que, parmi les méthodologies agiles analysées, il existe, en effet, plusieurs facteurs pouvant favoriser la maintenabilité. Toutefois, ces pratiques se trouvent dispersées dans plusieurs méthodologies différentes.

Malgré cette dispersion, on peut constater que XP et Lean SD, semblent introduire et traiter, dès l'origine de la méthodologie, un ensemble plus conséquent de facteurs favorables à la maintenance.

De plus, plusieurs personnes interviewées soulignent le fait que, en pratique, lorsque l'on développe en agile, on se sert régulièrement des pratiques de différentes méthodologies agiles, voire même des pratiques traditionnelles. Créant ainsi des tendances à l'utilisation de méthodologies hybrides de type :

- agile/agile (Scrum/XP)
- agile/lean (Scrum/Kanban),
- agile/traditionnel (Water-Scrum-Fall)
- agile/lean/traditionnel (DAD et SAFe®)

Cette analyse nous permet de conclure que l'agilité « lorsqu'elle est bien pratiquée » pourrait en effet favoriser la maintenabilité mais au prix d'un réel effort pour faire cohabiter différentes méthodologies, pratiques et tendances agiles. Or, cet effort a un coût considérable tant au niveau du business qu'au niveau des méthodologies utilisées pour appliquer l'agilité.

4.5 La maintenabilité agile

Précédemment, nous avons pu constater la nécessité d'avoir recours à une diversité de sources méthodologiques lorsque l'on souhaite favoriser la maintenabilité des systèmes construits à partir de méthodologies agiles. En effet, la « Agilité bien appliquée » mentionnée par plusieurs personnes interviewées, demande bien plus que des rituels de gestion de projets et d'équipe tels que ceux prescrits par Scrum.

En effet, favoriser la maintenabilité agile exige l'application de pratiques qui pourraient ajouter passablement de complexité à certains cadres méthodologiques. Ces pratiques représentent un coût et un challenge important pour les entreprises concernées par des systèmes complexes et ayant de forts besoins en maintenabilité.

Lors des entretiens, deux pratiques ont régulièrement été avancées comme étant particulièrement coûteuses : les tests automatisés et le refactoring. En même temps, ces pratiques ont souvent été évoquées comme garantes de la qualité des systèmes agiles et ainsi associées à la favorisation de la maintenabilité. C'est pourquoi, nous allons les développer ci-après.

4.5.1 Les tests automatisés

Les entreprises interviewées ont toutes mentionné la difficulté de réaliser une couverture suffisante de tests automatisés (à l'exception de JTI). En cause, le coût d'écriture des tests. Pour ces professionnels du terrain, le coût lié à la construction des tests serait *« équivalent au coût de l'écriture du code de la fonctionnalité ».* Ce qui reviendrait à doubler le coût de l'écriture d'un code fonctionnel et ce à chaque fois que l'on souhaiterait appliquer systématiquement des pratiques telles le TDD, les tests unitaires, les tests de régression, les tests d'intégration et les tests d'acceptation.

Or, la garantie de la « *bonne* » qualité du code lors des développements agiles est souvent basée sur les pratiques de tests proposées par ses différentes méthodologies.

4.5.2 Le refactoring

Une autre pratique souvent avancée comme étant garant de la qualité de la maintenabilité agile est le refactoring régulier du code, du design et de l'architecture qui émergent lors des itérations. Or, ce refactoring régulier a un coût directement proportionnel à la taille et à la complexité du système auquel il s'applique.

A ce propos, D. Turk et al. [22], écrivent *«l'hypothèse que le refactoring du code supprime la nécessité de concevoir un système apte à évoluer et à changer ne peut pas s'appliquer à tous les systèmes et en particulier aux systèmes complexes. En cause, le fait que, dans ce type de logiciel, il peut exister des aspects architecturaux critiques qui sont difficiles à changer à cause du rôle essentiel qu'ils jouent dans les services de base offerts par le système».*

Le fait est alors soulevé que dans de tels cas l'effort de la modification de ces aspects critiques peut être très élevé et supérieur à l'effort demandé pour réaliser une anticipation fondée sur des modèles architecturaux qui permettraient l'évolution et le changement du système. De plus, l'existence de ces modèles fournit également un support important aux équipes de maintenance qui pourront s'y référer pour comprendre le système, les relations entre les parties et le lien avec le métier. Ce qui éviterait d'avoir à vérifier des millions de ligne de code pour trouver ces informations.

4.5.3 Utilisation effective des techniques agiles favorisant la maintenance

Ces réflexions à propos des coûts élevés de deux des piliers de la garantie agile, les tests et le refactoring, nous mènent à nous questionner sur l'utilisation effective des techniques agiles favorisant la maintenabilité.

A ce propos, VersionOne[6] conduit, depuis 2006, des larges enquêtes[7] sur l'agilité. Nous y trouvons des statistiques sur l'utilisation de différentes techniques agiles. Parmi ces techniques, nous trouvons des nombreuses pratiques liées à celles présentés dans le tableau 2. Nous avons alors filtré la liste des techniques pour ne garder que celles identifiées précédemment comme favorisant la maintenabilité et avons regroupé les données par année.

Ce travail sur les données de VersionOne nous a permis de construire la figure 10. Nous pouvons y observer l'évolution de l'utilisation des techniques retenues, sur six ans, de 2014 à 2008[8]. Les données utilisées ont été obtenues à partir des rapports [23][24][25][26][27].

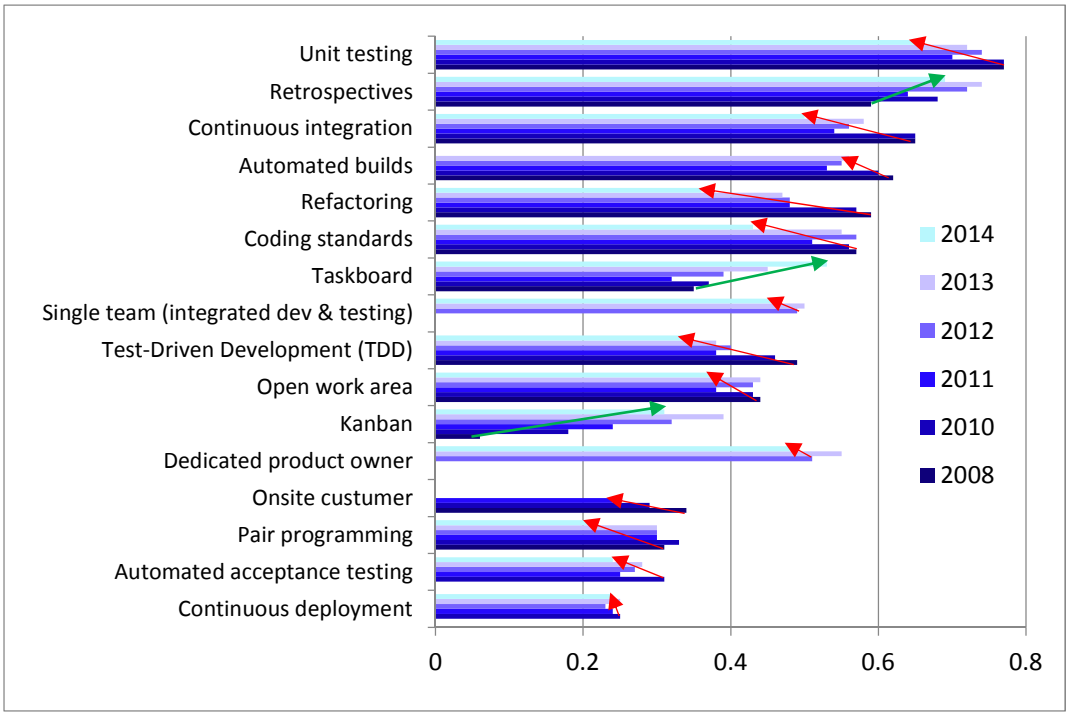


Figure 10 : évolution sur 6 ans de la fréquence d'utilisation des techniques agiles favorisant la maintenabilité[9].

[6] Dont l'actuel *«The 9th Annual State of Agile™ Report»* http ://info.versionone.com/state-of-agile-development-survey-ninth.html

[7] Environ 4'000 participants en 2014

[8] L'année 2009 n'a pas pu être présentée car le rapport original ne contenait pas les données de d'utilisation des techniques agiles (seule une représentation graphique y figurait).

[9] Figure créée sous la base des données des rapports «State of Agile™ Report». VersionOne. http ://www.versionone.com/

Ce graphique nous permet de constater que la fréquence d'utilisation de quasiment toutes les techniques ayant été avancées comme favorables à la maintenabilité des systèmes a connu un recul ces six derniers années, à l'exception des deux techniques de management visuel – Kanban et Taskboard - et des séances de rétrospectives.

4.6 Prochaine étape

Le prochain chapitre de cette étude fera état des conclusions et recommandations bouclant ce travail de recherche.

5. Conclusion

Après plusieurs mois de recherches, d'interviews, de discussions et de réflexions sur le sujet, je suis convaincue qu'aucune réponse absolue et déterministe ne peut être donnée à la question principale posée par cette étude : « Les méthodes agiles sont-elles favorables ou non pour la maintenabilité des systèmes ? » - chaque contexte compte, modifie et influence la réponse.

Toutefois, les théories, cas pratiques et constats mentionnés dans ce document nous permettraient d'esquisser les conclusions suivantes :

Oui

Dans un contexte peu complexe où la pression du time-to-market et la réactivité aux changements est importante.

Dans ce type d'environnement, les développements agiles pourraient favoriser la maintenance. A condition que les solutions restent légères et découplées du système global de l'entreprise.

Le tout en tenant rigoureusement compte des préconisations agiles liées notamment aux tests automatisés - fonctionnels et techniques - et au refactoring. De plus, il serait souhaitable de déléguer la responsabilité de la maintenance aux équipes qui ont construit la solution, selon le principe : « *you did it, you fix it* ».

Non

Dans un contexte plus complexe ou à grande échelle.

Dans ce type d'environnement l'application d'une seule méthodologie agile aurait un impact négatif car celle-ci pourrait s'avérer trop focalisée sur la gestion du projet et le développement.

Toutefois, il faudrait retenir le fait positif que l'agilité au sens large possède des nombreux outils pouvant effectivement favoriser la maintenabilité des systèmes. L'enjeu étant de bien choisir ces outils tout en mesurant l'effort, les coûts et l'efficacité de leurs applications.

Or, dans la plupart des cas, l'agilité, telle qu'elle est appliquée aujourd'hui, ne reflète pas un renforcement de la fréquence d'utilisation de ces techniques pouvant favoriser la maintenance. Au contraire, celles-ci sont plutôt en train de décliner, au profit de pratiques plutôt de management (visuel ou d'équipe).

Aussi, les techniques favorisantes étant reparties sous plusieurs méthodologies, la maîtrise globale du contexte agile serait un facteur de complexité méthodologique non négligeable.

5.1 Avis et recommandations de l'auteur de l'étude

Basée sur les connaissances acquises pendant la réalisation de ce travail de recherche, je pense qu'il est important d'être conscient que les principes agiles diffusés par le Manifeste agile, ne sont probablement pas appropriés pour toutes les organisations ou projets informatiques et ce malgré toutes les « bonnes choses » que ces méthodologies apportent au panorama de développement et de management des systèmes de l'information (SI).

C'est pourquoi, garder une vision globale des SI de l'entreprise, de son projet d'évolution et de ses buts principaux me semble être la meilleure manière d'aborder le choix d'un développement agile si une telle approche est adoptée, ou être confiant dans le choix d'une approche «traditionnelle» ou itérative. L'important étant de faire le choix qui s'adaptera au mieux au contexte présent tout en permettant de faire évoluer et d'améliorer les SI.

Lorsque l'entreprise est concernée, la maintenabilité des systèmes devrait avoir une présence certaine dans cette vision globale du présent tout comme une part importante dans la préparation du meilleur des futurs possibles pour les SI de l'entreprise.

Dans ce sens, et à mon avis, si l'entreprise est intéressée par l'agilité et également concernée par la maintenabilité, une bonne approche serait de se faire accompagner par des professionnels expérimentés. Le meilleur étant de choisir des professionnels sans rapport avec une méthodologie spécifique. Cette indépendance pourrait garantir le fait que le professionnel serait apte à vous proposer les meilleures techniques sans avoir à « vendre une méthode ». Vous pourriez ainsi être accompagné dans votre démarche de changement par une personne capable de vous proposer le meilleur des trois pratiques : agile, itératif ou traditionnel.

Si un tel accompagnement, interne ou externe, n'est pas possible, des combinaisons méthodologiques d'agile, traditionnel et Lean pourraient contrebalancer les possibles effets négatifs de l'agilité sur la maintenabilité de systèmes complexes ou à grande échelle.

Un exemple de cette combinaison est Scaled Agile Framework® (SAFe®) [18] - un framework combinant les meilleures pratiques agiles à des pratiques RUP; le tout

encapsulé par des principes, outils, pratiques et pensées issus du Lean. Disciplined Agile Delivery (DAD) est également un autre exemple de méthodologie hybride.

Il convient de signaler que ces deux méthodologies sont relativement nouvelles (2012 et 2013 respectivement) ce qui ne permet pas d'avoir du recul pour évaluer leurs efficacité. Toutefois, de plus en plus d'entreprises, semblent adopter SAFe® dans un contexte de développement agile à grande échelle, le positionnant ainsi comme un challenger des méthodologies « purement agiles », cf. figure 11.

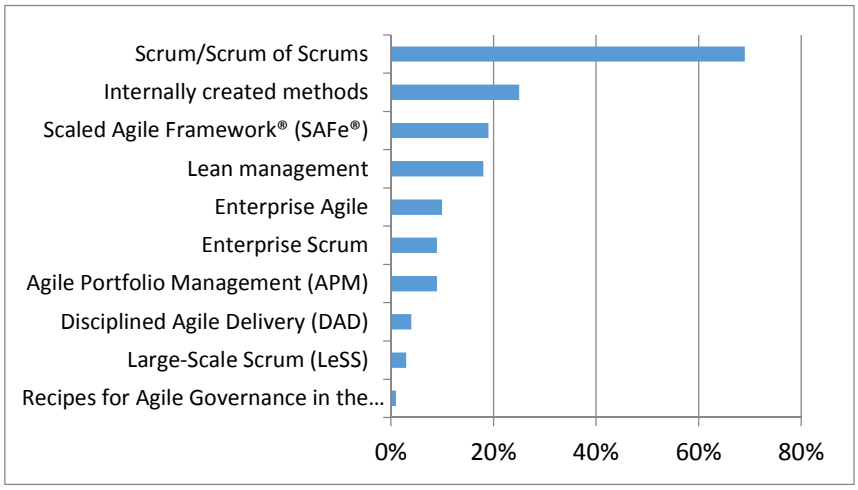


Figure 11 : méthodologies et approches agiles utilisées grande échelle[10]

Cette figure permet également d'observer que le Lean Management trouve sa place dans les contextes de développement agile à grande échelle. Ce fait me permet de conclure cette étude par ma découverte du Lean IT.

