REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Systèmes d'Informations Avancés

Par: FEKROUN FATIMA ZAHRA

SUJET

La proposition d'une composition agile basée sur les paramètres Green Computing

Soutenu publiquement le: 31 /06 /2016 devant le jury composé de :

MHENNI TAHER	**Université de M'sila Président**
MANSOURI KAMAL	**Université de M'sila Rapporteur**
BOUGHRARA SAI D	**Université de M'sila Examinateur**

Promotion : 2015 /2016

Dédicace

Je dédie ce modeste travail :

- *A mon père qui m'a donné le gout de savoir Dr : S. FEKROUN.*
- *A ma mère qui m'a aidé.*
- *A mes frères Alla Eddine et Issam Eddine.*
- *A mon mari S.L. Nassim*
- *A ma belle famille*
- *A ma copine Yasmine.*
- *A tous ceux qui m'ont aidé, de près ou de loin. Même qu'il soit un mot d'encouragement et de gentillesse.*

...Et a toute ma famille

FEKROUN Fatima Zahraa

Remerciements

Ce mémoire est le résultat de l'engagement de plusieurs personnes qui ont décidé de nous accompagner résolument dans cet exaltant parcours. Je voudrions profiter de cet espace pour leur exprimer toutes mes gratitudes et mes reconnaissances.

Je premières pensées vont à notre encadreur Dr. MANSOURI KAMEL, d'avoir accepté de nous encadrer et de nous suivirent durant toute cette période.

Je voulant remercier mes parents d'avoir prié pour moi jours et nuits.

Et enfin je voudrais remercier globalement mais chaleureusement tous ceux qui ont aidé de près ou de loin.

Sommaire

Liste des tableaux

Liste des figures

Liste des abbreviations

CMMi*: Capability Maturity Model Integrated*

DoD*: Department Of Defense*

RAD: *Rapid Application Development*

FDD: *Feature Driven Development*

RUP:*Rational Unified Process*

XP: *eXtreme Programming*

OOPSLA:*Object-Orientes Programming, Systems, Langages, and Applications*

CMOS:*Complementary Métal-Oxide Semi-conducteurs*

Introduction général

Plusieurs organisations ou groupes qui développent des logiciels, ne connaissent que les approches traditionnelles qui sont souvent lourdes et coûteuses à gérer.

On critique principalement le dépassement de coûts et les délais de livraison, ce qui amène les petite séquipes à délaisser ces approches. À défaut, peu de méthodes alternatives sont présentées. Il y a un écart entre l'univers des praticiens et les modèles théoriques.
L'évolution des technologies et le manque des ressources sont des facteurs qui contribuent àla diminution de la grosseur des équipes. Il est important de trouver des approches adéquates à leur contexte organisationnel et écologique.

Les approches agiles offrent possiblement une opportunité afin de répondre au besoin d'une méthode suffisamment structurée. Elle doit permettre à mener des projets dans une zone sécuritaire.

D'autre part, Green Computing est une discipline qui améliore les l'impact des systèmes IT sur l'écosystème, tel que l'efficacité énergétique et la gestion des déchets informatique.

L'objectif est de proposer d'une méthode agile efficace pour développer un produit informatique, et d'autre part, le produit développer soit durable, ou / et non gourment au ressource, ou /et auto dégradable.

Le mémoire est composé de quatre chapitres, qui, dans l'ordre, présentent

- Les bases de l'agilité.
- la méthodologie SCRUM.
- les paramètres Green Computing.
- et en fin l'implémentation.

Le premier chapitre présente les éléments de base qui définissent l'agilité. Il établit quelle est l'offre de ces approches dans le domaine du génie logiciel. Il positionne ces approches par rapport aux modèles établis. Finalement, il dresse le bilan des points communs et des différences avec les approches traditionnelles.

Le deuxième chapitre présente l'historique de SCRUM, et un bref un tour d'horizon. Il permettra d'apporter les informations nécessaires pour une meilleure compréhension de processus de développement SCRUM et les rôles des intervenants, et finalement les avantages et les inconvénients de la méthode.

Le troisième chapitre présente la définition de Green Computing, les déférentes stratégies Green pour le hardware et le software, et enfin les domaines d'action permettant à une entreprise d'intégrer une démarche de green IT.

Le dernier chapitre présentera un moyen d'implanter cette méthodologie basé sur les paramètres Green au sein d'une équipe de développement d'un projet.

Finalement, une conclusion, avec une bibliographie, et une application Web comme exemple.

CHAPITRE 1

LES BASES DE L'AGILITE

Ce chapitre présente les éléments de base qui définissent l'agilité. Il établit quelle est l'offre de ces approches dans le domaine du génie logiciel. Il positionne ces approches par rapport aux modèles établis. Finalement, il dresse le bilan des points communs et des différences avec les approches traditionnelles.

1. Les bases de l'agilité

1.1. Définition du terme « Agile »

Présentement, l'attrait pour le terme « Agile » est répandu. Chaque secteur d'activité peut attacher ce terme à un paradigme. Il doit être interprété selon le domaine auquel il est appliqué. Ce terme n'est pas breveté, il n'y a pas de limitation sur son utilisation, ce qui peut donner lieu à différentes définitions apparentées.

La définition littéraire du mot « Agile » ne suffit pas. Il faut considérer ce terme comme étant la définition la plus proche, pouvant définir la capacité de répondre aux changements que proposent ces approches. L'agilité exige de s'adapter au changement inhérent aux environnements turbulents. Ces changements pouvant être au niveau des spécifications, des intervenants, des procédures et surtout, dans le domaine logiciel, des technologies.

Appliqué au génie logiciel, Jim Highsmith définit l'agilité ainsi : « L'agilité est l'habilité conjointe de créer et de répondre aux changements d'un environnement turbulent au profit de l'entreprise. » Traduit et tiré de [1].

Où elle est définit comme suite :

Mode de développement logiciel léger et adaptable qui permet de délivrer rapidement le maximum de valeur et de qualité. Elles préconisent un ensemble de « bonnes pratiques » rassemblées en un tout cohérent et qui se renforcent mutuellement : développement itératif et incrémental, équipe pluridisciplinaire incluant le donneur d'ordre, collaboration ouverte et franche, formalisme au service du projet, etc.

1.2. Origines des approches agiles

Avec l'évolution des technologies, les méthodes traditionnelles de développement logiciel s'avèrent trop lourdes dans plusieurs situations. Une opportunité pour trouver des alternatives structurées était souhaitable. Au cours des années 1990, différents praticiens ont mis sur pied

des méthodes qui changeaient la conception traditionnelle de mener un projet de développement logiciel.

Ces premiers « Agilistes » se sont rencontrés au Centre de ski Snowbird en Utha, le 13 novembre 2001. Cette rencontre avait pour objectif d'échanger et de trouver les points communs des méthodes promues par les dix-sept participants présents. C'est lors de cette rencontre qu'ils ont rédigé le manifeste agile [2], pierre d'assise du mouvement.

Par la suite, ils ont regroupé une liste de principes pour enrichir le manifeste. L'ensemble se veut un cri de ralliement, afin de faire connaître leur point de vue à l'industrie du logiciel [3].

L'origine du terme « Agile » est issue de l'ingénierie et des milieux manufacturiers qui ont introduit la notion de gestion agile. Lorsqu'il est venu le temps de définir le mouvement de pensée rassemblant les différentes méthodes, ce terme fût adopté. Il transpose le transfert d'expertise provenant des autres domaines d'ingénierie. Du coup, les termes « léger » ou « maigre » pouvant être perçus comme étant simplistes et négligés, furent écartés.

Les différentes approches agiles sont principalement définies par des méthodes qui mettent en œuvre différentes pratiques. Unitairement, les pratiques ne sont pas nouvelles, il s'agit pour la plupart d'un retour vers la simplicité. L'agilité amène une réflexion sur l'objectif du travail et l'esprit dans lequel il doit être réalisé.

1.3. Le manifeste agile

Les méthodes agiles se définissent par leur adhésion aux critères du manifeste qui constitue la pierre d'assise des approches agiles. Les projets de développement sont réalisés avec des ressources limitées qui imposent des choix. Le manifeste souligne les préférences des agilistes. Il communique les valeurs des fondateurs et leurs préoccupations, qui les menèrent à ces conclusions: Une méthode propriétaire d'entreprise peut être agile, si celle-ci respecte le manifeste et les principes agiles. Le manifeste est composé de quatre énoncés.

Il débute et se termine par deux autres phrases tout aussi intéressantes.

Nous sommes à découvrir de meilleures manières pour développer
des logiciels en les pratiquant et en aidant les autres à le faire.
À travers notre travail nous en sommes venus à valoriser :
les individus et les interactions, *davantage que les processus et les outils,*
les logiciels fonctionnels, *davantage que la documentation compréhensive,*
la collaboration avec le client, *davantage que la négociation de contrat,*
la réponse au changement, *davantage que le suivi d'un plan.*
Bien que l'élément de droite soit important, l'élément de gauche l'est davantage

Tableau 1.1 Le manifeste agile [2]

La phrase d'introduction comporte beaucoup d'éléments. Elle met en évidence que les méthodes agiles sont encore à l'étape de la découverte. Que leurs supporteurs sont des praticiens et qu'ils ont un esprit de collaboration, afin d'aider les autres dans leur apprentissage. Ils veulent ainsi introduire cette approche comme une alternative prometteuse plutôt qu'une solution ultime qui doit être inculquée. Le manifeste se poursuit avec quatre énoncés qui mettent l'accent sur différents aspects des projets de développement logiciel.