Au mois de Mai 2015, ma rencontre avec M. Frédérique Tremeau, coach agile chez Altran, m'a permis d'avoir un autre regard sur l'agilité. Un regard plus ouvert sur des outils de collaboration pouvant être assez puissant comme le Wiki d'entreprise Confluence, des pratiques de priorisation comme le MoSCoW, l'importance d'analyser la culture des entreprises avant de proposer des techniques agiles, etc.

Cette rencontre m'a surtout mené à m'intéresser aux pensées, principes et pratiques Lean et particulièrement le Lean IT. Ce nouvel intérêt m'a permis de découvrir d'autres possibilités pour l'amélioration de la maintenabilité des systèmes.

De plus, j'ai pu observer l'implémentation des premières briques d'amélioration Lean dans le contexte d'un grand service d'exploitation : celui de la Direction générale des systèmes d'information de l'Etat de Genève.

[10] Créée sous la base des données des rapports «State of Agile™ Report». 2015. VersionOne. http ://www.versionone.com/

Le Lean étant une pratique à part entière, au même titre que l'agilité, il y aurait été hors-propos de détailler ces « découvertes » dans ce document. Toutefois, j'ai rajouté à l'annexe 6 quelques principes et pratiques qui m'ont particulièrement marquée et donnée l'envie de m'intéresser d'avantage aux principes et pratiques du Lean dans les SI.

Pour conclure, je pense que le principe du « Go & See » de la démarche Lean est la meilleure manière de vérifier, consolider et faire évoluer les propos de cette étude car aucun document écrit ne peut prétendre être un miroir de la réalité qui est, bien souvent, beaucoup plus complexe que ce que l'on peut y relater.

5.2 Limitations et travail futur

Plusieurs propos tenus dans ce document sont issus d'analyses subjectives de l'influence de la maintenabilité. Dans ce sens, ils ne pourraient être validés que sous la forme de mesures permettant d'évaluer quantitativement et qualitativement l'influence, dans le temps, des méthodologies agiles sur la maintenance des systèmes. Cette validation pourrait se faire par la combinaison des framework GQM et AQS, présenté dans le chapitre 2.

Bibliographie

[1] LEHMAN, M., 1980. On Understanding Laws, Evolution, and Conservation in the Large-Program Life Cycle. Journal of Systems and Software 1 : 213–221, 1980. DOI :10.1016/0164-1212(79)90022-0.

[2] CONBOY, Kieran & FITZGERALD, Brian, 2004. Extreme Programming And Agile Methods - XP/Agile Universe 2004 : 4th Conference On Extreme Programming And Agile Methods, Calgary, Canada, August 15-18, 2004, Proceedings, chap. Toward a Conceptual Framework of Agile Methods. New York : Springer Verlag, 2004. ISBN 354022839X.

[3] CAPERS, Jones, 2006. The economics of software maintenance in the twenty first century. Version 3, February 14, 2006.

[4] KNIPPERS, Daniël, 2011. Agile Software Development and Maintainability. 15thTwente Student Conference on IT, June 20th, 2011.

[5] SARAIVA, Juliana, 2013. A Roadmap for Software Maintainability Measurement. San Francisco, CA, USA : IEEE, ICSE, ACM Student Research Competition, 2013. 978-1-4673-3076-3.

[6] Software Engineering — Software Life Cycle Processes — Maintenance. International standard ISO/IEC 14764 :2006(E) / IEEE Std 14764-2006. Second edition, 2006-09-01.

[7] ULLAH, Malik Imran & ZAIDI, Waqar Ali, 2009. Quality Assurance Activities in Agile - Philosophy to Practice. Master Thesis, Computer Science, School of Computing, Blekinge Institute of Technology. Thesis no : MCS-2009-37. Sweden, September 2009.

[8] WALLERSTORFER, Dirk. Improving Maintainability with Scrum. Fakultät für Informatik der Technischen Universität Wien. Autriche : 2011.

[9] OBERSCHEVEN, Martin. Software Quality Assessment in an Agile Environment. Master's thesis. Radboud University Nijmegen, 2013.

[10] Norman E. Fenton and Martin Neil, 2000. Software metrics : roadmap. In Proceedings of the Conference on The Future of Software Engineering, ICSE '00, pages 357-370. New York, NY, USA : ACM, 2000.

[11] BASILI, Victor R. & CALDIERA Gianluigi & ROMBACH, H. Dieter, 1994. The goal question metric approach. In Encyclopedia of Software Engineering. Wiley, 1994.

[12] LI, Wei & HENRY, Sallie, 1993. Object Oriented Metrics Witch Predicts Maintainability. Journal of Systems and Software - Special issue on object-oriented software, Volume 23 Issue 2, Nov. 1993, Pages 111 – 122. New York, NY, USA, Elsevier Science Inc.

[13] IEEE Std. 610.12-1990. Standard Glossary of Software Engineering Terminology, IEEE Computer Society Press, Los Alamitos, CA, USA : 1993.

[14] BASS, Len & CLEMENTS Paul C. & KAZMAN, Rick, 2003. Software Architecture in Practice (2nd Edition). Addison-Wesley Professional, 2003. ISBN : 0321154959

[15] Extreme programming explained : embrace change. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2000. ISBN :0-201-61641-6

[16] SCHWABER, Ken, 2001. Agile Software Development with Scrum, 20001. Prentice Hall PTR Upper Saddle River, NJ, USA, 2001. ISBN :0130676349

[17] POPPENDIECK, Mary & POPPENDIECK, Tom, 2003. Lean Software Development : An Agile Toolkit. Addison Wesley, 2003. ISBN : 0-321-15078-3

[18] LEFFINGWELL, Dean, 2010. Agile Software Requirements : Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional; Edition : 1, 2010.

[19] AMBLER, Scott & LINES, Mark, 2012. Disciplined Agile Delivery : A Practitioner's Guide to Agile Software Delivery in the Enterprise. Pearson Education Inc. Boston, USA, 2012. ISBN 978-0132810135.

[20] KNIBERG, Henrik, 2007. Scrum et XP depuis les Tranchées. C4Media, Editeur de InfoQ.com, 200. Accessible à http ://www.infoq.com/resource/news/2007/06/scrum-xp-book/en/resources/ScrumAndXpFromTheTrenches_French.pdf

[21] GRANT, Tom - Navigate The Future Of Agile And Lean. Forrester Research. January 10, 2012

[22] Turk Dan, France Robert, Rumpe Bernhard, 2005 (Submitted on 22 Sep 2014). Assumptions Underlying Agile Software Development Processes. Journal of Database Management, Volume 16, No. 4, pp. 62-87, October-December 2005 Idea Group Inc., 2005. Accessible à http ://arxiv.org/abs/1409.6610v1

[23] 9th annual State of Agile™ Survey, 2015. http ://www.versionone.com/

[24] 8th annual State of Agile™ Survey, 2014. http ://www.versionone.com/

[25] 7th annual State of Agile™ Survey, 2013. http ://www.versionone.com/

[26] 6th annual State of Agile™ Survey, 2012. http ://www.versionone.com/

[27] 5th annual State of Agile™ Survey, 2011. http ://www.versionone.com/

[28] 3th annual State of Agile™ Survey, 2009. http ://www.versionone.com/

[29] IGNACE, Marie-Pia & IGNACE, Christian & MEDINA, Régis & CONTAL, Antoine, 2012. La pratique du Lean management dans l'IT, Agilité et amélioration continue. Pearson, France, 2012. ISBN : 978-2-7440-6544-6

[30] PRICE, Bill & JAFFE, David, 2008. The Best Service is No Service : How to Liberate Your Customers from Customer Service, Keep Them Happy, and Control Costs, Jossey-Bass, CA, USA, 2008. ISBN : 978-0-470-18908-5

LES ANNEXES

1. Le flyer comme outil de communication

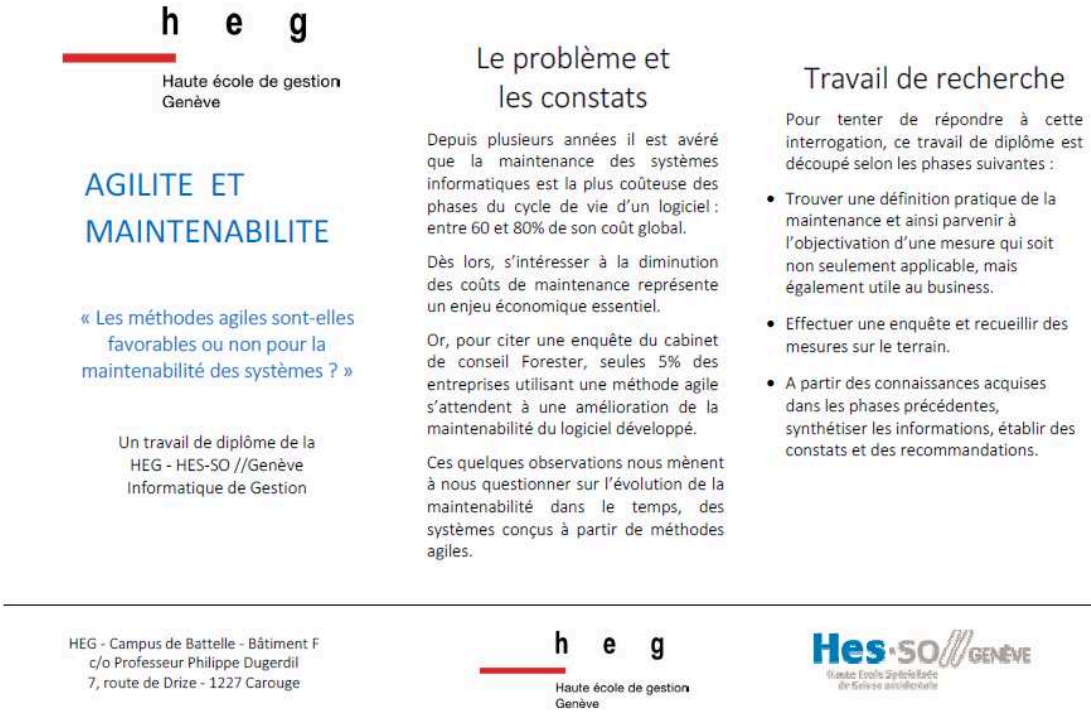
Connu de tous, le flyer est un des plus anciens outils de communication. Sur support papier, celui-ci a une vocation à être distribué à des grands ensembles de personnes et constitue ainsi un outil dit « grand-public ».

Pourquoi avoir choisi un tel outil alors que nous sommes à l'ère des réseaux sociaux et des outils de communication multimédia ? Simplement parce que la pollution des messages inutiles que nous recevions jadis sur nos boîtes aux lettres physiques s'est déplacée sur nos boîtes aux lettres virtuelles.

En effet, les managers des entreprises se trouvent envahis, chaque jour, de mails, news letters, communiqués numériques et autres nouveaux moyens de communication électronique. C'est pourquoi, dans un monde de communication digitale, nous avons supposé que la communication papier, reçue de manière tout-à-fait traditionnelle (par la poste ou en mains-propres) constituerait, paradoxalement, un outil de communication plutôt rare pour les managers IT des entreprises : rare et donc plus apte à attirer l'attention.

Le moyen choisi, il restait à créer le message : une reprise en quelques mots du but et de l'âme de l'étude. Cette phase a été construite en collaboration étroite avec le directeur de mémoire, le professeur Philippe Dugerdil. En effet, de par son expérience du monde de l'entreprise et sa capacité à communiquer et à banaliser des connaissances techniques, M. P. Dugerdil a été un acteur actif dans l'élaboration et l'épuration du message à véhiculer.

Ce document, présenté en figures 12 et 13, a été distribué par voie postale, ou en mains-propres, à 353 acteurs IT du canton de Genève. Par souci de protection de leur anonymat, cette liste n'est pas présentée.


h e g

Haute école de gestion
Genève

AGILITE ET MAINTENABILITE

« Les méthodes agiles sont-elles favorables ou non pour la maintenabilité des systèmes ? »

Un travail de diplôme de la
HEG - HES-SO //Genève
Informatique de Gestion

Le problème et les constats

Depuis plusieurs années il est avéré que la maintenance des systèmes informatiques est la plus coûteuse des phases du cycle de vie d'un logiciel : entre 60 et 80% de son coût global.

Dès lors, s'intéresser à la diminution des coûts de maintenance représente un enjeu économique essentiel.

Or, pour citer une enquête du cabinet de conseil Forester, seules 5% des entreprises utilisant une méthode agile s'attendent à une amélioration de la maintenabilité du logiciel développé.

Ces quelques observations nous mènent à nous questionner sur l'évolution de la maintenabilité dans le temps, des systèmes conçus à partir de méthodes agiles.