- **Les individus et les interactions, davantage que les processus et les outils.** La cohésion d'une équipe est plus importante que le suivi des conventions. Cette phrase s'attarde à la qualité des individus impliqués et l'esprit de collaboration qui en fait un véritable travail d'équipe. Cet élément reflète la synergie et la motivation qui doit laisser au second plan la rigidité du processus ou la complexité des outils. On laisse de côté l'élitisme, souvent lié aux connaissances techniques, au profit des qualités interpersonnelles et à la motivation.
- **Les logiciels fonctionnels, davantage que la documentation compréhensive.** Un logiciel partiellement fonctionnel, est toujours plus valable qu'une documentation soignée. L'objectif d'un développement logiciel est de produire un logiciel, la documentation doit être considérée comme un support complémentaire aidant à sa compréhension. Il est préférable d'investir ses efforts sur le produit, plutôt que sur les éléments qui le décrivent.
- **La collaboration avec le client, davantage que la négociation de contrat.** Le client et le fournisseur sont des partenaires qui partagent un risque. L'agilité vise la collaboration des deux parties pour atteindre un objectif commun. Le client doit s'impliquer et diriger l'équipe dans les étapes de réalisation. Il doit s'approprier l'évolution de son produit. L'agilité évite le gaspillage des négociations et des demandes de changement. Elle propose d'investir ces efforts de manière constructive.
- **La réponse au changement, davantage que le suivi d'un plan.** L'agilité reconnaît l'apprentissage fait au cours d'un projet. Que tout n'est pas connu d'avance et que des adaptations sont inévitables. Le changement est intégré au modèle de développement. La planification originale ne doit pas être trop détaillée, car certains aspects ne seront probablement jamais réalisés. Ceci évite de détailler inutilement des éléments trop tôt dans le projet. Avec la dernière phrase, le manifeste conclut en établissant des priorités. Les Agilités reconnaissent l'importance des éléments de droite, mais jugent que les éléments de gauche doivent être favorisés. Ce n'est pas un désaveu ou un

abandon de la pratique, les Agilistes prônent la transparence et avouent leur préférence. Dépendamment des méthodes, ces préférences sont aussi nuancées.

1.4. Les principes agiles

Le manifeste ne suffit pas à communiquer toute la richesse de la réflexion des Agilistes. Une série de principes précisant leur vision, suit la rédaction du manifeste [4]. *Nous suivons ces principes:*

- **Notre plus haute priorité est de satisfaire le client en lui livrant rapidement et de façon continue, un logiciel de qualité.** Mettre le logiciel rapidement à l'épreuve pour connaître les ajustements décelés par le client. Il peut ainsi apporter son retour d'expérience.
- **Accepter les changements de besoins, même lors du développement.** Toutes les méthodes agiles incorporent la gestion des changements. Les modifications sont intégrées lors de la planification d'une itération.
- **Les processus agiles exploitent les changements pour augmenter les avantages compétitifs du client.** En cours de projet, les opportunités avantageuses pour le client peuvent être incluses à son avantage.
- **Livrer fréquemment un logiciel fonctionnel, en visant les délais les plus courts, de quelques semaines à quelques mois.** De cette manière, nous obtenons une preuve d'avancement du produit et minimisons les risques. Ces livraisons rapides motivent l'équipe, ce qui aide à maintenir un bon rythme de travail.
- **Gestionnaires et développeurs doivent travailler ensemble, de façon quotidienne, pour toute la durée du projet.** Peu de contrôle formel existe, la reddition de compte n'est pas une activité prioritaire. Des rencontres quotidiennes augmentent la qualité de la communication et permettent le suivi au gestionnaire.
- **Bâtir des projets autour d'individus motivés.** Le cœur des projets agiles passe par les individus impliqués. Les gens doivent avoir l'énergie et le courage nécessaire que demandent de tels projets.
- **Donner leur l'environnement et le support nécessaire et ayez confiance qu'ils feront le travail.** Des équipes autonomes et autorisées à prendre des décisions techniques peuvent optimiser les résultats. Les gestionnaires doivent accepter de faire confiance et déléguer.
- **La méthode la plus efficace pour transmettre l'information à l'équipe de développement et à l'intérieur de celle-ci, est par conversation de personne à personne.** L'interaction directe entre les individus est le médium de transmission le

plus simple et efficace. La communication non-verbale peut être perçue et des précisions apportées.

- **Un logiciel fonctionnel est la mesure principale de l'avancement.** Mettre à l'essai le produit est la seule preuve valable de progression. Le client doit constater l'avancement des travaux. Les schémas ne sont que des hypothèses tant que le système n'est pas réalisé et testé.
- **Les processus agiles favorisent le développement maintenable.** L'ensemble du processus suit différentes règles qui simplifient l'entretien. Ces pratiques augmentent la rapidité d'intégration des changements.
- **Les responsables, les développeurs et les usagers devraient pouvoir conserver un rythme constant indéfiniment.** Pour assurer une progression constante, la charge de travail doit être soutenable. Il faut prendre en considération les besoins humains qui exigent des temps d'arrêt. Les procédures de fin d'itérations comblent ce besoin.
- **Une attention continuelle à l'excellence technique et un bon design, augmentent l'Agilité.** Une bonne conception facilite les modifications. La charge mentale est moins imposante. Il est plus facile de comprendre le code afin de le modifier. Cette attention a un impact direct sur les coûts d'entretien.
- **La simplicité ou « l'art de minimiser la quantité de travail fait inutilement » est essentielle.** La simplicité et la conception minimale évitent des travaux inutiles. Les travaux en lien avec des hypothèses ou des plans à long terme, doivent être évités. Le code sera adapté lorsque nécessaire dans un contexte concret.
- **Les meilleures architectures, exigences et designs surgissent d'équipes auto organisées.** L'équipe doit pouvoir prendre les décisions techniques, car elle est la mieux placée pour prendre ce type de décision. Leurs compétences technologiques doivent leur permettre l'autonomie requise pour optimiser les résultats dans ce domaine de compétences. Les développeurs sont souvent fiers de leur travail et recherchent l'excellence technique.
- **À intervalles réguliers, l'équipe réfléchit sur une façon de devenir plus efficace, puis elle adapte et ajuste son comportement en conséquence.** Combiné avec les temps d'arrêt entre les itérations et sa capacité d'autogestion, l'équipe est appelée à corriger ses pratiques dans l'objectif de trouver un équilibre convenable (sweetspot). Il n'y a pas de méthode ou de culture uniforme, celle-ci doit trouver son équilibre au cours de différentes itérations, afin que tous soient satisfaits des conventions établies.

- **Il existe plusieurs méthodes connues, cadrant dans ces valeurs et principes.** D'autres peuvent aussi avoir un succès moins rayonnant et parfaitement correspondre à ces critères qui en font une méthode agile.

Ce qui définit l'agilité est l'adhésion à ces principes. Appliquer cette philosophie est plus importante que le suivi de standard. Ce principe rejoint l'énoncé du manifeste qui indique qu'ils aident les gens dans cette découverte d'efficacité. Ensemble, le manifeste et les principes définissent les valeurs agiles selon l'alliance. Ils constituent les bases pour les méthodes désirant porter l'appellation. Dépendamment des pratiques implantées, le niveau d'attente de ces valeurs peut varier. La prochaine section propose une analyse approfondie des bases de l'agilité.

2. L'offre agile

L'esprit entourant les approches agiles est souvent comparé aux sports d'équipes. L'une des approches s'appuie littéralement sur le soccer. D'autres proposent une analogie avec l'escalade en tant que jeux coopératifs, car elle comporte plusieurs similitudes.

La coopération des autres est nécessaire pour atteindre un but. On ne connaît pas la voie précise à suivre, car elle est découverte au cours de la progression. Nous progressons rapidement avec des pistes sûres et lentement dans les pistes incertaines [5].

Afin de définir l'offre des approches agiles, nous allons premièrement préciser certaines perceptions. Ensuite, on présente une analyse sur l'apport qu'elles offrent au domaine de l'industrie du logiciel selon différentes visions. Finalement, nous situerons l'agilité dans l'offre globale des différentes approches en génie logiciel.

2.1. Perceptions initiales

Lors d'un premier contact avec le manifeste et les principes agiles, la perception des gens diffère en fonction de leurs expériences. Généralement, les gens adhèrent aux valeurs agiles mais, anticipent certaines lacunes ou demeurent perplexes. Avant de pousser l'analyse, je désire préciser certaines perceptions erronées.

On pourrait croire que les méthodes agiles sont des « anti-méthodes » négligées, sans structure de processus. Qu'elles constituent l'antithèse des méthodes rigoureuses. Que leurs adeptes sont des Hackers ou des Cow-boys. Que l'aspect itératif consiste à coder et corriger.

En fait, l'alliance agile n'a pas de droit sur l'appellation « Agile » et ne peut limiter son utilisation. N'importe qui peut utiliser ce terme, ce qui porte à confusion. Les méthodes associées à l'alliance agile comportent des modèles de processus et des pratiques à suivre.

Leurs pratiques sont exigeantes et demandent beaucoup de discipline, ce qui est loin d'être à négliger [6].

On pourrait croire que l'objectif principal des approches agiles est la rapidité de livraison. Que le code est programmé plus rapidement par des techniques spéciales ou des outils de productivité. En fait, les gains de temps se font par des livraisons réduites respectant des échéances fixes. Ces méthodes cherchent à minimiser la lourdeur des échanges d'information et les tâches inutiles. Les équipes sont plus autonomes, ce qui diminue les délais d'approbation.

Tous ces éléments contribuent indirectement à livrer plus rapidement. De plus, certains pourraient croire que le produit est de meilleure qualité et contient moins de défauts. Que le client est plus satisfait et que le projet est moins risqué. En réalité, la qualité du produit n'est pas nécessairement meilleure avec ces approches. Une bonne équipe de développeurs évoluant dans un environnement traditionnel, réalisera un produit d'aussi bonne qualité. Les mêmes bonnes pratiques peuvent être utilisées dans les deux types d'approches. L'intégration des tests dans le développement contribue à améliorer la qualité du produit. L'intégration des changements répondant mieux aux, satisfait davantage le client.

2.2. L'apport des approches agiles

Cette section vise à mettre en relief l'apport des approches agiles selon différentes visions. Il est intéressant de noter que ces approches touchent aussi le domaine de la psychologie et de la sociologie. Les projets agiles se réalisent dans un état d'esprit qui tient compte du facteur humain. À travers différents ouvrages de synthèse, plusieurs auteurs ont fait ressortir des éléments communs [1; 7]. En suivant l'ordre des énoncés du manifeste, nous faisons ressortir l'apport pour les développeurs, le produit, le client et la gestion de projet.

2.2.1. L'apport agile pour les développeurs

La première phrase du manifeste touche directement les individus et les interactions. Au-delà des outils, il y a les gens. Les développeurs impliqués dans un projet agile doivent être prêt à vivre une expérience de groupe. La cohésion de l'équipe est au centre du potentiel de réussite du projet.

En dehors des connaissances techniques et des règles méthodologiques, l'aspect humain est considéré [8]. Différents éléments de ces méthodes s'attardent aux besoins de nature psychologique et sociologique. Différentes pratiques recherchent la sécurité, la reconnaissance des pairs et l'accomplissement personnel par la créativité. [9].

Cette préoccupation s'assure d'obtenir la meilleure contribution de chacun, tout en conservant la motivation. La planification limitée sur de courtes échéances conserve le focus et permet d'être capable de concrétiser l'apport du travail journalier.

Les méthodes agiles s'appuient sur les individus. Elles ont besoin de gens compétents et motivés. Elles s'attardent aux attitudes plutôt qu'aux aptitudes [10]. On préfèrera un développeur moyen et motivé plutôt qu'un expert individualiste. En diminuant les efforts de communication, on favorise l'accès aux informations [11]. On contribue au partage des connaissances techniques des experts et à la compréhension générale du système.

Les connaissances de l'individu s'améliorent continuellement, ce qui est une source de motivation. L'intégration de la conception, de la réalisation et des tests, élimine des transferts de connaissance lourds et coûteux. Elle responsabilise davantage les personnes en charge, ne pouvant ainsi invoquer des carences dans les spécifications. On évite ainsi la déresponsabilisation liée au transfert (throw over the wall). Les développeurs intègrent différents champs de compétences. En diminuant le cloisonnement des tâches, on facilite la relève lors d'absence ou de départ.

Les développeurs procèdent à l'apprentissage du domaine d'affaire du client. Au fils des rencontres, le client amène son retour d'expérience avec le produit. Les deux parties se connaissent mieux et diminuent leurs préjugés. Le client est plus sensible aux contraintes techniques. Cette collaboration permet d'optimiser les solutions proposées. Il en résulte un réel partenariat.

L'utilisation de délimiteurs fixes, améliorent le moral de l'équipe. En utilisant des périodes de temps fixe, les gens ne pourront être découragés par une limite continuellement repoussée.

Cette caractéristique a l'avantage de banaliser l'horaire, pour en simplifier la gestion et favoriser la présence des participants. De plus, elle force la prise de décision pour prioriser les tâches.

2.2.2. L'apport agile pour le produit

La seconde phrase du manifeste insiste sur un logiciel fonctionnel. Les fonctionnalités couvertes par le développement du produit évoluent et s'adaptent aux changements. C'est principalement ce qui démarque les produits issus des approches agiles.

Souvent implanté dans les technologies orientées objets, l'utilisation de patrons de conception (design patterns) est encouragée, car ils permettent de simplifier les communications. Les autres techniques d'amélioration de la productivité, telles que la réutilisation ou l'approche par composants sont appliquées selon les méthodes. Les métriques

de codes établies sont aussi applicables. Leur application est rarement discutée, car cette facette du développement est souvent laissée à la discrétion de l'équipe de développement.

Les qualités internes du produit sont importantes. En adoptant un design qui encourage l'adaptabilité, l'ajout de fonctionnalités est plus facilement intégrable. L'assurance qualité et les tests effectués à tous les cycles, minimisent les défauts. Le produit est continuellement remis opérationnel par l'intégration continue. Les tests automatisés peuvent être livrés avec le produit pour valider son fonctionnement.

En priorisant les fonctionnalités, chaque livraison du produit répond précisément au besoin le plus important. Chaque itération ajoute quelque chose d'utile au produit. Dépendamment des méthodes, plusieurs variables influencent ce qui doit être priorisé. Le produit est toujours le résultat des décisions prises par le client.

On maximise les efforts investis directement dans la production du logiciel. Il y a peu d'effort perdu sur des éléments inutiles. Les documents utiles ou exigés par le client sont produits. Les efforts logistiques sont limités à leur plus simple expression. Le produit final est un système qui comporte les fonctions essentielles et qui reflète ce qui peut être fait de mieux en fonction de l'investissement [12].

2.2.3. L'apport agile pour le client

La troisième phrase du manifeste souligne la collaboration avec le client. Dans le cas des projets agiles, la collaboration avec le client est essentielle. Le client est impliqué dans le projet et détermine les priorités. La planification du projet évolue et s'adapte aux changements qu'il propose. Une telle implication amène un changement de vocabulaire. Le client parle alors de « son » système plutôt que du « votre ». Le client est le principal décideur dans le projet. Le produit est le résultat de ses décisions. La planification récurrente des itérations, évite de gaspiller des efforts dans des fonctionnalités inutiles. Il est responsable de s'assurer que l'investissement comporte réellement une valeur pour lui.

En recevant de petites livraisons rapidement, le client essaie le produit et le valide. Il peut bénéficier rapidement des fonctions essentielles du système et l'exploiter. Il est moins lourd et engageant de procéder de cette manière. Il peut ainsi obtenir un retour sur son investissement plus rapidement. S'il juge que le projet devient trop risqué, il peut décider de le suspendre ou d'y mettre un terme.

Si le produit ne convient pas, il pourra être corrigé. Cette adaptation est un avantage concurrentiel pour le client. On évite l'effet de tunnel et permet de développer les opportunités découvertes au cours du projet. Cette démarche favorise l'émergence des connaissances tacites qui sont difficilement exprimables au départ.

2.2.4. L'apport agile pour la gestion de projet

La dernière phrase du manifeste met de l'avant l'adaptabilité. Toutes les méthodes agiles sont inspirées du modèle itératif en spirale de Boehm [13] et par la roue de Deming [14]. On ajoute à ce modèle l'aspect incrémental. Toutes les parties du système peuvent être améliorées au cours des itérations. Autrement, le modèle itératif peut devenir une série de cascades.

Les planifications récurrentes sur de courtes échéances, permettent un meilleur suivi et diminuent les risques. Les éléments doivent être complétés à l'intérieur d'un cycle, ce qui évite l'étirement des tâches. L'intégration des étapes de réalisation (conception, codage et tests) contribue à clore les fonctionnalités prévues. Les changements sont introduits lors de ces renouvellements. Au terme du projet, la somme des efforts de planification peut équivaloir à un projet traditionnel, mais réparti différemment à travers le temps.

La révision périodique de l'avancement du projet, peut s'harmoniser avec les suivis budgétaires de l'entreprise et répartir les dépenses suivant les trimestres [15].

Cette révision des ententes partage mieux les risques entre le client et son fournisseur. Le client évite les demandes de changement qui peuvent dépasser le budget. Le fournisseur peut estimer plus précisément, sans avoir besoin de surévaluer pour compenser d'éventuels dépassements [16]. Cette réduction d'engagement peut faciliter le démarrage d'un projet.

Plusieurs des méthodes agiles ne sont pas complètement structurées et doivent être précisées. L'équipe doit trouver la méthode suffisante pour réaliser le projet, tout en restant dans une zone de confort, qui lui offre une sécurité raisonnable. Avec ces approches, l'équipe a le pouvoir de questionner la pertinence de certaines pratiques. Il s'agit parfois d'analyser un processus pour réaliser que des éléments peuvent être allégés ou éliminés.

3. Position de l'agilité

Il est intéressant de positionner l'agilité par rapport à des approches établies. L'alliance agile définit ses exigences via ses valeurs et ses principes. Les autres approches peuvent s'y comparer et déterminer leur conformité. Il est possible de satisfaire différentes approches simultanément. Tout en étant agile, il est possible de se conformer à certaines normes, moyennant certains compromis, donc il est possible d'établir un équilibre satisfaisant.