Travail de recherche

Pour tenter de répondre à cette interrogation, ce travail de diplôme est découpé selon les phases suivantes :

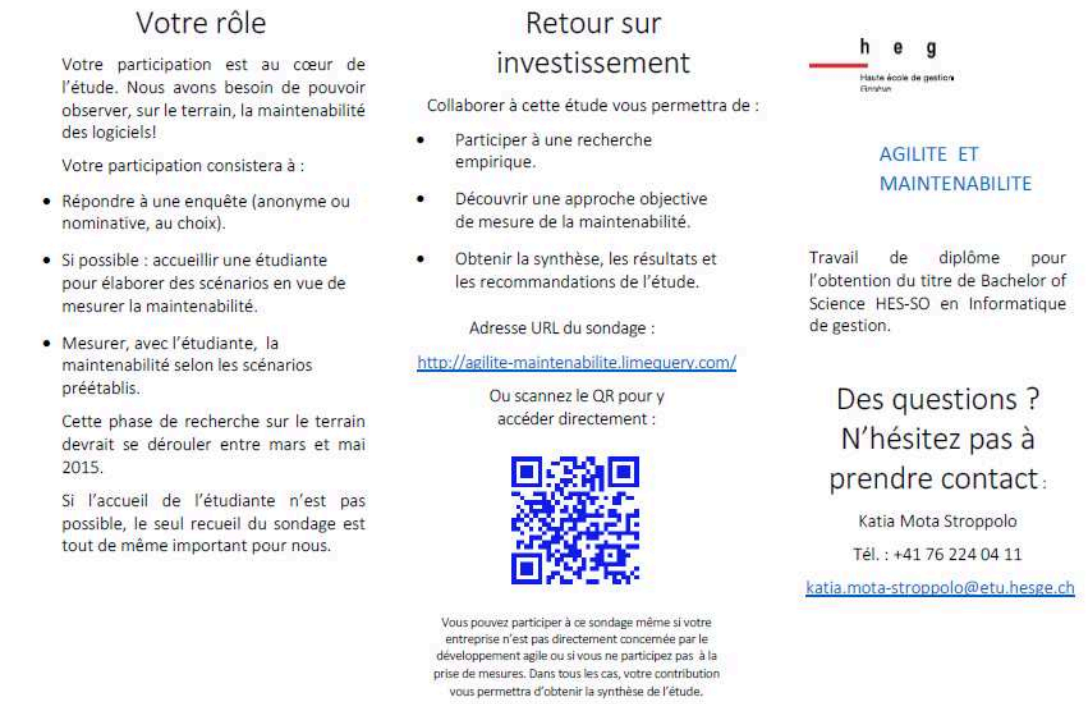
- Trouver une définition pratique de la maintenance et ainsi parvenir à l'objectivation d'une mesure qui soit non seulement applicable, mais également utile au business.
- Effectuer une enquête et recueillir des mesures sur le terrain.
- A partir des connaissances acquises dans les phases précédentes, synthétiser les informations, établir des constats et des recommandations.

HEG - Campus de Battelle - Bâtiment F
c/o Professeur Philippe Dugerdil
7, route de Drize - 1227 Carouge

h e g
Haute école de gestion
Genève

Hes·SO // GENÈVE

Figure 12 : page 1 du flyer de recherche de partenariat


Votre rôle

Votre participation est au cœur de l'étude. Nous avons besoin de pouvoir observer, sur le terrain, la maintenabilité des logiciels!

Votre participation consistera à :

- Répondre à une enquête (anonyme ou nominative, au choix).
- Si possible : accueillir une étudiante pour élaborer des scénarios en vue de mesurer la maintenabilité.
- Mesurer, avec l'étudiante, la maintenabilité selon les scénarios préétablis.

Cette phase de recherche sur le terrain devrait se dérouler entre mars et mai 2015.

Si l'accueil de l'étudiante n'est pas possible, le seul recueil du sondage est tout de même important pour nous.

Retour sur investissement

Collaborer à cette étude vous permettra de :

- Participer à une recherche empirique.
- Découvrir une approche objective de mesure de la maintenabilité.
- Obtenir la synthèse, les résultats et les recommandations de l'étude.

Adresse URL du sondage :

http://agilite-maintenabilite.limequery.com/

Ou scannez le QR pour y accéder directement :

Vous pouvez participer à ce sondage même si votre entreprise n'est pas directement concernée par le développement agile ou si vous ne participez pas à la prise de mesures. Dans tous les cas, votre contribution vous permettra d'obtenir la synthèse de l'étude.

h e g
Haute école de gestion
Genève

AGILITE ET MAINTENABILITE

Travail de diplôme pour l'obtention du titre de Bachelor of Science HES-SO en Informatique de gestion.

Des questions ? N'hésitez pas à prendre contact :

Katia Mota Stroppolo

Tél. : +41 76 224 04 11

katia.mota-stroppolo@etu.hesge.ch

Figure 13 : page 2 du flyer de recherche de partenariat

2. Le questionnaire en ligne

Lors de la revue de littérature, nous avons pu observer que plusieurs travaux de recherche empirique se servaient de questionnaires pour leurs analyses. Plusieurs d'entre eux se limitaient même à cet outil pour baser leurs conclusions.

Dans le cadre de cette étude, le questionnaire trouve sa place et sa valeur dans l'établissement du contexte de l'entreprise, de la manière dont celle-ci aborde l'agilité.

Pour déployer ce questionnaire nous avons étudié 4 outils, cf. résumés ci-après :

SurveyMonkey

Un des outils les plus populaires, SurveyMonkey offre une grande palette de fonctionnalités et l'extraction des données récoltées est assez user-friendly. De plus, nous pouvons trouver sur leur site des modèles de sondage. Il est également possible de créer des questions conditionnelles. Limitation : la version gratuite est limitée à 10 questions et à 100 réponses.

Google form

Accessible depuis un compte Google, cet outil est fait pour interfacer avec Google Drive. Les réponses peuvent être facilement stockées sur une feuille de calcul spreadsheet (équivalent d'Excel pour Google docs). Inconvénient : est accessible soit par un compte Google soit par un site internet personnel.

Lime Survey

Disponible sous licence libre, Lime Survey peut être utilisé gratuitement et sans limitation. La documentation est complète, on y trouve un sondage vitrine proposé en démo et il supporte les questions conditionnelles. De plus, le site propose un hébergement gratuit pour le sondage. Bonus : l'utilisateur peut enregistrer son sondage s'il souhaite répondre en plusieurs étapes.

Google Consumer Surveys

Outil propriétaire et donc payant.

Le choix a été porté sur Lime Survey car les fonctionnalités proposées par cet outil répondaient à tous nos besoins. De plus, celui-ci est distribué sous licence libre; le site propose un serveur d'hébergement pour le sondage et est supporté par une communauté importante, en français !

Les questions et le rendu final peuvent être consultés à l'adresse suivante :

http ://agilite-maintenabilite.limequery.com/

Ci-dessous, vous trouverez une version imprimable des questions qui ont été posées aux entreprises :

2.1 Version imprimable du questionnaire en ligne

Agilité et maintenabilité : un travail de recherche en entreprise

Ce sondage permet d'établir le contexte des entreprises, les pratiques agiles et les attentes business vis-à-vis de la maintenabilité des systèmes informatisés maintenus par les sociétés collaborant au travail de diplôme réalisé par Katia Mota Stroppolo, étudiante en informatique de gestion à la Haute Ecole de Gestion de Genève (HEG // HES-SO Genève).

Un travail de recherche encadré par le professeur Philippe Dugerdil, responsable de la recherche et professeur de Génie logicielle à la HEG // HES-SO Genève.

Lien vers : plaquette "Agilité et Maintenabilité"

Bienvenu sur ce sondage.

Dans sa globalité, celui-ci devrait vous prendre environ 10-15 minutes.

N'hésitez pas à me contacter si vous avez besoin de support, si vous avez des questions, des propositions d'amélioration ou toute autre remarque à me transmettre.

katia.mota-stroppolo@etu.hesge.ch

Bon sondage!

Il y a 17 questions dans ce questionnaire

2.1.1 Contexte de l'entreprise

[] Quel est votre secteur d'activité ? *

Veuillez choisir toutes les réponses qui conviennent :

☐ Assurances
☐ Asset Management
☐ Banques
☐ Cleantech
☐ Commerce de détail et biens de consommation
☐ Energie, approvisionnement & industrie minière
☐ Immeubles
☐ Industrie chimique
☐ Industrie manufacturière
☐ Life Sciences
☐ Secteur de la santé
☐ Secteur public
☐ Services aux entreprises
☐ Technologie, télécommunications, information-communication et médias
☐ Tourisme
☐ Transport et logistique
☐ Autre

[] Quelle est la taille de votre entreprise ? *

Veuillez sélectionner une seule des propositions suivantes :

○ Micro-entreprises (jusqu'à 9)

○ Petites entreprises (10-49)

○ Moyennes entreprises (50-249)

○ Grandes entreprises (250 et plus)

Taille de l'entreprise selon le nombre de postes éq. Plein-temps

[] Quel est votre rôle principal dans l'entreprise ?

Veuillez choisir toutes les réponses qui conviennent :

☐ Gestionnaire de projet
☐ Programmeur/Developpeur/Team
☐ Executive Manager
☐ Manager IT
☐ Manager hors-IT
☐ Business Analyst
☐ Autre

[] Développez-vous ou maintenez-vous des applications spécifiquement créées pour vous ? *

Veuillez sélectionner une seule des propositions suivantes :

○ Oui

○ Non

OUI : si votre entreprise crée ou maintien des programmes développés spécifiquement pour votre business

[] Utilisez-vous des méthodes de développement agile ?

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui' à la question '4 [Q3]' (Développez-vous ou maintenez-vous des applications spécifiquement créées pour vous ?)

Veuillez sélectionner une seule des propositions suivantes :

○ Oui, exclusivement

○ Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)

○ Non

[] La méthodologie utilisée dépend-elle de la durée de vie attendue de vos applications? *

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)' à la question '5 [Q4]' (Utilisez-vous des méthodes de développement agile ?)

Veuillez sélectionner une seule des propositions suivantes :

○ Oui

○ Non

C'est à dire que pour les applications non-pérennes vous utilisez une autre méthode que pour les applications pérennes

[] En fonction de la durée de vie de vos systèmes, quelle méthodologie de développement utilisez-vous ? *

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui' à la question '6 [Q19]' (La méthodologie utilisée dépend-elle de la durée de vie attendue de vos applications ?)

Choisissez la réponse appropriée pour chaque élément :

	Agile	Hybride (Agile + non-agile)	Non-agile	Aucune
< 1 an	○	○	○	○

	Agile	Hybride (Agile + non-agile)	Non-agile	Aucune
Entre 1 et < 3 ans	◯	◯	◯	◯
Entre 3 et 5 ans	◯	◯	◯	◯
Plus de 5 ans	◯	◯	◯	◯

2.1.2 Pratiques agile

Cette section a pour but de relever les méthodologies et pratiques agiles utilisées par les entreprises participant à l'étude.

[] Quelle méthode agile utilisez-vous ? *

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)' *ou* 'Oui, exclusivement' à la question '5 [Q4]' (Utilisez-vous des méthodes de développement agile ?)

Veuillez choisir toutes les réponses qui conviennent :

☐ Agile Unified Process (AUP)
☐ Crystal Clear Methods (Crystal Clear)
☐ Disciplined Agile Delivery (DAD)
☐ Dynamic Systems Development Method (DSDM)
☐ Extreme Programming (XP)
☐ Feature Driven Development (FDD)
☐ Kanban (development)
☐ Lean software development
☐ Rapid Application Development (RAD)
☐ Scrum
☐ Autres

[] A quelle fréquence utilisez-vous les pratiques agiles suivantes :

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui, exclusivement' *ou* 'Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)' à la question '5 [Q4]' (Utilisez-vous des méthodes de développement agile ?)

Choisissez la réponse appropriée pour chaque élément :

	Quotidienne	Hebdomadaire	Mensuelle	Rarement	Jamais
Feedback constant du client ou du Product Owner					
Révision régulière du Product Backlog					
Intégration continue					
Standup Meeting					
Revue d'itération					
Itération Planning					
Burndown Chart					
Livraison Continue					
Test First Programming					
Itérations courtes (2-3 semaines)					
Pair Programming					

[] Depuis combien de temps utilisez-vous des méthodes agiles de développement ?

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui, exclusivement' *ou* 'Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)' à la question '5 [Q4]' (Utilisez-vous des méthodes de développement agile ?)

Veuillez sélectionner une seule des propositions suivantes :

○ < 1 an
○ entre 1 et < 3 ans
○ entre 3 et < 5 ans
○ 5 ans et plus

2.1.3 Perception et attentes vis-à-vis de la maintenabilité

Ce groupe de questions vise à esquisser la perception de la maintenabilité et les attentes business concernant la maintenabilité de ses programmes informatiques.

[] Mesurez-vous le temps consacré à la maintenance de vos solutions informatiques ?

Veuillez sélectionner une seule des propositions suivantes :

○ Oui
○ Non

[] Est-ce que vos développeurs s'occupent eux-mêmes de la maintenance des systèmes qu'ils développent ?

Veuillez sélectionner une seule des propositions suivantes :

○ Toujours
○ Souvent
○ Parfois
○ Jamais

[] Avez-vous constaté une amélioration ou une dégradation de la durée de maintenance en fonction de la méthodologie utilisée ?

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui' à la question '4 [Q3]' (Développez-vous ou maintenez-vous des applications spécifiquement créées pour vous ?) *et* La réponse était 'Oui, exclusivement' *ou* 'Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)' à la question '5 [Q4]' (Utilisez-vous des méthodes de développement agile ?)

Veuillez sélectionner une seule des propositions suivantes :

○ Oui, plus rapide avec agile que traditionnel
○ Oui, plus lente avec agile que traditionnel
○ Non, aucune différence notoire entre agile et traditionnel

[] A quoi attribuez-vous la dégradation de la maintenabilité ?

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui, plus lente avec agile que traditionnel' à la question '13 [Q10]' (Avez-vous constaté une amélioration ou une dégradation de la durée de maintenance en fonction de la méthodologie utilisée ?)

Veuillez choisir toutes les réponses qui conviennent :

☐ Documentation plus succincte
☐ Pas de chef de projet
☐ Pas de modélisation de l'architecture de la solution
☐ Pas de spécification détaillée
☐ Code plus dificile à comprendre
☐ Manque d'uniformité dans le développement (entre différents teams)
☐ Autre :

[] Seriez-vous intéressé à participer à la suite de l'étude qui consisterait à :

•Accueillir une étudiante pour élaborer des scénarios en vue de mesurer l'évolution du temps consacré à la maintenance.
• Mesurer la maintenabilité selon les scénarios préétablis.

Répondre à cette question seulement si les conditions suivantes sont réunies :
La réponse était 'Oui, exclusivement' *ou* 'Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)' à la question '5 [Q4]' (Utilisez-vous des méthodes de développement agile ?)

Ajoutez un commentaire seulement si vous sélectionnez la réponse.