3.1. Conformité au modèle certifié

La conformité aux normes préoccupe les entreprises. La certification peut être exigée des certains donneurs d'ordre. Leurs objectifs de prévisibilité ou de répétitivité diffèrent des approches agiles, mais ne sont pas nécessairement incompatibles. L'adaptation d'une méthode

agile, pour une certification, demande souvent de compromettre un peu son agilité [16, 17,18]. C'est à l'organisation de déterminer ses priorités. Que l'on vise une simple amélioration des processus en s'inspirant d'une norme ou que l'on doive obtenir une certification, les approches agiles ajoutent un ensemble de pratiques intéressantes. La norme ISO 9001, vise la satisfaction de la clientèle. Ce système de gestion de la qualité a été créé par un besoin de l'industrie, afin de permettre la classification des fournisseurs.

Les exigences sont relatives à quatre grands domaines

- Responsabilité de la Direction : exigences d'actes de la part de la direction en tant Qu'acteur premier et permanent de la démarche.
- Système Qualité : exigences administratives permettant la sauvegarde des acquis. Exigence de prise en compte de la notion de système.
- Processus : exigences relatives à l'identification et à la gestion des processus contribuant à la satisfaction des parties intéressées.
- Amélioration continue : exigences de mesure de performance à tous les niveaux utiles, ainsi que l'engagement d'actions de progression.

Il est possible de rejoindre les objectifs de chaque domaine avec les valeurs agiles. Mais, c'est avec le domaine du processus que les organisations peuvent définir des processus lourds, qui limitent l'intégration de l'agilité.

Le CMMi est un modèle visant l'efficience. Il a été créé pour classifier les fournisseurs de logiciel du département de la défense américaine (DoD). Il est composé de cinq niveaux de maturité.

- **Initial** (chaotique): Les facteurs de réussite des projets ne sont pas identifiés, la réussite ne peut donc être répétée.
- **Piloté** : Les projets sont pilotés individuellement et leurs succès sont répétables.
- **Standardisé** : Les processus de pilotage des projets sont mis en place au niveau de l'organisation par l'intermédiaire de normes, procédures, outils et méthodes.
- **Quantifié** : La réussite des projets est quantifiée. Les causes d'écart peuvent être analysées.
- **Optimisé** : La démarche d'optimisation est continue.

Chaque niveau de ce modèle fait progresser les processus en place, pour être capable de les optimiser. Plusieurs méthodes agiles allèguent équivaloir au niveau standardisé (niveau 3) [17; 11]. Nous retrouvons parmi les approches agiles les mêmes principes qui cherchent à améliorer le rendement et la productivité.

Le CMM étant une plate-forme, il comporte un large spectre de modèles de planification acceptables. Le type de planification proposée par les méthodes agiles peut alors cadrer dans le modèle du CMM tel qu'illustré par la figure suivante. L'indiscipline et l'improvisation du « hacking » dans la section gauche contre la planification et la gestion de jalons granulaires à la droite.

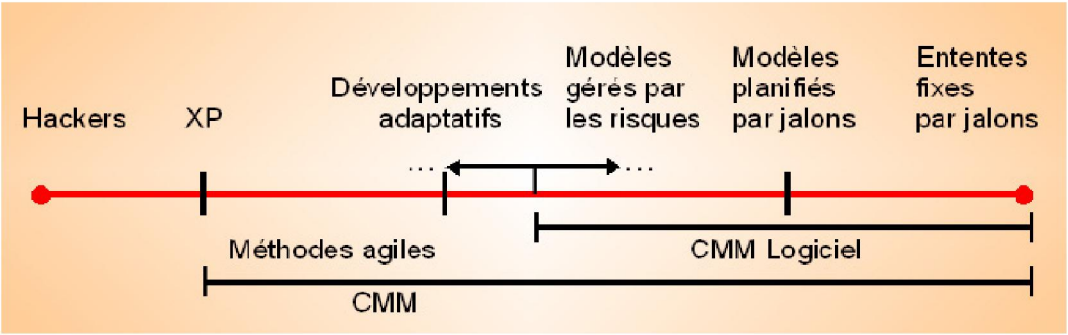

Figure 1.1 Spectre de la planification [19]

Quelle que soit son adaptation, un problème fréquent lorsqu'un processus est défini, est d'imaginer qu'il est figé puisqu'il est rédigé [17]. Les gens se contraignent à suivre la procédure, sans songer à l'adapter. Parfois, ils préfèrent croire que le processus est figé [14].

La conciliation est possible, mais nous pouvons attribuer cette méconnaissance à la perception plutôt qu'à une contrainte réelle.

4. Distinction entre les approches traditionnelles et agiles

Les méthodes agiles ont été développées en réponse à l'inefficacité et à la lourdeur des approches traditionnelles [20]. Ces projets sont planifiés minutieusement et visent à prédire leurs déroulements. À l'opposé, les approches agiles préfèrent s'adapter au cours du projet.

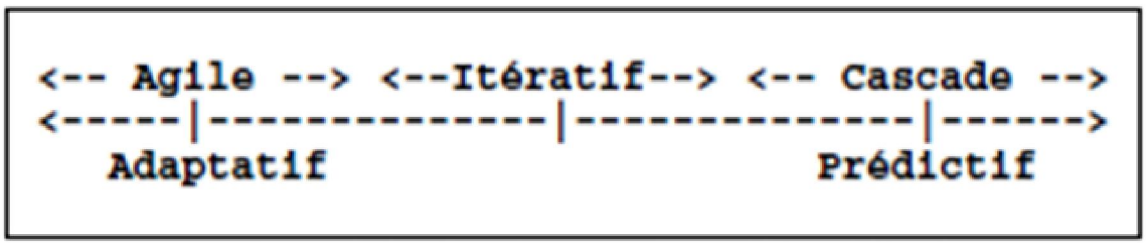

Figure 1.2 Continuum adaptatif à prédictif [18]

En imaginant un continuum (voir Figure 1.2) partant de l'adaptatif jusqu'au prédictif, nous pouvons établir que les approches agiles sont du côté adaptatif et que les méthodes traditionnelles sont du côté prédictif. Bien que cet aspect les éloigne, elles ne s'opposent pas sur tous les plans. Nous pouvons grouper les propriétés communes aux deux approches et celles qui leurs sont spécifiques.

4.1. Propriétés communes et spécifiques

Les approches agiles et traditionnelles se rejoignent sur différents aspects. Les propriétés communes et les propriétés spécifiques, qui caractérisent les deux types d'approches, sont présentées à travers différents tableaux qui touchent un ensemble de caractéristiques. [21,11,22,19]

Caractéristiques des équipes de développement	Traditionnelle	Agile
Propriétés communes	Apprendre par l'expérience. Formation et expérimentation. Intégration de nouvelles technologies.	Apprendre par l'expérience. Formation et expérimentation. Intégration de nouvelles technologies.
Grosseur des équipes	Peut aller au-delà de 20 développeurs.	Moins de 12 développeurs.
Séparation des tâches	Liées à la fonction ou à la spécialité, en fonction du livrable à produire.	Liées à l'intérêt ou à la compétence, s'appuyant sur la collaboration de l'équipe.
Communication des Spécifications	Par écrit. Définit selon la méthode et rétractable.	Par des échanges verbaux, journaliers et en personnes
Localisation des équipes	Bureaux séparés. Équipes distribuées.	Préférablement dans une aire commune.
Profil des compétences de l'équipe	Équipe multidisciplinaire. Accès à des spécialistes. Débutant. Encadré.	Personnel compétent. Autonome. Discipliné. Partage de connaissances.

Tableau 1.2 Caractéristiques des équipes de développement

Caractéristiques liées au produit	Traditionnelle	Agile
Propriétés communes	Établir les spécifications fonctionnelles. Production de la documentation utile. Rencontrer les exigences de qualité.	Établir les spécifications fonctionnelles. Production de la documentation utile. Rencontrer les exigences de qualité.
Architecture du système	Architecture optimisée en prévision des plans.	Architecture pour les besoins immédiats et émergeants.
Spécifications fonctionnelles	Rigoureuses.	Spécifications ouvertes répondants à l'objectif.
Techniques particulières de développement	Réutilisation de classes. Couplage de composants. Assemblage par étape.	Automatisation des tests. Refonte du code (refactoring).

Tableau 1.3 Caractéristiques liées au produit

Caractéristiques de la gestion de projet	Traditionnelle	Agile
Propriétés communes	Planification du projet. Études préliminaires de faisabilité. Gestion des coûts et reddition de comptes.	Planification du projet. Études préliminaires de faisabilité. Gestion des coûts et reddition de comptes.
Facteur de succès	Finir en temps et dans le budget.	Fournir une valeur ajoutée.
Type de projet	Besoins connus et stables. Incidence critique. Grand projet.	Besoins changeants. Incidence peu critique. Livraison rapide.
Modèles du processus	Modèles variés (cascade, v, etc.) visant à prédire le déroulement. Suit toujours l'ordre : Designer – coder – tester.	Modèle itératif incrémental. Adaptable selon les pratiques et l'environnement. Réalise simultanément : Designer – coder – tester.
Planification	Détaillée pour l'ensemble du projet. Diagramme de Gantt élaborant les affectations de tâches à travers le temps.	Grossière pour l'ensemble, détaillée pour l'itération. Production d'une liste de fonctionnalités et de tâches dans un tableau décroissant (burn chart).
Suivi de la progression	Respect de la planification. Approbation des livrables (dossiers, modèles, documentations).	Respect des fonctionnalités. Présentation fonctionnelle du logiciel.
Gestion du risque	Suivi plus rigoureux.	Courtes itérations favorisant le plus grand risque au départ.
Gestion des changements	Préfère les minimiser. Limitée aux éléments les plus valables, car elles sont souvent coûteuses.	Favorable à leur intégration. Intégrée au plan d'une itération.
Délais de livraison	À la fin du projet ou dès la première itération, pouvant dépasser 6 mois.	De deux semaines à quatre mois. Généralement un mois.