Veuillez choisir toutes les réponses qui conviennent et laissez un commentaire :

☐ Oui
☐ Non
☐ Contactez-moi pour plus de détails :

[] N'hésitez pas à me laisser vos coordonnées si vous souhaitez :

Ajoutez un commentaire seulement si vous sélectionnez la réponse.
Veuillez choisir toutes les réponses qui conviennent et laissez un commentaire :

☐ Recevoir un exemplaire des résultats de l'étude
☐ Avez des questions
☐ Participer à la prise de mesure de la maintenabilité
Autre :

[] Vos coordonnées (entreprise, personne de contact, mail ou téléphone) :

Veuillez écrire votre réponse ici :

Merci d'avoir participé à ce sondage!

Katia Mota Stroppolo

+41 76 224 04 11
Envoyer votre questionnaire.

2.2 Les statistiques du questionnaire on-line

Nombre d'enregistrement(s) pour cette requête :	**27**
Nombre total d'enregistrements pour ce questionnaire :	**27**
Pourcentage du total :	**100,00%**

Quel est votre secteur d'activité ?		
Réponse	nb réponses	Pourcentage
Banques (3)	3	10,00%
Commerce de détail et biens de consommation (5)	2	6,67%
Energie, approvisionnement & industrie minière (6)	2	6,67%
Secteur public (13)	7	23,33%
Services aux entreprises (14)	3	10,00%
Technologie, télécommunications, information-communication et médias (15)	10	33,33%
Transport et logistique (17)	3	10,00%

Quelle est la taille de votre entreprise ?		
Réponse		Pourcentage
Micro-entreprises (jusqu'à 9) (1)	3	12,50%
Petites entreprises (10-49) (2)	3	12,50%
Moyennes entreprises (50-249) (3)	5	20,83%
Grandes entreprises (250 et plus) (4)	13	54,17%

Quel est votre rôle principal dans l'entreprise ?		
Réponse	Décompte	Pourcentage
Gestionnaire de projet (1)	8	29,63%
Programmeur/Developpeur/Team (3)	2	7,41%
Executive Manager (5)	2	7,41%
Manager IT (7)	12	44,44%
Manager hors-IT (8)	3	11,11%

Développez-vous ou maintenez-vous des applications spécifiquement créées		
Réponse		Pourcentage
Oui (Y)	15	60,00%
Non (N)	10	40,00%

Utilisez-vous des méthodes de développement agile ?		
Réponse		Pourcentage
Oui, exclusivement (A1)	0	0,00%
Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile) (A2)	12	80,00%
Non (A3)	3	20,00%

La méthodologie utilisée dépent-elle de la durée de vie attendue de vos		
Réponse		Pourcentage
Oui (Y)	2	16,67%
Non (N)	10	83,33%

En fonction de la durée de vie de vos systèmes, quelle méthodologie de		
Réponse		Pourcentage
Agile (A1)	2	100,00%

En fonction de la durée de vie de vos systèmes, quelle méthodologie de		
Réponse		Pourcentage
Non-agile (A3)	2	100,00%

En fonction de la durée de vie de vos systèmes, quelle méthodologie de		
Réponse		Pourcentage
Non-agile (A3)	2	100,00%

En fonction de la durée de vie de vos systèmes, quelle méthodologie de		
Réponse		Pourcentage
Aucune (A4)	2	100,00%

Quelle méthode agile utilisez-vous ?		
Plusieurs Réponses possibles	nb réponses	Pourcentage
Agile Unified Process (AUP) (3)	2	20,00%
Extreme Programming (XP) (7)	2	20,00%
Feature Driven Development (FDD) (8)	2	20,00%
Kanban (development) (9)	3	40,00%
Rapid Application Development (RAD) (11)	2	20,00%
Scrum (12)	8	100,00%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Hebdomadaire (A2)	7	77,78%
Rarement (A4)	2	22,22%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Hebdomadaire (A2)	5	62,50%
Mensuelle (A3)	3	37,50%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Quotidienne (A1)	5	62,50%
Hebdomadaire (A2)	3	37,50%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Quotidienne (A1)	3	37,50%
Hebdomadaire (A2)	5	62,50%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Hebdomadaire (A2)	3	37,50%
Mensuelle (A3)	5	62,50%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Hebdomadaire (A2)	3	37,50%
Mensuelle (A3)	5	62,50%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Quotidienne (A1)	3	37,50%
Hebdomadaire (A2)	3	37,50%
Jamais (A5)	2	25,00%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Hebdomadaire (A2)	2	25,00%
Rarement (A4)	3	37,50%
Jamais (A5)	3	37,50%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Hebdomadaire (A2)	2	22,22%
Rarement (A4)	5	55,56%
Jamais (A5)	2	22,22%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Quotidienne (A1)	5	55,56%
Hebdomadaire (A2)	2	22,22%
Jamais (A5)	2	22,22%

A quelle fréquence utilisez-vous les pratiques agiles suivantes :		
Réponse		Pourcentage
Hebdomadaire (A2)	2	22,22%
Mensuelle (A3)	2	22,22%
Rarement (A4)	2	22,22%
Jamais (A5)	3	33,33%

Depuis combien de temps utilisez-vous des méthodes agiles de développement ?		
Réponse		Pourcentage
< 1 an (1)	2	22,22%
entre 1 et < 3 ans (2)	7	77,78%

Mesurez-vous le temps consacré à la maintenance de vos solutions informatiques ?		
Réponse		Pourcentage
Oui (Y)	12	60,00%
Non (N)	8	40,00%

Est-ce que vos développeurs s'occupent eux-mêmes de la maintenance des		
Réponse		Pourcentage
Toujours (A1)	8	42,11%
Souvent (A2)	5	26,32%
Parfois (A3)	3	15,79%
Jamais (A4)	3	15,79%

Avez-vous constaté une amélioration ou une dégradation de la durée de		
Réponse		Pourcentage
Oui, plus rapide avec agile que traditionnel (A1)	3	37,50%
Oui, plus lente avec agile que traditionnel (A2)	0	0,00%
Non, aucune différence notoire entre agile et traditionnel (A3)	5	62,50%

3. Le canevas de base de l'interview

Les interviews d'une durée d'environ une heure ont été réalisées sous un format libre. Toutefois, un canevas de base a servi à élaborer au préalable un profil agile, basé sur le questionnaire en ligne, et à formuler des questions spécifiques au sujet de la maintenabilité. Ces questions avaient les buts principaux suivants :

- Spécifier les informations obtenues à partir des réponses du questionnaire en ligne
- Établir le contexte de l'utilisation de l'agilité dans l'entreprise
- Recueillir une vision subjective de l'influence de la maintenabilité sur les systèmes construits à partir des méthodologies agiles.

Vous trouverez ci-dessous la figure 14 montrant la page de garde du canevas de base de l'entretien ainsi que les questions utilisées pour guider l'interview (section 3.1, « Les questions de base de l'interview ».


h e g

AGILITE ET MAINTENABILITE :

« LES METHODES AGILES SONT-ELLES FAVORABLES OU NON POUR LA MAINTENABILITE DES SYSTEMES ? »

TROISIEME PARTIE : ETUDE EN ENTREPRISE

Interview de

Monsieur XX

Titre

Entreprise

Genève, Suisse

Site

Katia Mota Stroppolo | Travail de Bachelor | 28 août 2015

Filière Informatique de gestion

Hes·so // GENÈVE

Figure 14 : page de garde du canevas

3.1 Les questions de base de l'interview

BUTS DE l'ENTRETIEN :

- Analyse subjective du sujet de l'étude : *« Les méthodes agiles sont-elles favorables ou non pour la maintenabilité des systèmes ? »*
- Contexte de développement du-des systèmes abordés lors de l'entretien

1ère PARTIE : SPECIFICATION DU QUESTIONNAIRE

Réponse du sondage	
ID de la réponse	*24*
Date de soumission	*2015-04-01 15 :05 :12*
Quel est votre secteur d'activité ? [Commerce de détail et biens de consommation]	*Oui*
Quelle est la taille de votre entreprise ?	*Grandes entreprises (250 et plus)*
Quel est votre rôle principal dans l'entreprise ? [Manager IT]	*Oui*
Développez-vous ou maintenez-vous des applications spécifiquement créées pour vous ?	*Oui*
Utilisez-vous des méthodes de développement agile ?	*Oui, mais combinées avec des méthodes traditionnelles (Hybride : agile + non-agile)*
Pourquoi ?	
Est-ce pour des raisons…	
Historiques	
Culturelles	
Méthodologiques	
Besoin d'évolution des systèmes	
Complexité des systèmes	
Autres	
Quelles méthodes traditionnelles utilisez-vous ?	
La méthodologie utilisée dépend-elle de la durée de vie attendue de vos applications ?	*Oui*
En fonction de la durée de vie de vos systèmes, quelle méthodologie de développement utilisez-vous ?	
[< 1 an]	*Agile*
[Entre 1 et < 3 ans]	*Non-agile*
[Entre 3 et 5 ans]	*Non-agile*
[Plus de 5 ans]	*Aucune*
Pourquoi ?	
Vous ne développez aucune solution qui pourrait avoir une durée de vie supérieure à 5 ans ?	
Quelles autres méthodes traditionnelles utilisez-vous ?	
Quelle méthode agile utilisez-vous ?	*[Scrum]*
Lorsque vous utilisez Scrum, y introduisez-vous des pratiques « traditionnelles » ?	
Si oui, lesquelles ?	
A quelle fréquence utilisez-vous les pratiques agiles suivantes :	
[Feedback constant du client ou du Product Owner]	*Hebdomadaire*
[Révision régulière du Product Backlog]	*Hebdomadaire*
[Intégration continue]	*Hebdomadaire*
[Standup Meeting]	*Hebdomadaire*
[Revue d'itération]	*Hebdomadaire*
[Iteration Planning]	*Hebdomadaire*
[Burndown Chart]	*Hebdomadaire*
[Livraison Continue]	*Hebdomadaire*
[Test First Programming]	*Hebdomadaire*
[Itérations courtes (2-3 semaines)]	*Hebdomadaire*
[Pair Programming]	*Hebdomadaire*
Depuis combien de temps utilisez-vous des méthodes agiles de développement ?	*entre 1 et < 3 ans*

Mesurez-vous le temps consacré à la maintenance de vos solutions informatiques ?	*Oui*
Comment ? Quelle méthode de sélection de tâches, d'actions et de collecte de mesure du temps utilisez-vous ?	
Est-ce que vos développeurs s'occupent eux-mêmes de la maintenance des systèmes qu'ils développent ?	*Souvent*
Avez-vous constaté une dégradation du temps de la maintenance lorsque ce n'est pas le cas ?	
Avez-vous constaté une amélioration ou une dégradation de la durée de maintenance en fonction de la méthodologie utilisée ?	*Oui, plus rapide avec agile que traditionnel*
A quoi attribuez-vous l'amélioration de la durée de maintenance avec agile ?	
Avez-vous une méthode de comparaison permettant de confirmer ce constat (si oui, laquelle)?	
Comparez-vous des systèmes entre eux ?	
Si oui, comment vous assurez-vous que ces systèmes soient comparables au niveau de la complexité ?	
Dans votre entreprise, lorsque vous développez en agile, mettez-vous en place de mesures pour contrebalancer les possibles fragilités liées à l'agilité, tel que *(et si oui, lesquels ?)* :	
[Documentation plus succincte]	
[Pas de chef de projet]	
[Pas de modélisation de l'architecture de la solution]	
[Pas de spécification détaillée]	
[Code plus difficile à relier aux besoins business]	
[Manque d'uniformité dans le développement (entre différents teams)]	
[Autres]	

2ème PARTIE : CONTEXTE DE DEVELOPPEMENT

Contexte de la solution à analyser dans le cadre de l'étude (subjective ou pratique)
En quelques lignes, quel est le service offert par l'application ?

1.	A quel public s'adresse-t-elle ? (et si possible, pour quel nombre d'utilisateurs ?)
2. 2.1. 2.2. 2.3.	Quel est son niveau de complexité en : Lignes de code Nombre de modules Nombre de méthodes
3. 3.1. 3.2. 3.3. 3.4.	Comment documentez-vous les aspects suivants : BUSINESS spécifications 2- ARCHITECTURE TECHNIQUE MAINTENANCE
4. 4.1. 4.2.	Avez-vous, dans les équipes agiles, des collaborateurs qui remplissent les rôles suivants : Product Owner (interne ou externe à l'équipe) ? Scrum Master ?
5.	Etaient-ils vos collaborateurs avant l'introduction de Scrum ?
6.	Comment et par qui ont-ils été formés ?

3ème PARTIE : SUITE DE L'ETUDE

Seriez-vous intéressé à participer à la suite de l'étude qui consisterait à : •Accueillir une étudiante pour élaborer des scénarios en vue de mesurer l'évolution du temps consacré à la maintenance. • Mesurer la maintenabilité selon les scénarios préétablis.	
[Contactez-moi pour plus de détails :]	*Oui*
commentaire	*Potentiellement. A discuter.*
[Recevoir un exemplaire des résultats de l'étude]	*Oui*
- commentaire	
Vos coordonnées (entreprise, personne de	

contact, mail ou téléphone) :	

1- Dans l'éventualité où vous seriez disposez à participer à la prise de mesure, quel système pourrait être mesuré (un ou plusieurs)?
2- Disposez-vous des historiques de mesures pour ce système ?
3- A-t-il été conçu entièrement à partir d'une méthode agile ?
4- Seriez-vous disposé à le mesurer selon le Framework proposé : GQM et QAS ?
5- Auriez-vous des propositions, suggestions, buts business à atteindre en participant de manière active à l'étude ?
6- Des contraintes particulières ?
7- Quel niveau de visibilité / confidentialité (pour l'entreprise et pour le système à analyser)
8- Quel délai ?
9- Quels contacts à me transmettre (développeurs, chefs de projet, etc)?
10- Quelles démarches à mettre en place pour démarrer l'étude ?
11- Quand puis-je commencer ?

4. Résumé des interviews

Ci-après, vous trouverez les résumés des interviews menées dans le cadre de ce travail de recherche.

Au plus près de ce qui a été dit par les personnes interviewées, les propos exprimés prennent ici la forme de compte-rendu dont le contenu a alimenté le chapitre 4 «Analyses et constats ». Toutefois, les propos et affirmations contenus dans cet annexe reflètent ceux des personnes interviewées et non le mien, auteur de l'étude.

4.1 Interview avec M. Luis Marcos, Japan Tobacco International (JTI)

JTI est une entreprise globale et un des leaders mondiaux de l'industrie du tabac. Son siège social est établi à Genève, en Suisse. La compagnie emploi environ 25'000 collaborateurs, dans différents pays.

4.1.1 Le contexte agile

L'entreprise s'est mise à l'agilité depuis environ 3 ans. La méthode utilisée est Scrum.

Pour implémenter l'agilité, il a fallu changer des processus très standardisés et historiquement ancrés. Un des buts de ces changements étant notamment de diminuer le temps de latence entre l'identification du besoin et le développement. En pratique, cela a contribué à réduire les délais liés aux étapes de gestion de projet qui sont préliminaires au développement (analyse, approbation, etc.)

Chez JTI, les applications mobiles sont 100% agiles, sans aucun apport de méthodologies traditionnelles. Les utilisateurs y sont impliqués en tant que testeurs. Un exemple de cette implication a été le développement du social network d'entreprise, projet interactif « allant bien avec le développement agile », car :

- Exposée rapidement aux utilisateurs
- Feedback régulier
- Actions prises sur la base des avis des utilisateurs

A JTI, il existe deux types de releases. Le premier type de release, les « releases techniques » est hebdomadaire et intègre l'ensemble du code réalisé pour la solution en développement. Les buts de ces releases techniques sont la stabilisation de l'application, le recensement des bugs et la réinjection des corrections à faire dans le pipeline de développement du Sprint suivant (l'input des tests alimente les releases techniques suivantes). Ces releases ne sont pas exposés aux utilisateurs.

Une fois que plusieurs releases techniques forment un ensemble de fonctionnalités conséquent et suffisamment stables, ceux-ci sont regroupées dans une « Release utilisateurs ». Celle-ci va donc comporter un ensemble de plusieurs releases techniques hebdomadaires stables et testées. Le nombre de releases utilisateurs varie selon la complexité de l'application.

Cette livraison par lot de releases techniques assure que, lorsque l'application est exposée à l'utilisateur, celle-ci comporte très peu de bugs (voire aucun). Ce qui atténue la contrainte liée au fait que l'utilisateur doive dégager du temps pour tester.

Il est arrivé que des "goutes d'agilité" soient instillés dans des projets "traditionnels" et ce dans les buts de diminuer le time-to-market, dynamiser le projet, impliquer les utilisateurs et gérer des pressions budget/temps. Ceci était le cas lorsque JTI a utilisé l'agilité pour développer le projet « Leaf ».

« Leaf » était à la base un projet WEB traditionnel et son interface avec l'ERP de l'entreprise rendait les spécifications relativement complexes. Pour une raison de pression sur le time-to-market et pour intégrer les utilisateurs dans le développement, il a été décidé de développer Leaf avec Scrum. Ce projet a duré une année.

4.1.2 Contexte de développement

Chez JTI, les solutions développées avec des méthodologies agiles sont des applications mobiles natives. Ce sont environ 90 applications utilisées par les collaborateurs de l'entreprise et tournant sur des supports mobiles Apple (Iphone et Ipad, propriétés de la compagnie). Cette standardisation des supports permet à JTI de diminuer les coûts de développement (pas de développement multi-support).

Dans un environnement où ça bouge aussi vite que celui des applications mobiles, le choix de Scrum convient parfaitement pour le développement de ces solutions dont le time-to-market est d'à peine quelques semaines.

Environ ¼ des travailleurs connectés au réseau d'entreprise utilisent ces applications, soit :

- ~5'000 utilisateurs Iphone
- ~1'000 utilisateurs Ipad

Les applications mobiles développées avec Scrum sont considérées, de manière générale, comme étant des solutions à usage interne et non-critiques pour le business. Ce sont des applications qui supportent des processus métiers. Il s'agit principalement

de l'accès digital à des bases de données documentaires, revues d'information, matériel formatif et réseau d'entreprise. Quelques exemples d'utilisation :

- Consultation de rapports
- Consultation des marchés
- Consultation de lois
- Consultation de procédures
- Réseau social d'entreprise
- Intranet
- Magasines d'information
- Pocket Office : localisation et infos sur les différents locaux de JTI dans le monde (application mobile multilingue)
- JTI events : conférences internes, pour les collaborateurs ; toute la documentation des conférences n'est disponible que sur format numérique, via l'application mobile.

Beaucoup de ces applications sont le résultat d'une digitalisation ou d'automatisation de documents. Des solutions qui *«ne coûtent pas trop cher mais qui apportent beaucoup de satisfaction aux collaborateurs de JTI».*

L'équipe de développeurs est composée ~13 personnes (dont 3 à Genève + une dizaine de personnes basées en Inde).

Les applications critiques telles SAP et les applications WEB destinées à un public externe et/ou critiques pour le business sont développées à partir de méthodologies traditionnelles, avec des phases d'initialisation (collecte de besoins, architecture de nouvelles fonctionnalités, etc) d'une durée d'environ 6 à 8 mois.

Camion : stable ; complexe ; traditionnel ➔ ERP

Mercedes : stable ; complexité moyenne ; traditionnel ➔ applications WEB

Voiture de course : fast-line ; agile ➔ Applications mobile

Figure 15 : analogie des processus de développement chez JTI et évoquée lors de l'entretien

4.1.3 Les mobiles apps chez JTI

Chez JTI, les solutions mobiles ont un bas coût et un faible risque pour le business car :

- non critiques pour le business
- utilisateurs internes
- durée de développement moyenne de 2-3 semaines
- coût de développement entre 5'000 et 15'000 EUR

Compte-tenu de ces faits, si la solution doit être abandonnée, le risque prit par JTI est considéré comme étant minime.

Les applications sont très peu documentées : à peine quelques pages de Powerpoint pour les spécifications, souvent basées sur des solutions qui existent déjà. Ce manque de documentation est directement lié au manque de complexité des applications développées.

Au niveau de la maintenance, si cela devait coûter trop cher, on jette et on recommence ("solutions Kleenex"). Cette flexibilité et faible prise de risque permet de switcher le développent sur des supports Android s'il faut (cela a même déjà été fait pour des pays comme le Brésil, où les produits Apple coûtent beaucoup trop cher sur le marché local).

Toutefois, il faut souligner que les apps-mobiles en productions sont très stables. A titre d'exemple, sur l'ensemble de ~25 solutions livrées en 2014, seul 4 buggs ont été signalés et corrigés. Ce faible nombre de corrections justifie le fait que chez JTI il n'existe même pas de budget de maintenance pour les apps-mobiles.

Personne interviewée : M. Luis Marcos, Software Architecture & Mobility lead. Genève, le 16/04/2015.

4.2 Interview avec M. F. Trémeau, Altran Suisse

Altran Suisse[11] fait partie du groupe Altran, leader mondial du conseil en innovation et ingénierie avancée. Altran Suisse dispose de bureaux à Zurich, Bâle, Lausanne et Genève. Les domaines d'intervention de l'entreprise sont : le conseil en organisation et systèmes d'information; le conseil en technologies et innovation; le conseil en stratégie et management; et le conseil en solutions de formation sous la marque Altran Education Services.

4.2.1 Le contexte agile

L'offre d'Altran autour de l'agilité est très large : formation, coaching, transition, projets clés en main et placement de ressources.

Pluridisciplinaire et expérimentée, l'équipe de la cellule agile d'Altran a particularité de ne pas être formellement et exclusivement liée à une méthode agile. Ainsi faisant, ils peuvent adapter leur offre au contexte particulier de chaque client.

[11] Source : http ://www.altran.ch/fr/a-propos-daltran/altran-suisse/notre-approche.html#.Vc3qi_ntmkp

Par ailleurs, Altran propose aux entreprises une étude d'opportunité agile lors de laquelle une immersion est faite afin d'établir un diagnostic des pratiques en cours, avec une attention particulières aux contraintes organisationnelles et culturelles des organisations analysées.

La prise en compte de l'aspect culturel permet ensuite d'affiner et de proposer les pratiques et outils les plus adaptés au contexte de l'entreprise et ainsi mieux gérer le processus de transition agile.

« Agilité : une table de mixage avec laquelle on compose pour offrir les services et produits les plus adaptés »

Figure 16 : analogie de l'offre agile d'Altran et évoquée lors de l'entretien

4.2.2 Le wiki d'entreprise comme outil de communication

Les outils collaboratifs de type wiki d'entreprise occupent également une place importante dans le champ de vision de l'agilité chez Altran. En effet, lors de leurs implémentations de l'agilité, les coaches d'Altran forment à l'utilisation des wikis comme outils de communication et peuvent mettre à disposition de leurs clients, via le Cloud, un espace de travail collaboratif Confluence[12] (ou se greffer sur l'outil du client lorsque celui-ci existe déjà).

Personne interviewée : M. Frédéric Trémeau, practice manager – agile coach. Genève, 21/05/2015.

4.3 Interview avec MM. A. Mas et R. Perroud, Etat de Genève

La Direction générale des systèmes d'information (DGSI), rattachée au Département de la sécurité et de l'économie (DSE), est responsable des matériels, logiciels et moyens nécessaires pour garantir le bon fonctionnement des systèmes informatiques et des télécommunications de l'administration cantonale genevoise[13].

4.3.1 Le contexte agile

A la DGSI, l'équipe de développement PHP-Drupal, attachée au service mutualisé en ingénierie logicielle, s'est mise à pratiquer l'agilité depuis environ 6 mois. Pour ce faire, ils ont été coachés par M. Frédéric Tremeau, coach agile externe, employé par Altran Genève et mandaté par la DGSI dans le cadre de ce projet. De plus, ils ont complété leurs connaissances par de l'auto-formation.

[12] Confluence est un logiciel de travail collaboratif développé par Atlassian , https ://fr.atlassian.com/software/confluence .
[13] Source : https ://www.ge.ch/dgsi/missions.asp

4.3.2 L'équipe PHP-Drupal

Première équipe à implémenter l'agilité au sein de la DGSI, l'équipe PHP-Drupal est composée d'environ ~10 personnes, avec une majorité de stagiaires et/ou membres externes (le noyau est composé de 3-4 personnes). Cette équipe est responsable de ~40 petites applications (RAD) et de la mise à disposition des modules fonctionnels Drupal (sur lesquels se basent plusieurs pages WEB internes et externes des services de l'Etat).

Ils maintiennent, corrigent, enrichissent et font évoluer les applications et modules déjà existants ; dans le périmètre de responsabilité décrit précédemment. L'architecture est donc héritée.

Quelques informations à citer :

- Équipe composée de spécialistes (et non de généralistes).
- Spécifications réalisées par les PO
- Apache Subvertion est utilisé pour le lien entre le code et les fonctionnalités ou bugs (via un ID présent dans les messages de commit).
- Tests réalisés avec Selenium.

Les principales pratiques agiles introduites sont :

- Kanban pour le management visuel
- Daily meeting
- Backlog (avec priorisation MoSCoW et indicateur de maturité pour chaque story)
- Revue de code
- Itérations courtes (2 semaines)
- Releases fréquents (1 mois)

Les deux personnes interviewées soulèvent le principal frein à la validation des parties complexes du code par des tests unitaires : le coût. En effet, et surtout en agile, plus la notion de qualité et de complexité est élevée, plus les tests devraient être automatisés. La relation « plus de qualité / plus de tests automatisés / plus de coûts » est donc à prendre en compte dans les choix et compromis effectués.

Il est également soulevé le fait qu'il n'est pas toujours évident de faire participer activement et régulièrement le « client ». Ce d'autant plus que les clients du service sont des acteurs internes et donc des « clients-non-payants ». Il est donc toujours d'actualité, pour certains de leurs clients, de souhaiter n'intervenir qu'une seule fois au départ de la demande et d'espérer recevoir la « bonne » solution une fois que celle-ci est finie.

Toutefois, ils ont pu observer que le fait de livrer régulièrement contribue à diminuer des éventuelles tensions entre le « client » et le service dans le sens où le client augmente sa confiance dans la capacité de l'équipe à livrer le produit. Ce gain de confiance a un effet positif sur la motivation du client à participer au développement de la solution.

4.3.3 Effets perçus sur l'amélioration de la maintenabilité

Pour les interviewés, le management visuel apporte à l'équipe une amélioration de la collaboration interne et une augmentation des actions partagées. Ces bienfaits sont perçus comme étant favorables à la maintenance dans le sens où ils mettent fin au travail isolé et individuel que l'équipe pratiquait auparavant.

De plus, les développements itératifs et les releases plus fréquents sont également vus comme étant favorables à la qualité de la maintenance des solutions gérées par l'équipe. Cette amélioration est notamment exprimée par une amélioration de la qualité du code et une plus grande adéquation, du point de vue fonctionnel, entre le résultat final et les attentes du client.

Personnes interviewées : M. Alexandre Mas, responsable du service « Services Mutualisés en Ingénierie Logicielle » et M. Remy Perroud, team leader de l'équipe de développement PHP-Drupal, DGSI, Etat de Genève. Genève, le 22/05/2015

4.4 Interview avec M. J. Daudel, Qim info

Qim info est une entreprise fournissant des services dans le domaine des technologies de l'information : placement de collaborateurs en régie, projets clés en main, prestations à la demande et conseils. Etablie à Genève depuis 2005, celle-ci possède également un bureau à Lausanne[14].

4.4.1 Le contexte et résumé de l'entretien

Qim info développe des solutions agiles depuis environ une année. La méthode la plus utilisée est Scrum.

Une fois la proposition commerciale acceptée[15], le mandat est découpé en lots qui formeront les itérations. Des librairies intégrées sont utilisées pour l'architecture de haut niveau. Cette architecture de haut niveau peut ensuite être détaillée sous forme d'Use cases ou des maquettes (surtout pour les applications mobiles).

[14] Source : http ://www.qiminfo.ch/index.php/fr/

[15] Lors de la structuration et proposition d'une offre commerciale, les documents suivant sont produits : product-Backlog, document de vision et étude d'opportunité.