Tableau 1.4 Caractéristiques de la gestion de projet

Les deux approches cherchent à satisfaire les besoins du client, en trouvant les meilleures solutions possibles. La différence est au niveau des moyens utilisés pour y arriver. Il n'est pas nécessaire de suivre complètement une méthode. L'organisation doit trouver son d'équilibre en fonction de ses priorités, sa tolérance aux risques et son efficacité.

5. Description de méthodes agiles

Un certain nombre de méthodes respectent les critères du manifeste Agile. Ce sont d'ailleurs elles qui ont poussé à publier ce consensus. Certaines sont mises en pratiques et ont prouvé leur efficacité. Voici les méthodes les plus répandues :

- **RAD** (*Rapid Application Development*) est mise au point par James Martin en 1991.Même si le cycle « en spirale » permettait de revenir sur chaque phase, elle est la première méthode de développement de logiciels où le cycle de développement est en rupture fondamentale avec celui des méthodes antérieures dites « en cascade ». C'est principalement la vision en cycle itératif et à la construction d'un produit partiel mais livrable à chaque itération que **RAD** fut innovante [23].
- **SCRUM** est créée par Ken Schwaber et Jeff Sutherland en 1996. Le terme SCRUM est emprunté au rugby et signifie mêlée, Il permet de souligner que cette méthode met en avant l'aspect équipe soudée qui cherche à atteindre un but comme c'est le cas en rugby pour avancer le ballon lors d'une mêlée. Le principe de base de Scrum est d'impliquer le client comme membre de l'équipe, de le focaliser sur un ensemble de fonctionnalités à réaliser dans des itérations de durée fixe de deux à quatre semaines et de fournir un produit partiel fonctionnel à chaque itération. Par contre, la méthode Scrum ne propose aucune technique d'ingénierie du logiciel comme des normes ou pratiques de développement. Il est nécessaire de lui adjoindre une méthode complémentaire [24].
- **FDD** (*Feature Driven Development*) est proposée en 1997 par Jeff De Luca avec l'influence de Peter Coad et son approche de modélisation objet. C'est lors d'un projet ayant échoué à deux reprises dans le domaine de la restructuration bancaire à Melbourne en Australie que Jeff De Luca a pu faire ses preuves en utilisant une approche minimaliste de cinq étapes nommée FDD. Cette méthode est plus exactement un guide qu'un processus normatif qui respecte elle aussi un cycle itératif avec des livraisons régulières [25].
- **RUP** (*Rational Unified Process*) proposée en 1998 par Rational Software (IBM) est l'une des plus célèbres implémentations de la méthode UP (*Unified Process*). Même s'il en existe d'autres comme XUP (*Extreme Unified Process*) qui est une instanciation hybride intégrant UP avec Extreme Programming, ou AUP (*Agile Unified Process*) mettant plus l'accent sur le contexte du projet que sur la théorie, RUP permet plutôt de donner un cadre au développement logiciel afin d'allier philosophie Agile et modélisation répondant aux exigences fondamentales préconisées par les créateurs d'UML (*Unified Modeling Language*) [26].
- **XP** (*eXtreme Programming*) a été mise en place par Kent Beck, en collaboration avec Ward Cunningham et Ron Jeffries sur un projet pour la compagnie Chrysler dans les années 1990. C'est en 1999 qu'est officialisée la méthode XP avec la sortie du livre

«*eXtreme Programming explained*». Elle recherche l'efficacité maximale en concentrant l'effort de travail sur l'objectifde développer le bon logiciel et de ne pas s'égarer. La démarche est légère, pragmatique, disciplinée, empirique et adaptative. Celle-ci correspond très bien à la couche orientée implémentation complémentaire à SCRUM car la philosophie est exactement la même [27].

6. Conclusion

L'agilité se définit dans le domaine logiciel par sa capacité d'adaptation et son attention aux individus. Le manifeste intègre l'ensemble des valeurs priorisées par ce mouvement de pensée. L'agilité est un changement de paradigme, qui impacte sur la culture organisationnelle. Elle représente un défi, car elle implique à la fois les développeurs, les gestionnaires et les clients. Ultimement, elles visent à satisfaire le client par un produit fonctionnel adéquat aux besoins réels.

Différents intervenants tirent des avantages de ces approches. Comparativement aux approches traditionnelles, les approches de type agile veillent à ce que les efforts investis maximisent le bénéfice du client ; que le produit soit adapté aux besoins qui seront découverts au cours du développement ; que les gens impliqués dans le projet soient motivés et heureux d'y participer, et finalement que le projet soit bien suivi pour limiter les risques.

Les approches agiles viennent élargir l'offre méthodologique. Elles émergent de l'évolution du domaine logiciel qui précise son identité. Elles s'harmonisent avec l'amélioration des technologies et l'accessibilité du domaine au grand public. Les organisations n'ont pas toutes besoin d'être agiles. Par contre, elles ont toutes besoin de savoir quelle est l'offre des approches agiles, afin qu'elles puissent se positionner.

CHAPITRE 2

LA METHODOLOGIE SCRUM

Ce chapitre présente l'historique de SCRUM, et un bref tour d'horizon. Il permettra d'apporter les informations nécessaires pour une meilleure compréhension de processus de développement SCRUM et les rôles des intervenants, et finalement les avantages et les inconvénients de la méthodologie.

1. Introduction

La méthodologie SCRUM est un processus de développement qui demande une coordination entre le client, les équipes de développement et la direction. Ce processus exige d'implanter des méthodes de travail et de communication qui sont souvent nouvelles pour bien des individus. Elle requiert du client de s'impliquer activement dans le projet et à l'équipe de changer ses méthodes de travail. Ainsi, plus l'entreprise est grande, plus il peut être difficile d'implanter SCRUM.

SCRUM présente une solution intéressante pour les grandes entreprises qui aimeraient gagner en flexibilité. En utilisant une méthode évolutive de développement qui implique une plus grande participation du client dans le processus de développement, les deux parties voient leurs chances de succès augmentées proportionnellement à la qualité de leurs communications et de leurs relations. En utilisant la méthodologie SCRUM dans les grandes entreprises, la qualité du logiciel est accrue et les nouveaux besoins commandés par la réalité changeante du client sont considérés tout au long du processus. Une synergie qui gagnerait à être reconnue.

2. Historique de SCRUM

En 1993, Jeff Sutherland [28] et Ken Schwaber ont fait l'analyse des processus de développement logiciel de l'époque. Leurs conclusions étaient les suivantes : les processus analysés ne convenaient plus à la méthode empirique et devenaient imprévisibles ce qui les empêchaient d'être répétés.

Le **tableau 2.1** suivant présente les résultats de leur étude comparative des méthodologies de développement logiciel existant soit : cascade, spiral, itératif et SCRUM.

	Cascade	Spirale	Itératif	**Scrum**
Processus défini	Requis	Requis	Requis	**Planification et fin de projet Seulement**
Produit final	Déterminé durant la planification	Déterminé durant la planification	Défini durant le Projet	**Défini durant le Projet**
Cout de projet	Déterminé durant la planification	Partiellement variable	Défini durant le Projet	**Défini durant le Projet**
Date de fin de projet	Déterminé durant la planification	Partiellement variable	Défini durant le Projet	**Défini durant le Projet**
Adaptable a l'environnement flexibilité et créativité de l'équipe	Limité – approche livre de recettes	Limité – approche livre de recettes	Limité – approche livre de recettes	**Illimité durant les Itérations**
Transfert de connaissance	Formation avant le début du projet	Formation avant le début du projet	Formation avant le début du projet	**Formation entre les membres de l'équipe durant le projet**
Probabilité de Succès	Faible	Moyen à faible	Moyen	**Élevé**

Tableau 2.1 Comparaison des méthodologies de développement [28]

Jeff Sutherland et Ken Schwaber, en s'appuyant sur l'analogie du jeu du rugby, ont alors proposé un nouveau processus de développement logiciel : le SCRUM.

L'analogie du SCRUM au rugby a été utilisée pour la première fois par Takeuchi et Nonaka [29] qui ont publié une étude dans le « Harvard Business Review ». Dans cette étude les auteurs comparaient les équipes performantes et multifonctionnelles aux mêlées utilisées par les équipes de rugby.

Le SCRUM est un processus itératif et incrémental qui s'adapte à la réalité des projets de développement logiciel. Le processus SCRUM a été ensuite combiné aux principes de la programmation extrême proposée par Mike Beedle. Cette version de SCRUM est la première qui a été présentée à la conférence internationale OOPSLA (« Object-Orientes Programming, Systems, Langages, and Applications ») de 1995.