En principe, les livraisons partielles des itérations sont faites à un informaticien et non à une personne du coté métier. La raison étant le besoin de faire tester et valider la solution. Ce qui implique le besoin d'avoir une personne qui puisse comprendre les tests cases et valider l'application. Cette démarche de validation est souvent effectuée avec l'assistance du chef de projet pour les projets traditionnels et du PO pour les projets agiles, ces deux rôles étant assumés par des collaborateurs de Qim info.

Les équipes sont composées de généralistes[16] et non de spécialistes. Toutefois, Qim info compte parmi son équipe 1 ou 2 spécialistes de tests. Les collaborateurs travaillent autant sur des projets agiles que traditionnels.

Quelques outils utilisés :

- SharePoint
- Management visuel (post-it)
- Tableaux de bord pour le suivi interne
- Excel pour le backlog et les cas de tests.

4.4.2 L'agilité dans des projets traditionnels

A Qim info, l'agilité est instillée également sur des projets traditionnels car le fonctionnement interne de l'équipe est basé sur des pratiques Scrum. Ils estiment ne pas percevoir de différence entre le code final produit par une méthodologie agile et celui produit par une méthodologie traditionnelle.

Lors d'une approche traditionnelle (cf. *"où le client ne s'implique pas et ne s'intéresse qu'au résultat final")* le rôle de PO est interne à Qim info et le client est sollicité moins souvent. Dans ce cas, pour le fonctionnement interne agile de l'équipe, le rôle du client est assumé par le chef de projet qui devient le point de contact traditionnel pour le client.

En développement traditionnel, les livraisons intermédiaires correspondent aux maquettes (~5 maquettes + 1 livraison finale). Ce livraisons intermédiaires ne sont pas mises en production (sauf pour les maintenances).

4.4.3 La maintenabilité des solutions

Qim info assure également la maintenance d'applications qu'elle développe ainsi que la maintenance de solutions développées par d'autres sociétés. Cette maintenance,

[16] Souvent 3 par team. Ce qui est considéré comme étant « juste assez ». Idéalement 5 par team.

qu'elle soit de caractère évolutif ou correctif, est gérée de manière agile sous la forme de release composée d'un ou plusieurs sprints.

Les applications maintenues sont généralement isolées. La complexité est traitée en renforçant le niveau de détail des tests cases et en assurant la présence de quelqu'un qui connait déjà l'application. En absence de ce type d'expertise, les tests cases et interactions avec le métier sont renforcés.

Lorsque l'on évoque la maintenabilité d'applications dont la durée de vie est supérieure à 3 ans, il semblerait qu'une des difficultés soit de trouver l'équilibre entre les coûts et les gains apportés par la maintenance. Il est alors préféré de privilégier les petites maintenances, voire de contourner le problème plutôt que d'engager des grands frais.

Quelques autres caractéristiques de la maintenance :

- Les documents utilisés sont les manuels utilisateurs (pas de documentation technique, le lien entre le métier et le code est fait au travers du manuel uniquement).
- Les interactions liées aux tâches de maintenance sont réalisées au plus près du métier (plutôt que des équipes techniques).
- Les tests sont privilégiés par rapport à l'architecture.

4.4.4 Agilité et lien avec le métier

Qim info a pu observer qu'en développant avec agile on pourrait faire face au risque de se retrouver avec un patchwork d'applications incohérentes entre elles et qui s'éloignent du cœur de métier.

Même si Qim info ne travaille pas avec des équipes multiples et distribuées sur un même projet, l'effet patchwork d'applications semble être une conséquence de ne se focaliser que sur le développement. Cette focalisation pourrait résulter en une perte du lien global avec le métier.

Un exemple cité est la correction et l'ajustement des données et le rôle du responsable BD qui doit modifier la structure de la BD. La reprise de la BD et le refactoring de son structure n'apporte pas de plus-value visible pour l'utilisateur qui n'en voit pas l'intérêt. D'où la difficulté de placer ce travail dans un projet agile.

Le développement traditionnel favoriserait plus ce type de travail "invisible pour le client" en apportant une vue d'ensemble du système qui est plus marquée. Toutefois, le développement traditionnel demande beaucoup plus de temps pour la réalisation des tâches qui apportent peu ou pas de plus-value au client.

4.4.5 Un framework en dessus de l'agilité

Il resterait donc à trouver une solution qui apporterait un équilibre entre plus-value et vue d'ensemble. Une solution possible serait de placer un framework fédérateur au-dessus du développement agile. Le but étant de trouver un compromis permettant de bénéficier de l'agilité, tout en gardant un lien fort avec l'ensemble du métier et en évitant de se retrouver avec un ensemble de choses inutiles du point de vue de l'utilisateur.

Ce framework devrait apporter plus que du pilotage : il devrait apporter le moyen de comprendre le métier, d'en avoir une vision partagée et de accroître la conscience qu'un développeur qui comprend le métier sera toujours plus efficace. Il s'agirait également de trouver des niveaux d'abstraction pouvant représenter tant les vues des détails que les vues globales. Ce qui représente une difficulté tant au niveau technique que métier.

Personne interviewée : M. Jacques Daudel, directeur technique, Qim info. Genève, le 17/06/2015.

4.5 Interview avec M. D. Lo Giudice, Forrester Research

Forrester Research est une entreprise indépendante qui fournit à ses clients des études de marché sur l'impact des technologies dans le monde des affaires. Forrester Research a 19 implantations géographiques à travers le monde, dont huit centres de recherche[17].

L'analyste interviewé, M. Diego Lo Giudice[18], est un des principaux experts de Forrester sur les processus et pratiques liés aux cycles de vies des développements software. Il couvre ainsi des sujets tels que le développement agile, agile et lean transformations, stratégies et services de développement d'approvisionnement agiles, les pratiques et les outils de test agile, DevOps et tests de logiciels et de la qualité. M. Lo Giudice couvre également les métriques de livraison, l'intelligence artificielle et la gouvernance de l'open source.

4.5.1 Les bienfaits de l'agilité sur la maintenabilité

Différentes enquêtes menées par M. Lo Giudice lui ont permis d'observer que lorsque l'agilité est appliquée correctement celle-ci favorise la maintenabilité des systèmes. Selon lui, cette favorisation est due à l'amélioration de la qualité fonctionnelle et technique des softwares développés en agile.

[17] Source : https ://fr.wikipedia.org/wiki/Forrester_Research
[18] Source : https ://www.forrester.com/Diego-Lo-Giudice

4.5.1.1 Agilité et amélioration de la qualité fonctionnelle

En ce qui concerne la qualité fonctionnelle, il semblerait que, appliquée correctement, l'agilité aiderait les organisations à produire les bons logiciels qui répondent aux besoins fonctionnels des entreprises. Ce gain en justesse vis-à-vis des besoins fonctionnels des clients permettrait de diminuer les tâches de la maintenance qui consistaient à corriger, faire évoluer et réparer des problèmes fonctionnels.

Selon M. Lo Giudice, ces frais de maintenance étaient justifiés par le fait que, dans un développement traditionnel, il existerait plus de fonctionnalités ne répondant pas aux besoins et attentes du client. Corriger ce manque de qualité fonctionnelle prendrait une large part du budget de maintenance.

Or, dans un développement agile, le business s'implique beaucoup plus, collabore étroitement avec les équipes IT et les aide à prioriser les fonctionnalités qui sont à implémenter. La qualité fonctionnelle est donc augmentée et ceci a un effet directement positif sur la maintenance.

4.5.1.2 Agilité et amélioration de la qualité technique

M. Lo Giudice a également observé que, appliquée correctement, l'agilité contribuerait à produire des softwares d'une plus grande qualité du point de vue technique. C'est-à-dire, "des systèmes qui ne présentent pas de problèmes techniques qui impactent le business".

Cette qualité augmentée grâce à l'agilité peut être typiquement attribuée aux pratiques suivantes :

- Building journalier dans un serveur d'intégration continue.
- Exécution journalière de tests automatisés.
- Une rigoureuse discipline d'engineering servant à mettre en production un logiciel plus stable.

Les deux aspects précédents contribueraient à produire des systèmes qui sont plus maintenables tant d'un point de vue fonctionnel que technique.

4.5.2 « L'agilité appliquée correctement »

Lors de cet entretien, M. Lo Giudice a mis l'accent sur le principe des bénéfices de l'agilité lorsque celle-ci est appliquée correctement. Pour lui, être « agile » a un sens bien plus large que juste suivre une et une seule méthodologie.

Selon lui, beaucoup d'entreprises font l'erreur de croire que parce qu'ils adoptent Scrum ils deviennent agile. Or, Scrum seul n'est pas agile : Scrum reste une

méthodologie de management de projet qui, malgré un grand nombre de pratiques nécessaires et de grande qualité, ne peut pas être ni effectif ni suffisant lorsque celle-ci est appliquée toute seule. Pour résumer, le concept de « être agile » est bien plus grand que Scrum.

Par ailleurs, les tendances montrent que beaucoup d'entreprises que disent utiliser Scrum utilisent, en fait, un mix de plusieurs méthodologies. Dans cette tendance de mixité, DevOps[19] surgit comme un moyen de « devenir agile » tout au long du chemin et ce en renforçant la communication et la collaboration entre l'équipe de développement et les autres professionnels IT.

Une autre tendance montre la possibilité d'appliquer l'agilité au travers de frameworks qui intègrent l'ensemble du contexte de l'entreprise - et non seulement le contexte de développement. Un exemple donné, *« même s'il ne va pas forcement apporter des réponses à tous les problèmes »*, serait SAFe®[20]. Ce framework est de plus en plus adopté et permet aux entreprises de développer de manière agile tout en prenant en compte d'autres besoins tels les planning à long terme, les PMO, les BPM, les Enterprise Architect team, les Architect de changements, etc. Or, dans un contexte plus complexe, toutes ces activités sont absolument nécessaires pour délivrer des projets agiles.

Surtout lorsqu'il s'agit de travailler à grande échelle car il y aura d'autant plus de besoins en architectures de haut niveau. Sans compter le besoin de prendre en considération de facteurs organisationnels. Facteurs qui vont au-delà de l'organisation du team dans le projet.

Dans ce cas, *"si l'on utilise juste Scrum, rien ne va être agile"*. De plus, le business va être insatisfait car, malgré la vitesse de développement, ils ne pourront même pas déployer les solutions : les processus en place au niveau opérationnel ne permettant pas un fonctionnement agile effectif (notamment au niveau des intégrations continues).

4.5.3 La disparition de la maintenance dans un contexte agile

Selon M. Lo Giudice, la maintenance, telle que l'on la connaissait dans un environnement de développement traditionnel, tendrait à disparaitre lorsque l'on devient « vraiment agile ». En effet, en agile et Lean, les personnes qui ont développé le système restent celles qui le connaissent le mieux. D'où le besoin, clairement

[19] https ://en.wikipedia.org/wiki/DevOps
[20] http ://www.scaledagileframework.com/

exprimé par ces deux courants, de garder ensemble le team aussi longtemps que possible afin que ceux qui ont construit soient également ceux qui maintiennent.

Ainsi faisant, la maintenance faite en deux temps, par deux différentes équipes disparaitrait. Permettant notamment de régler un grand problème qui est la perte de connaissance lors des transitions. En effet, ce passage pose un grand problème : même si l'on documente très bien les choses, le travail sera repris par des personnes qui ne l'on pas réalisé, ce qui provoque une perte de savoir.

Toujours selon lui, lorsqu'on est sur le chemin de « devenir plus agile », la tendance est de garder l'équipe ensemble et de ne pas dire : « ok, vous êtes l'équipe de développement, quand vous aurez fini votre boulot, celle-là sera l'équipe de maintenance ». C'est manière de voir la maintenance est très inefficace et ce quel que soit la manière de documenter le système. De plus, devoir documenter pour quelqu'un d'autre tout le travail effectué ainsi que la manière dont on a travaillé introduit des coûts.

Évidemment, garder l'équipe ensemble tout le long du cycle de vie du système est un idéal difficile à atteindre. M. Lo Giudice observe des entreprises qui se lancent dans ce défi en y employant un bon management qui prête attention à la satisfaction de leurs collaborateurs. Ce management reconnait l'effort qu'il faut faire pour garder leurs équipes en place et investissent des ressources pour conserver leurs bons éléments. Ainsi faisant, ils arrivent en effet à mieux garder leurs équipes dans l'entreprise.

Toutefois, il se peut que cet idéal difficile à atteindre justifie le fait que certaines entreprises n'appliquent pas l'agilité à tous les niveaux mais juste là où la plus-value business peut être plus vite réussie et plus facilement visible.

4.5.4 Le cycle de vie de l'application

Ces changements de pratique impliquent un changement dans le cycle de vie de l'application : ce n'est plus une application qui est finie puis mise en opération et maintenue - c'est une application en constante évolution et en livraison continue.

Or, dans ce contexte de livraison continue, il se pose la question de savoir quand retirer une application et quel serait l'impact de cette fin de vie. A ce sujet, l'agilité emmène des nouvelles métriques bien plus proches du business et qui permettent de déterminer quand une application commence à ne plus être utilisée. C'est alors que l'on peut commencer à planifier de la retirer.

Ces nouvelles manières d'organiser et de voir le développement peuvent être plus facilement applicables par des petites entreprises. Toutefois, l'on voit des plus en plus de grandes entreprises qui choisissent d'agir comme une start-up et d'appliquer les concepts du Lean-startup[21] management. Ce qui représente un grand et difficile défi.

4.5.5 Gestion des flux

M. Lo Giudice a pu observer des entreprises qui ont pu utiliser l'agilité pour obtenir des améliorations de la maintenabilité des leurs systèmes développé en traditionnel. Celles-ci n'utilisaient pas Scrum mais Kanban[22].

Kanban emmène le principe Lean de la gestion des flux.

Selon M. Lo Giudice, lorsque l'on maintient un large système qui est en production depuis des années, l'on reçoit un montant très élevé de requêtes de changement. La seule façon de vraiment gérer ce flux de requêtes est de le considérer comme un flux continu de demandes et ensuite optimiser le parcours de ce flux.

Cette optimisation figure parmi les gains apportés par des techniques Kanban telles : le management visuel (le mur Kanban), le WIPL et l'amélioration continue, issues de la pensée et pratiques Lean.

4.5.6 La documentation agile

M. Lo Giudice pense également qu'il n'est pas vrai que les équipes agiles ne produisent pas de la documentation. Lorsque l'agilité est appliquée correctement, il existe, selon lui, une grande source de documentation dans les outils utilisés. Il s'agit là d'une documentation qui n'est pas nécessairement réalisée manuellement mais néanmoins très riche en information. La différence d'approche avec une méthodologie traditionnelle réside dans le fait qu'en agile on n'écrit la documentation que lorsque celle-ci répond à un réel besoin.

Un exemple de documentation agile est le Backlog. Celui-ci peut être documenté à différents niveaux, selon le besoin et le contexte de l'entreprise. Notamment pour répondre à des standards et besoins de conformité liés au business. D'autres exemples comme les tests, les versions, les UC et les prototypes viennent enrichir la liste de documents souvent présents dans les projets agiles.

[21] https ://fr.wikipedia.org/wiki/Lean_Startup
[22] https ://fr.wikipedia.org/wiki/Kanban_(d%C3%A9veloppement)

4.5.7 Quand choisir l'agilité

M. Lo Giudice pense qu'il s'agit d'une question de contexte. Un exemple donné est celui des applications mobiles car, selon lui, celles-ci pourraient difficilement se passer de l'agilité. Par ailleurs l'évolution du marché montre bien que, dans le développement mobile, l'agilité est la norme.

Toutefois, les méthodologies de développement purement agiles ne conviennent pas forcement à tout type de projet. Il existe en effet une difficulté à appliquer l'agilité à une large échelle. Des méthodologies comme Scrum ne peuvent pas être simplement appliquées à large échelle sans employer des grands efforts. Efforts nécessaires et sans lesquels l'expérience deviendrait un échec.

Le fait étant que, dans un environnement à grand échelle, Scrum seul n'est pas suffisant. *"Par ailleurs, tout le monde dit utiliser Scrum mais en fine il ne s'agit pas de Scrum tout seul. Le concept Agile est bien plus grand que Scrum".*

Pour finir, il pense qu'il ne serait pas efficient d'utiliser l'agilité dans une application âgée et comportant ~10'000'000 LOC. Au mieux, utiliser Kanban servirait à augmenter son efficacité. C'est par ailleurs ce qui font plusieurs entreprises : elles n'appliquent pas l'agilité partout et surtout par sur des zones qui possèdent un trop grand nombre de "legacy".

Concernant les méthodologies employées par le marché, il semblerait que lors de ces deux dernières années on ait pu observer une baisse d'adoption de méthodologies telles RUP et CMMI. Ces 2 dernières étant remplacées par SAFe® (et très peu par DAD).

Personne interviewée : M. Diego Lo Giudice, vice President, principal analyst serving application development & delivery professionals, Forrester Research. Genève, le 22/05/2015. Monsieur Lo Giudice ne parlant pas français, il n'a pas pu valider le contenu ici présenté.

5. Etapes de construction du tableau 2 : mapping entre les facteurs favorisant la maintenabilité et les principes, pratiques et rôles agiles

Le tableau 2 de page 28, cf. figure 17, propose une correspondance entre les principaux facteurs favorisant la maintenabilité des systèmes et des principes, pratiques et rôles agiles proposées par les méthodologies XP, Scrum, Lean SD, DAD et SAFe®.

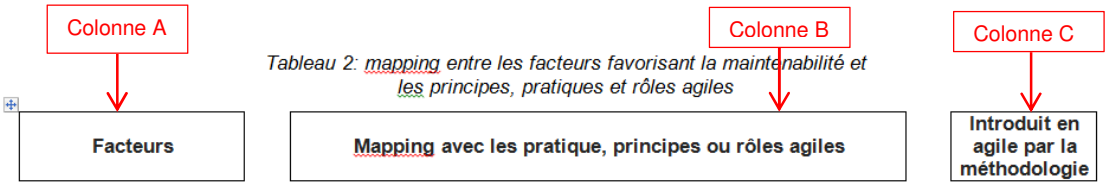


Figure 17 : extrait de l'en-tête du tableau 2, page 28

Ce tableau a été constitué selon les étapes schématisées à la figure 18. Les correspondances entre les colonnes A et B ont été réalisées à partir de l'analyse des documents et publications reportés dans la colonne « Sources analysées » du tableau 3 de cette même annexe.

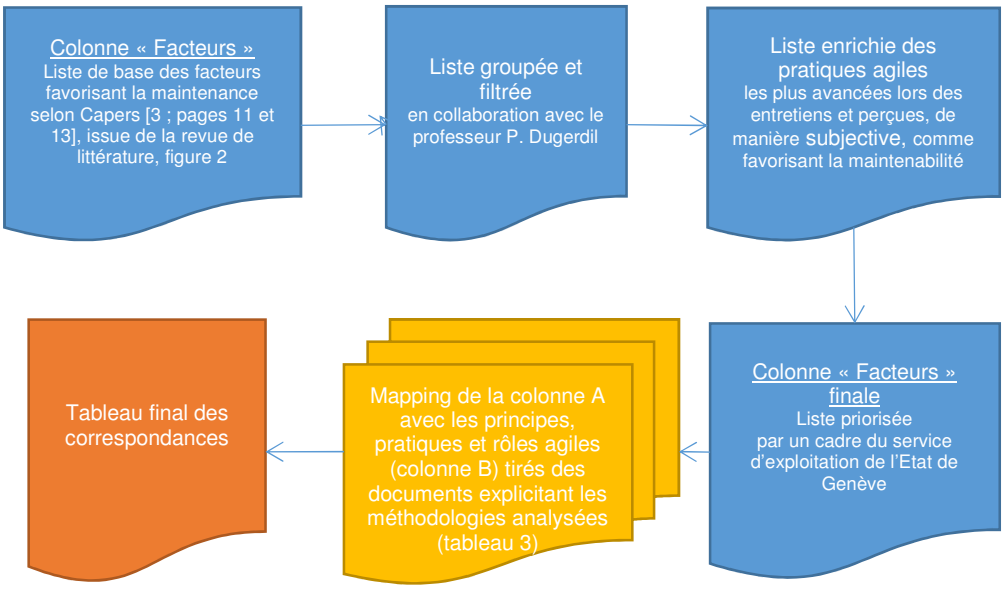


Figure 18 : étapes de construction du tableau 2, page 28

Les méthodologies ont été analysés dans l'ordre chronologique de la première publication la définissant (cf. colonne « année d'origine » du tableau 3). Cette analyse chronologique nous a permis de déterminer, dans la colonne C « Introduit en agile par la méthodologie » - figure 17, quelle méthodologie agile a introduit le principe, rôle ou pratique. L'origine des méthodes a été établie sous la base de l'année de la première publication la définissant et reporté dans la colonne « année d'origine » du tableau 3.

Tableau 3 : origine des méthodes et documents utilisés

Méthode	Année d'origine	Sources analysées
XP	2000 [15]	http ://www.extremeprogramming.org/rules.html
SCRUM	2001 [16]	http ://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-FR.pdf & Scrum et XP depuis les Tranchées [20]
Lean Software Développement	2003 [17]	Lean Software Development : An Agile Toolkit [17]
SAFe®	2010 [18]	http ://www.scaledagileframework.com/
DAD	2012 [19]	Disciplined Agile Delivery : A Practitioner's Guide to Agile Software Delivery in the Enterprise [19]

Une fois le tableau de correspondance établi, celui-ci a été soumis à M. Gaël Heulin, Gestionnaire du service de soutien d'ingénierie d'exploitation, Coordinateur SIRH – Surveillance Applicative Généralisée, Direction générale des systèmes d'information (DGSI) de l'Etat de Genève.

Le tableau 4 présente la liste priorisée selon la méthode MoSCoW dont la définition Wikipédia est la suivante :

*« La méthode **MoSCoW** est une technique visant à prioriser des besoins ou des exigences en matière d'Les lettres majuscules de l'*acronyme *MoSCoW signifient (en anglais) :*

- M - MUST have this, c'est-à-dire 'DOIT être fait' (Vital).
- S - SHOULD have this if at all possible, c'est-à-dire 'DEVRAIT être fait dans la mesure du possible' (Essentiel).
- C - COULD have this if it does not affect anything else, 'POURRAIT être fait dans la mesure où cela n'a pas d'impact sur les autres tâches' (Confort).
- W - WON'T have this time but WOULD like in the future, 'NE SERA PAS fait cette fois mais sera fait plus tard' (Luxe, c'est votre zone d'optimisation budgétaire).

Les o dans MoSCoW sont ajoutés uniquement pour rendre l'acronyme prononçable et sont, dans la majorité des cas, écrites en minuscule pour indiquer qu'elles ne correspondent pas à des mots. Toutefois, l'écriture MOSCOW, avec les o en majuscule, est tolérée. »

Tableau 4 : liste priorisée des facteurs favorisant la maintenabilité

Facteurs favorisant la maintenabilité	MoSCoW
Respect des standards de l'entreprise	**Must**
Partage et acquisition continue de connaissance	**Must**
Vision globale des buts de l'entreprise	**Should**
Contrôle et suivi des changements	**Should**
Expérience élevé de l'équipe de maintenance dans le produit à maintenir.	**Should**
Reporting, suivi de l'occurrence et élimination des défauts.	**Should**
Pratiques d'amélioration continue	**Should**
Stratégie globale de management de l'amélioration et évolution continue de la qualité du SI	**Should**
Management visuel du flux de travail en cours	**Should**
Architecture et design favorisant la maintenabilité	**Could**
Compréhension du besoin métier	**Could**
Documentation	**Could**
Expérience élevé de l'équipe de maintenance dans le langage	**Could**
prévention d'erreur (mistake-proofing)	**Could**
Critères d'acceptation métier clairs	**Could**
Comprendre le métier dans sa globalité et en avoir une vision partagée.	**Could**
Intégration continue	**Could**
Transition développement / maintenance	**Could**
Espace de travail adéquat	**Won't**
Faible complexité du code de base	**Won't**
Mesure de satisfaction des clients comme indicateur pour évolution / refactoring.	**Won't**
Tests automatisés.	**Won't**

6. Le Lean IT - une autre manière de penser la maintenabilité des systèmes

6.1 Quelques extraits pour un brin d'histoire

Historiquement, le Lean est apparu à la fin de la 2e guerre mondiale et appuyé sur un besoin des japonais de relancer leur économie suite à la défaite face aux américains[23].

Succès après succès, le Lean s'est étendu et plusieurs déclinaisons en sont nées. Toutefois, ses déclinaisons gardent relativement inchangés les principes fondateurs exprimés notamment par le monument Lean [figure 19] et l'esprit principal tiré de la représentation du toit : « produire la meilleure qualité, au meilleur prix et dans les délais demandés par le client ».

6.1.1 Le monument Lean

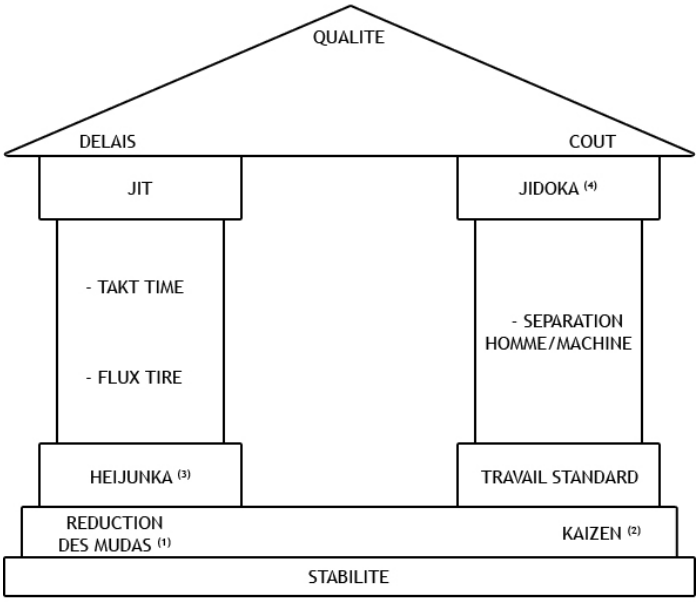


Figure 19 : Maison Lean-Manufacturing
(1) Gaspillage (2) Amélioration continue (3) Séquençage
(4) Autonomation : stop et notifications des anomalies
Source : http ://www.vision-lean.fr/lean-manufacturing/monument-lean-manufacturing/

« Le monument Lean est le symbole utilisé par ses fondateurs pour expliquer la cohérence et l'harmonie du système Lean (Lean Manufacturing).

La stabilité est la fondation du monument Lean. Appliqué à l'organisation, on parle de stabilité des équipes, de standardisation des méthodes, de stratégie suivie dans le temps…

[23] pour en savoir plus sur les origines du Lean, lire l'article wikipédia à l'adresse : https ://fr.wikipedia.org/wiki/Toyotisme

Le socle du monument Lean, sur lequel tout le reste est bâti, est constitué de 2 éléments : la dynamique Kaizen - ou progrès continu - et l'élimination des mudas : tous deux mettent le système en mouvement.

Les deux piliers du monument Lean (JIT et JIDOKA) reposent sur :

- Heijunka : lissage - séquençage de la production
- Travail standard : une variabilité réduite du rythme et des processus de travail : Un système destiné à absorber le plus possible les à-coups de la demande.

Les outils utilisés dans les murs du monument Lean pour soutenir son toit (l'objectif de la méthode) sont :

- Pour le pilier JIT : flux tiré, Takt time et flux continu.
- Pour le pilier Jidoka : séparation homme - machine (un opérateur gère plusieurs machines) et automation : machines autonomes détectant leurs propres erreurs.

Le toit, ou objectif de la méthode Lean Manufacturing, est résumé par CQD, baisse de coûts de production, amélioration du niveau de qualité, adaptation des délais des processus aux besoins du client. »

Extrait de la page : http ://www.vision-lean.fr/lean-manufacturing/monument-lean-manufacturing/

6.2 Le Lean IT

Le Lean IT est une variante d'application du Lean adaptée aux systèmes de l'information.

Dans leur livre sur la pratique du Lean dans l'IT [29] les auteurs présentent le Lean comme n'étant ni un idéal normatif que l'entreprise devrait suivre jusqu'au bout ni comme une complète improvisation. Ils utilisent plutôt la métaphore avec la recherche d'un idéal impossible à atteindre : celui de la recherche de l'étoile du Nord, figure d'un idéal, qui est certes inatteignable mais qui conduit chacun à apprendre comment s'en approcher sans pour autant imposer une voie unique et toute tracée.