En 2001, Ken Schwaber et Mike Beedle ont publié le premier ouvrage de référence sur le sujet intitulé : « Agile Software Development with Scrum ».

Finalement, c'est en février 2008 que Ken Schwaber, Mike Cohn, et Esther Derby ont fondé l'alliance SCRUM, une organisation sans but lucratif. Cette organisation a pour but et de soutenir les utilisateurs intéressés à SCRUM pour réussir l'implantation et l'utilisation de cette méthode par la promotion de publication d'articles, de ressources, de formations et d'événements s'y rattachant. Elle a pour mission de sensibiliser et de soutenir l'amélioration itérative de la profession du développement logiciel.

2.1. Tour d'horizon de SCRUM

SCRUM est une méthodologie agile qui permet de livrer un logiciel plus rapidement avec plus de qualité. SCRUM permet au client d'un logiciel développé de contrôler la qualité du travail à effectuer tandis que l'équipe contrôle la quantité de travail effectué. Cette méthodologie permet de s'adapter rapidement aux changements d'un client puisqu'à fréquence régulière (chaque fin d'itération) l'équipe et le client réévaluent les spécifications du logiciel. Ainsi, le client reçoit les spécifications qu'il a demandées plus souvent (jusqu'à une possibilité d'un livrable par semaine) ce qui ne fait qu'accroître sa satisfaction. Les spécifications développées sont toujours les plus pertinentes pour le client, car elles sont réévaluées avant d'être entamées. Ce qui permet d'optimiser les ressources de développement selon les besoins réels du client.

SCRUM pourrait se résumer par deux actions soient inspecter et adapter. Ces actions, à elles seules, décrivent presque toutes les situations rencontrées durant l'utilisation de cette méthodologie.

La méthodologie SCRUM propose les intervenants suivants :

- Le gestionnaire de produit (« Product Owner »);
- Le maître SCRUM (« Scrum Master »);
- L'équipe de développement

La méthodologie SCRUM propose les activités suivantes :

- Planification d'itération (« Sprint Planning »);
- Revue d'itération (« Sprint Review »);
- Rencontre/mêlée quotidienne (« Daily Scrum Meeting »).

La méthodologie SCRUM propose la création d'artefacts :

- Le carnet de produit (« Product Backlog »);
- Le carnet d'itérations (« Sprint Backlog »);

- Le graphique d'avancement (« Burn Down Chart »).

La figure 1 donne un aperçu de la méthodologie SCRUM dans son ensemble qui sera traitée dans les prochains chapitres.

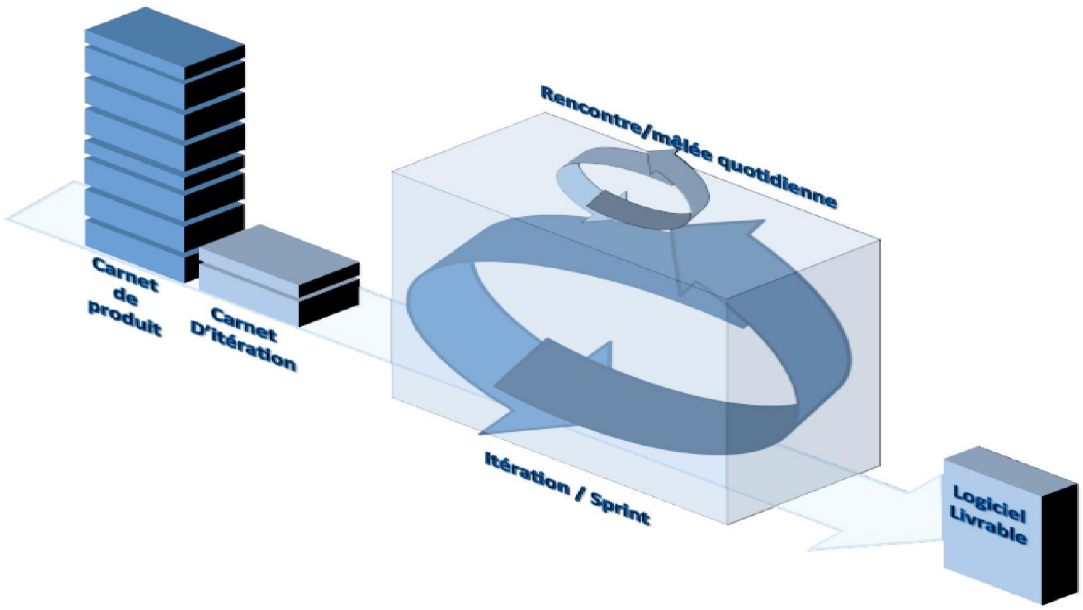

Figure 2.1 Cycle de vie de SCRUM [28]

3. Définition du processus de développement

3.1. Avant de débuter votre première itération

Avant de démarrer une première itération, la méthodologie SCRUM demande de créer le carnet de produit. Le carnet de produit représente la liste des spécifications logicielles classées selon leur valeur d'affaires respectives.

Avant de poursuivre, définissons ce qu'est la « valeur d'affaires » selon SCRUM. La valeur d'affaires est la valeur attribuée à la spécification qui apportera le plus de retour sur investissement après son développement pour le logiciel. C'est une mesure qui quantifie un ensemble de facteurs déterminés par le client et l'entreprise. Par exemple : un logiciel de traitement de texte qui n'aurait pas la possibilité de créer de nouveaux documents est impensable, c'est pour cette raison que cette spécification a une grande valeur d'affaires et serait la première à être développée comparativement à celle qui permet de changer la couleur du fond de l'écran.

Le processus itératif de la méthodologie permet de rendre un rapport de causalité inversement proportionnel de la valeur d'affaires en vertu de la durée des itérations. La **figure 2.2**, démontre cette relation sans toutefois contenir de données statistiques réelles. [28]

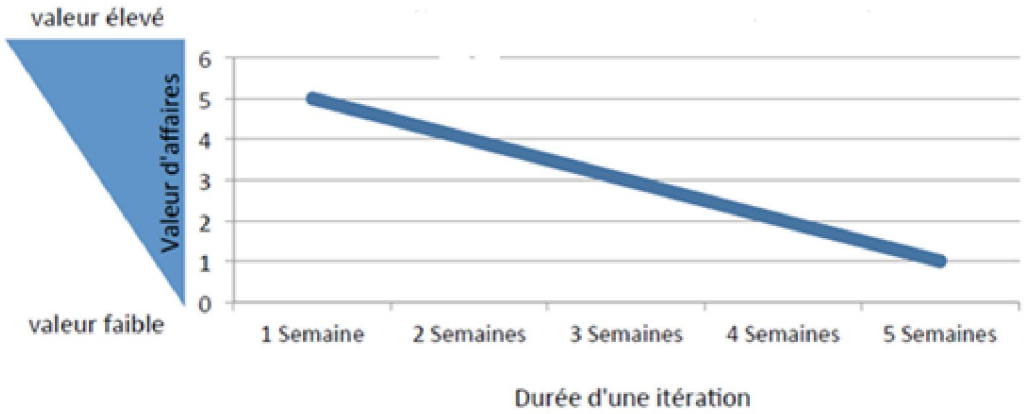

Figure 2.2 valeur d'affaires en fonction de la durée de l'itération [28]

La liste des spécifications présentées dans le carnet de produit peut être répertoriée différemment soit :

- Selon les spécifications du client ou domaine d'affaires
- Autrement, sera désignée comme Liste d'histoires (« User Stories»).

- Permettre la création de compte utilisateurs.
- Permettre l'ajout de produit dans le panier d'épicerie.
- Permettre de payer en ligne.
- Permettre de changer les couleurs.

Le gestionnaire de produits devra faire l'exercice de prioriser ses demandes selon des critères respectant la mission et les objectifs de son logiciel. En précisant la valeur d'affaires, il estime l'impact et le retour sur investissement qu'aura chacun des items dans le carnet de produit. Comme les domaines d'affaires varient et qu'ici nous nous adressons spécifiquement à de grandes entreprises, il est recommandé d'utiliser des indicateurs simples qui pourront être utilisés par l'équipe de développement afin d'évaluer la fonctionnalité. Par exemple : bas, moyen, élevé et très élevé.

Les spécifications prioritaires listées dans le haut du carnet de produits doivent être :

- Assez précises pour être estimées par l'équipe;
- Assez petites pour être testées et développées durant une itération.

La priorisation des spécifications du carnet de produit doit être fait attentivement en se rappelant les objectifs de l'implémentation de cette fonctionnalité. Cette codification permettra d'optimiser les ressources de développement afin que le client ait un retour sur investissement le plus tôt possible sur sa décision. Ce processus permet d'établir, entre autres, le plan de livraison du produit.

Les spécifications devront ensuite être estimées, l'ensemble des spécifications n'a pas besoin d'être entièrement estimé pour débuter une itération. L'exercice d'estimation peut être interrompu après l'équivalent de 10 jours de travail de développeur estimé à l'avance. Le fait d'estimer davantage de spécifications pourrait entrer en conflit avec l'effort d'optimisation déployé puisqu'il ne tient pas compte de la pertinence que le client lui accordera en temps voulu, le moment venu. L'idée reste toujours ici de faire profiter le client d'un retour sur investissement le plus tôt possible dans le développement de sa solution.