Ce chapitre met en avant quelques un de ces principes dans un contexte d'évolution de la maintenabilité des systèmes.

6.3 L'identification et la résolution des problèmes

L'identification et la résolution des problèmes comme un levier d'amélioration est également un concept fort du Lean : plutôt que les cacher, on va les ramener à la lumière, on va les exposer, les investiguer pour en trouver les causes profondes tout

en évitant de se borner aux causes superficielles. Seulement après cet effort, on va établir des hypothèses, appliquer des actions dont les résultats sont mesurables, les mesurer et tout recommencer jusqu'à ce que le résultat attendu soit obtenu.

Lorsque l'on y arrive, on va s'efforcer de pérenniser les acquis apportés par ces cycles en renforçant ou en créant des nouveaux standards et en partageant la connaissance.

Il s'agit là d'une application de la roue de Deming : le PDCA appliqué de façon continue sur chaque problème que l'on a décidé explicitement d'améliorer. A la différence près que le cycle ne se fera pas à l'intérieur d'une seule tête bienpensante ni depuis une chaise de bureau : le cycle commencera par une période d'immersion sur le terrain, là où le problème se trouve, c'est l'incontournable Go & See du Lean (figure 20).

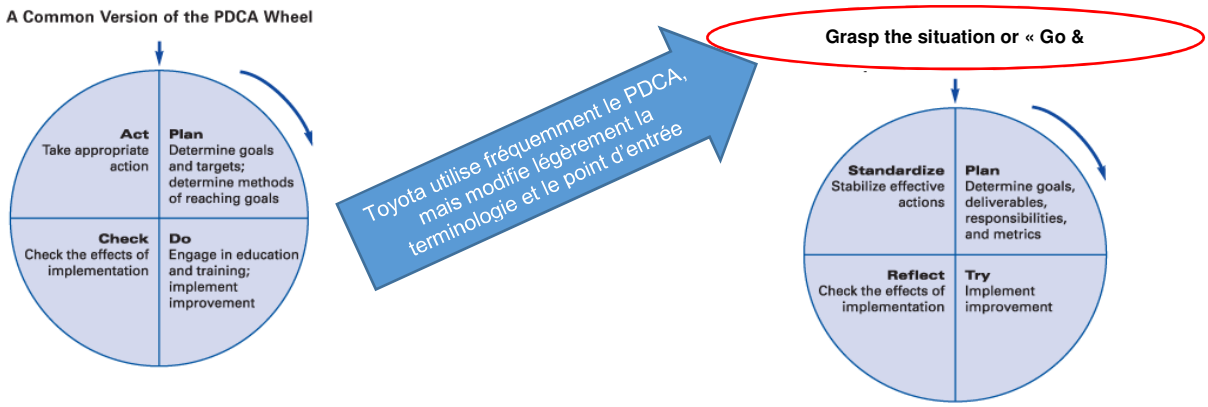


Figure 20 : PDCA en Lean, enrichissement du Go & See

Source : http ://www.lean.org/Common/LexiconTerm.cfm?TermId=286

La démarche Lean est inflexible sur ce dévoilage constant et recherche continue de résolution des problèmes. A ce titre, une place de choix est réservée pour les mettre en évidence dans l'Obeya (ou war-room, figure 21) - juste à côté du tableau de travail hebdomadaire - meublant ainsi le mur de la pièce où l'on s'attarde le plus.

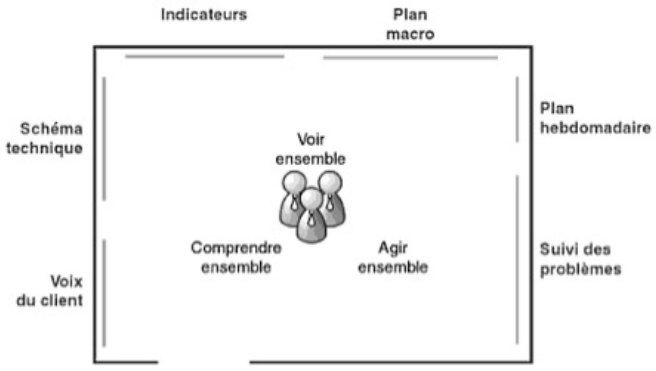


Figure 21 : Une Obeya

Cette vision du problème comme un outil d'amélioration nous mène à appréhender la maintenance comme un levier d'optimisation de l'ensemble du système d'information de l'entreprise.

En effet, de par sa proximité avec les clients internes et externes de l'entreprise, la maintenance apporte nombreux points d'entrée concrets pour mesurer la satisfaction du client : les défauts, les demandes et besoins d'évolution, l'adaptation aux changements imposés par le métier et son environnement, tout y passe !

Identifier les causes profondes d'un problème et imaginer des contre-mesures est une démarche commune dans le traitement des problèmes selon le Lean. En effet, trouver en équipe les causes, imaginer les meilleures solutions (les plus efficaces et moins coûteuses) est une démarche et un souci commun et constant. Démarche qui commence par l'identification et l'exposition des problèmes.

Le Lean propose plusieurs outils pour orienter cette démarche schématisée et illustrée à la figure 22.

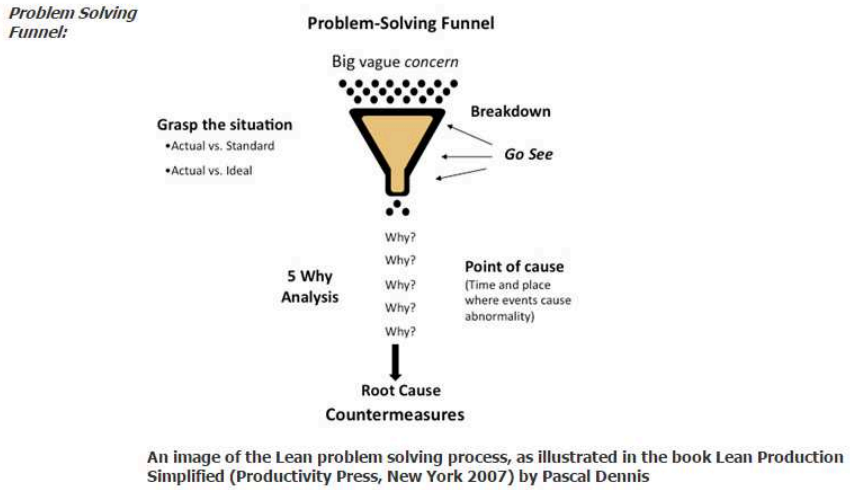


Figure 22 : démarche Lean de résolution de problèmes
Source : http ://www.leansystems.org/glossary.html

6.4 L'idéal du Zéro-défaut

Toute cette démarche de mise en lumière, de résolution de problèmes et d'amélioration continue devraient avoir par conséquence une augmentation de la perception de la valeur que le client ressent dans les produits qu'il utilise tout comme dans le service que lui est fourni par l'équipe opérationnelle.

Toutefois, pour ne citer que Bill Price «L'absence de service est le meilleur service» [30]. C'est pourquoi, la recherche de l'idéal Lean du Zéro-défaut devrait être une

préoccupation à tous les niveaux : du développement au management, de haut en bas et de bas en haut.

En effet, chaque étape du processus ajoutant de la valeur devrait également encapsuler la recherche de cet idéal : celui qui est responsable de l'ajout se sent aussi responsable de poursuivre l'idéal de passer son « œuvre » au suivant sans que celle-ci ne comporte aucun défaut, sans perdre de vue les finalités idéales qui sont un client émerveillé (par opposition à un client satisfait) et un ensemble optimisé (par opposition à une partie optimisée).

6.5 Voir plus loin, mieux agir de prêt

Or, cet idéal ne peut être atteint sans prendre de la hauteur par rapport à ce que l'on construit. Ce n'est qu'on gardant le tout que l'on arrive à produire des parties qui contribuent à un fonctionnement optimale du système.

Le Lean IT favorise ce changement de vision myope qui peut apporter certaines pratiques agiles en cherchant continuellement à améliorer le développement et le fonctionnement des équipes : le Lean IT propose (et non impose) des nombreux outils permettant de voir le tout dont un exemple est le VSM pour « Value Stream Map », un outil qui permet de regrouper toutes les actions, à valeur ajoutée et à non-valeur ajoutée, par lesquelles passent un produit ou un service.

Avoir cette vue d'ensemble permet de fournir un service où l'intégrité interne est assurée. Or, comme l'écrivent Poppendieck dans leur livre sur le Lean Software Developpement, "l'intégrité interne est un prérequis indispensable à l'intégrité perçue" et ce d'autant plus lorsqu'il s'agit de résoudre les demandes des clients dans un contexte de maintenance.

6.6 Réagir rapidement pour protéger le client

Lorsque un problème impactant le client est trouvé, chercher la plus petite solution immédiate et peu couteuse pour y remédier et l'appliquer. Tout en étant conscient que remédier à un problème et protéger le client ne solutionne pas ce problème. Pour ce faire, il faut s'attaque à en trouver la cause qui a provoqué l'anomalie. Autrement dit, il faut trouver le gap dans le processus qui a permis à l'anomalie d'atteindre le client, en comprendre les causes profondes et corriger.

6.7 Le bac rouge de résolution de problèmes

Comme dit précédemment, protéger le client n'est pas la même démarche que solutionner le problème. En Lean, solutionner signifie trouver « la » cause profonde qui provoque le problème et s'y attaquer.

Un des outils qui pourraient être facilement utilisé pour appuyer cette démarche est le « bac rouge ». En industrie, le bac rouge représente cet endroit, à vue de tous, où l'on entrepose les pièces défectueuses afin de les retirer du circuit de production. Ces pièces sont destinées soit à être réparées, soit à être remplacées. Dans tous les cas le bac rouge sert à mettre en évidence le principe qu'un problème est survenu et qu'il faut chercher la cause qui l'a provoqué puis la réparer pour éviter que d'autres erreurs se reproduisent.

En informatique, cela pourrait être représenté par un tableau, une pochette, ou tout autre outil qui servirait à cumuler les problèmes rencontrés. Ce bac serait exposé et à côté des tâches en cours. Pendant le Daily meeting, l'équipe passerait quelques minutes à réfléchir aux problèmes cumulés dans le bac rouge, à les priorisér[24] et à mener des actions pour en absorber le contenu[25].

6.8 QA à gauche

Lorsque l'on commence à s'intéresser au Lean, on comprend vite que la maison Lean n'existe pas sans son toit et que tout en haut de celui-ci on trouve la qualité : construire la qualité de l'intérieur est objectif présent tout au long de la démarche, quel que soit la variante.

L'agilité met en avant plusieurs outils de contrôle de la qualité sous forme de tests : fonctionnels, techniques, systèmes, etc. Des tests réalisés avant de coder, des tests exécutés régulièrement, etc.

Or, s'il semble admis que ce focus régulier sur la validation du code par les tests apporte une amélioration de la qualité interne du système en maximisant la correction des bugs et surtout en évitant la phase de validation et d'acceptation unique et réalisée tout à la fin des développements dit « classiques », il n'en est pas moins un fait que l'équipe de l'assurance qualité (QA), testeurs et autres, reste néanmoins celle qui est chargée de trouver les défauts une fois que ceux-ci sont créés. C'est un peu comme

[24] La cible Lean serait d'arriver à identifier les 20% de processus qui provoquent les 80% des incidents, selon la règle de Pareto.

[25] Le chapitre 4 du livre [29] aborde cette application en lien avec des exemples et une démarche d'amélioration continue.

les pompiers du développement : censés identifier et éteindre les incendies provoqués par les défauts.

Déplacer des ressources de la QA vers la gauche du processus, en pair avec l'analyse quand celle-ci a lieu, dans tous les cas avant le code (cf. figure 24), pourrait apporter des nombreux avantages :

- La collaboration analyste-QA pourrait renforcer la boucle de feed-back du client autour des tests. Or comme Poppendieck [29] mentionne les tests fonctionnels peuvent être des excellents outils de communication client-développeur. D'où l'importance de les faire au plus près des Stories.
  - Le développeur peut bénéficier des moyens de répondre à la question « comment je sais que je fais juste » en suivant les tests unitaires crées par la paire analyste-QA.
  - Les NFRs peuvent être prisent en compte dès le début du flux d'ajout de valeur et non-seulement validées ou refusées à la fin de celui-ci
  - Le collaborateur QA trouvera un rôle d'amélioration et d'évolution de la qualité plutôt qu'un seul rôle de pompier et le système entier convergera plus facilement vers un système possédant l'intégrité interne globale recherchée. En effet, bien souvent l'équipe QA est une équipe transversale ayant une vision globale qui va au-delà du produit développé. Position transversale qui les permet de maîtriser les contraintes liés aux qualités non-fonctionnelles requises par le système.

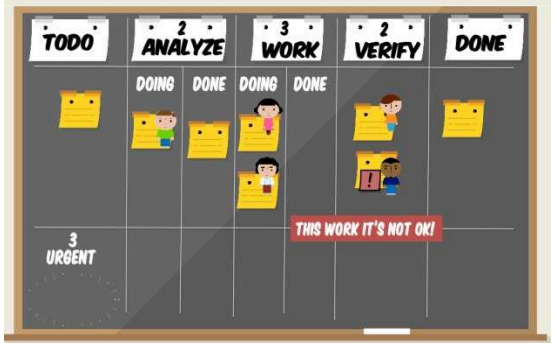


Figure 23 : Kanban - flux classique
Source : http ://fr.slideshare.net/GiulioRoggero/how-a-kanban-board-works

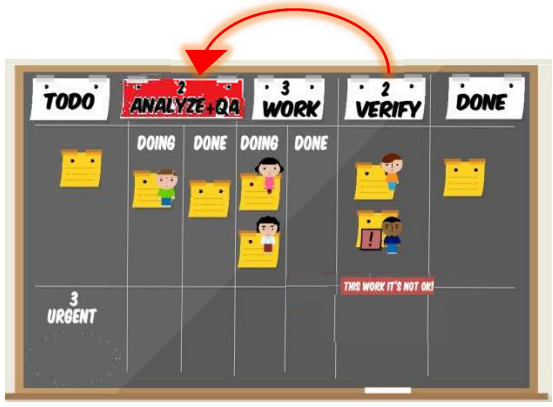


Figure 24 : pair Analyse + QA