Le carnet de produit n'est pas un document final. [28]

Le gestionnaire de produit, conserve le contrôle et le pouvoir d'agir en tout temps. Il peut ajouter, modifier ou effacer une spécification dans le carnet de produit à n'importe quel moment du développement. Ce qui lui permet de communiquer ses craintes et les changements que vit l'entreprise qu'il représente. Il conserve le droit de veto sur l'ordre d'exécution et peut ainsi répondre en temps réel aux changements inattendus qui surviendraient dans l'entreprise en offrant des outils tangibles et accessibles rapidement.

3.2. Planification d'itération

Une fois que le carnet de produit est avancé à un taux satisfaisant, on passe à la planification d'une itération.

Il est important de rappeler que les premières spécifications du carnet de produit doivent être :

- Assez, précise pour être estimé par l'équipe;
- Assez petite pour être testé et développé durant une itération.

Lors de cette planification, le gestionnaire de produit présente les spécifications logicielles qu'il désire inclure dans l'itération courante. C'est à ce moment que le gestionnaire de produit peut délibérément inclure une spécification qui serait devenue prioritaire suite à une nouvelle situation urgente; pertes de ressources et de connaissances inattendues qui presseraient l'entreprise à implanter une spécification répondant à ce besoin dans les meilleurs délais.

Nous ne travaillons pas les uns pour les autres, mais les uns avec les autres. – [STANLEY C.GAULT]

L'équipe évalue le nombre d'heures auquel ils se sentent en mesure de s'engager. Pour trouver ce nombre, il s'agit de demander à chacun des membres de l'équipe, d'évaluer le temps qu'il croit pouvoir s'engager à travailler sur cette itération par jour.

3.3. Mêlée quotidienne

La mêlée quotidienne est un moment privilégiée durant la journée qui permet à tous les membres de l'équipe de faire le point sur les réalisations de chacun depuis la dernière mêlée quotidienne. Néanmoins, dans les équipes moins expérimentées, le maître SCRUM soutient cette réunion afin de faciliter les discussions.

Avant le début de la réunion, les membres de l'équipe se préparent à répondre brièvement à 3 questions.

- Quel est le travail effectué depuis la dernière mêlée quotidienne dans le cadre du projet (et hors projet)?
- Quelles sont les tâches qu'il s'engage à faire d'ici la prochaine mêlée quotidienne dans le cadre du projet (et hors projet)?
- Quels sont les éléments susceptibles d'empêcher un des membres d'atteindre ses objectifs avant la prochaine mêlée?

La mêlée quotidienne dure, tout au plus, 15 minutes. Cette courte rencontre se fait debout, en cercle, et l'ensemble de l'équipe y participe incluant le gestionnaire de produit. La position debout permet de conserver le dynamisme et l'accent sur l'objectif de la réunion. L'ensemble des participants doit s'abstenir de poser des questions durant cette réunion. Ils doivent plutôt les garder pour après la mêlée quotidienne. À la fin de cette réunion, l'ensemble des membres de l'équipe doit être capable de brosser un statut d'achèvement et les étapes en cours d'exécution par ses collègues jusqu'à la prochaine mêlée. [29]

Le maître SCRUM, en plus de préparer ses réponses pour les trois questions se doit de préparer la réunion. Il s'assure de la liste du statut d'achèvement des tâches débutées, celles ré estimées ou même celles ajoutées. L'ensemble de ces informations lui permet de mettre à jour le graphique d'avancement qu'il peut utiliser dans la mêlée quotidienne. (Voir la **figure 2.3** et la **figure 2.4**).

Durant la mêlée quotidienne, il doit soutenir son équipe en priorisant et en éliminant les éléments bloqueurs. Le maître SCRUM est aussi en charge de gérer les problèmes interpersonnels dans l'équipe afin de maintenir la productivité et l'efficacité de l'équipe à son plus maximum.

3.4. Le graphique d'avancement

Le graphique d'avancement permet à l'équipe de connaître l'état d'avancement de l'itération. Ce graphique démontre le nombre d'heures restant à effectuer d'ici la fin de l'itération. Quand une tâche est accomplie, son temps est soustrait du temps restant à effectuer. Ce graphique est mis à jour quotidiennement et permet d'évaluer si la performance de la vélocité de l'équipe. Une itération est considérée réussie quand le temps restant est égal à zéro.

Le carnet d'itérations présenté par le graphique d'avancement est établi dans la planification d'itération, mais peut être modifié durant l'itération.

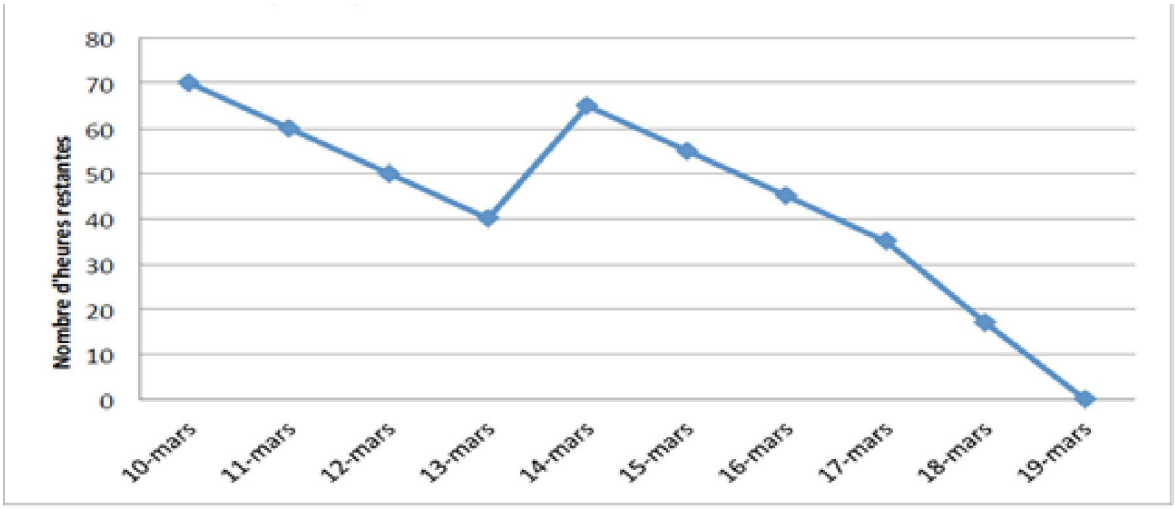

Figure 2.3 Exemple de graphique d'avancement simple

Plusieurs facteurs peuvent affecter le carnet d'itérations en voici quelques un :

- Sous-évaluation/surévaluation du temps requis par une spécification développée;
- Des tâches supplémentaires pour résoudre des problématiques identifiées par le gestionnaire de produit.
- Une séance de travail avec le gestionnaire de produit pour redéfinir la compréhension générale et l'objectif de l'itération.

Ces changements seront considérés par l'équipe et le gestionnaire de produit, s'ils sont nécessaires, mineurs et peuvent être inclus dans le temps alloués sans compromettre l'itération courante.

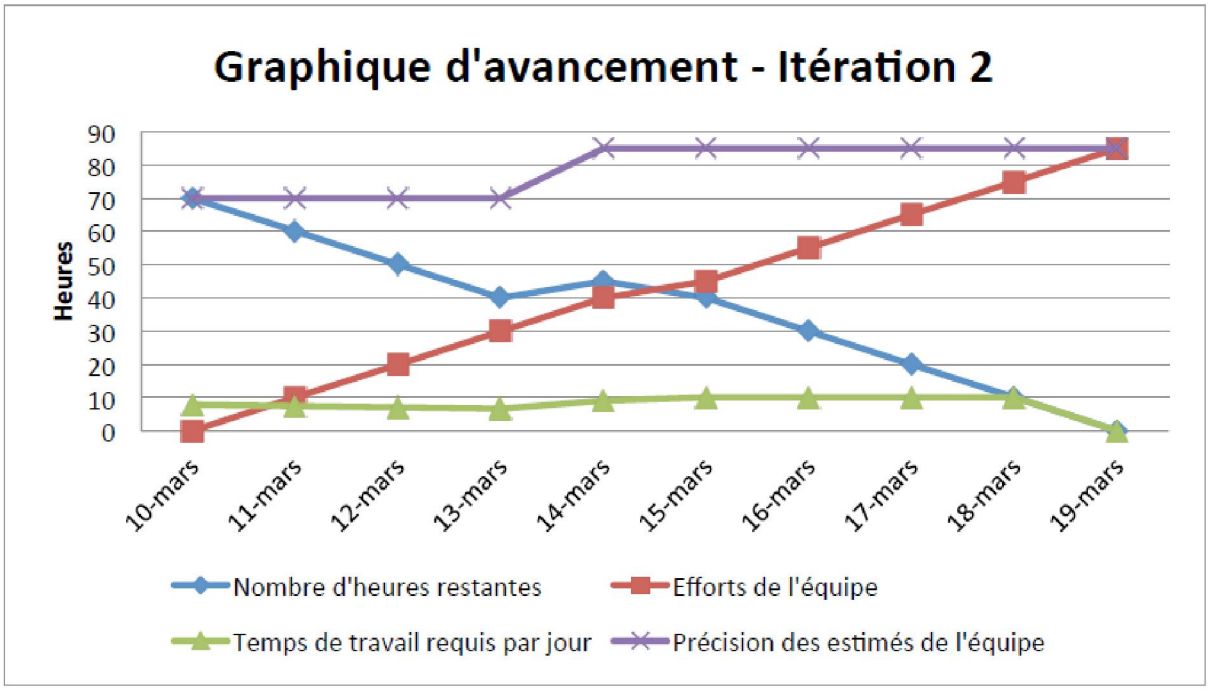

Figure 2.4 Exemple d'un graphique d'avancement plus complexe

3.5. Rencontre de revue d'itération

La rencontre de revue d'itération est effectuée à la fin de chaque itération. Elle est divisée en deux parties et dure environ 4 heures.

Durant la première moitié de la rencontre, le gestionnaire de produit présente à toutes les personnes concernées les spécifications complétées du carnet d'itération. Par la suite, le gestionnaire de produit entame une revue des prochaines spécifications pour l'itération suivante.

Pour la deuxième partie, le maître SCRUM anime une rétrospective de la dernière itération. Il passe en revue les bons et les moins bons coups de l'équipe, afin d'améliorer les pratiques de celles-ci pour les itérations futures. Lors de cette revue, l'équipe apporte des solutions aux difficultés rencontrées. Cette rencontre permet à chacun des membres de l'équipe de tirer parti du passé sans compromettre la vélocité de l'équipe. Inspecté et adapté, ici l'application des principes SCRUM se fait non seulement au développement, mais aussi aux principales ressources de production; les membres de l'équipe.

L'ensemble de la méthodologie SCRUM est défini dans la **figure 2.5.**

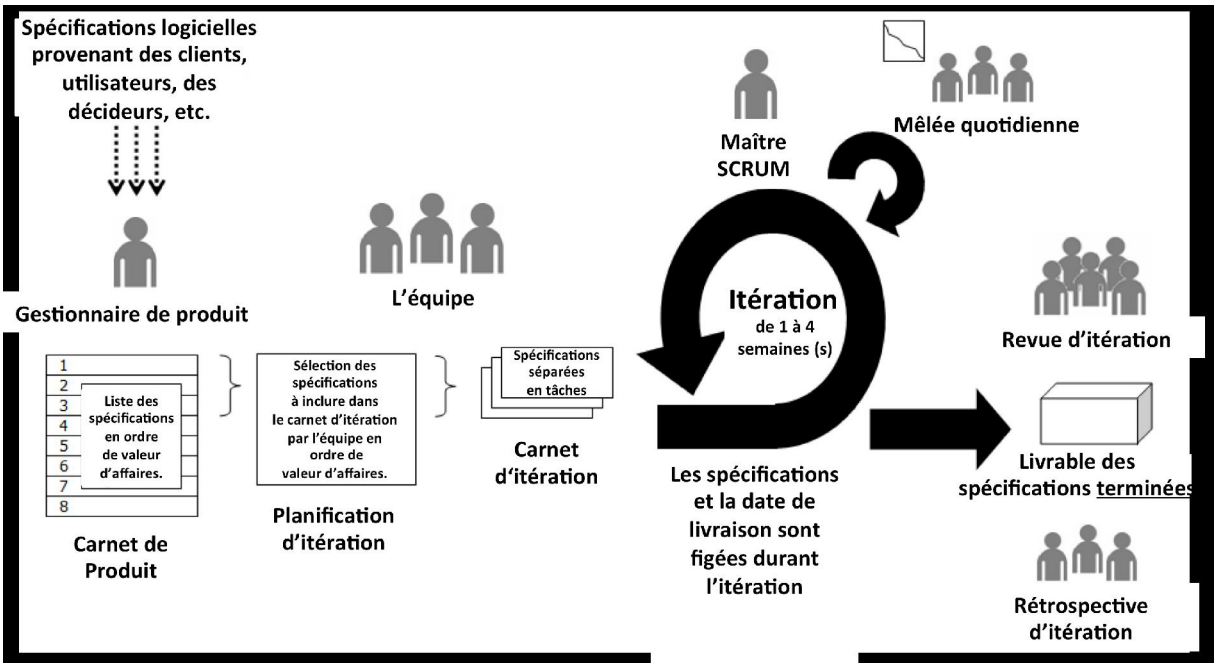

Figure 2.5 Processus SCRUM [30]

3.6. SCRUM de SCRUM

Lorsqu'un projet est trop gros pour être réalisé par une seule équipe, il est alors divisé et distribué à plusieurs équipes. Une équipe SCRUM est fonctionnelle lorsqu'elle est limitée à moins de 10 membres. Ainsi, la responsabilité de réalisation est distribuée par autant de membres que composent les équipes incluant un maître SCRUM par équipe.

Le SCRUM de SCRUM consiste à une mêlée quotidienne entre les maîtres SCRUM d'un même projet. Cette activité n'est pas aussi fréquente que la mêlée quotidienne elle peut arriver qu'une fois par semaine ou plus souvent selon les besoins du projet. Lors de cette activité, les maîtres SCRUM doivent répondre à ces questions :

- Qu'est-ce que votre équipe a fait depuis notre dernière mêlée?
- Qu'est-ce que votre équipe fera d'ici la prochaine mêlée ?
- Est-ce que votre équipe a rencontré des problèmes depuis la dernière mêlée ?
- Allez-vous ajouter des éléments qui pourraient impacter une autre équipe d'ici la prochaine mêlée?

Le SCRUM de SCRUM n'est pas identifié à la figure 2.4 puisqu'il s'agit d'une répétition de la méthode originale.

4. LES RÔLES

4.1. Le gestionnaire de produit

Le gestionnaire de produit est responsable de la vision du produit demandé par le client. Il est l'intermédiaire traduisant les besoins du client vers l'équipe SCRUM et vice versa. Il détient un droit de veto sur les spécifications développées et tout le succès de l'opération repose sur ses talents de communicateurs ainsi que ses compétences. Par exemple, il est le seul à déterminer si la spécification est bel et bien complétée et si l'équipe passe à la spécification suivante du carnet suivant ou une autre de son choix indépendamment des choix de l'équipe SCRUM.

Important : Le rôle de gestionnaire de produit doit être tenu par une seule personne par équipe SCRUM afin d'éviter des situations ambiguës. Dans le cas où le projet utilise plusieurs équipes, les gestionnaires de produit de toutes les équipes doivent se synchroniser afin d'avoir la même vision pour le produit.

4.2. Le maître SCRUM

Le maître SCRUM soutient l'équipe afin de garantir les meilleures chances de succès des engagements pris par celle-ci. Sa tâche se résume à résoudre les bloqueurs techniques, humains et matériels qui surviendront durant le développement. Le maître SCRUM, malgré son expérience, ne devrait pas être perçu comme un superviseur au sens classique du terme.
Tout au plus un excellent technicien qui voit au bon fonctionnement et au respect des normes à suivre. Il est souvent un facteur déterminant dans l'appréciation de la méthode SCRUM auprès de l'équipe qu'il soutient.

Ce rôle peut être tenu par un membre de l'équipe, un consultant ou un spécialiste à la seule exception du gestionnaire de produit. Dans ce dernier cas, la crédibilité et l'engagement du maître SCRUM pourraient en souffrir lors de décisions importantes à prendre.

4.3. L'équipe

L'équipe est responsable de développer le produit logiciel commandé par le gestionnaire de produit. Elle est constituée de 5 à 10 personnes ayant l'expertise nécessaire pour assurer la réalisation du produit. Au-delà de ce nombre, l'équipe perd de son efficacité et de sa flexibilité particulièrement lors des mêlées quotidiennes. Dans un tel cas, il est plutôt conseillé de se subdiviser en plus petites équipes et de fonctionner avec des SCRUM de SCRUM [31].

5. Les avantages et les inconvénients de la méthodologie Scrum

5.1. Avantage

- Méthode fun !

- Excellente réactivité vis-à-vis du client (product owners => PO)
- Réduction des risques, plus près de la réalité.
- Qualité accrue du fait de la présence des PO
- Favorise et facilite la communication avec les autres membres de l'équipe
- Rapidité et efficacité.

5.2. Inconvénients

- Gestion obligatoire de la communication.
- Mauvaise visibilité à long terme.
- Intégration continue.
- Les développeurs doivent accepter le changement (revenir sur du code pour le modifier voir le supprimer).

Changement radical de gestion de projet/produit : fort impact sur tous les intervenants [32].

6. Conclusion

Certains prétendent que SCRUM pourrait se voir étendre à divers secteurs de l'entreprise. C'est un retour aux sources, car à son origine, SCRUM ne s'adressait pas au développement logiciel [30].

Puisant ses racines au milieu des années 1980, SCRUM amène une dynamique différente de la gestion de projets des développements logiciels. Elle propose un équilibre entre la portion définie et la portion chaotique du développement de produit. Cette méthode ne précise pas de technique de réalisation précise. Elle laisse cet aspect à la discrétion de l'équipe. Cette ouverture permet l'inclusion de pratiques provenant de d'autres méthodes.

En priorisant sur la valeur ajoutée par chaque composante du produit, SCRUM assure que ce dernier est toujours la meilleure valeur d'affaire possible pour l'investissement du client.

En détachant l'aspect logiciel et en portant cette approche à d'autre domaine d'ingénierie, SCRUM se permet de porter l'agilité aux autres secteurs de développement de produits.

CHAPITRE 3

LES PARAMETRES GREEN COMPUTING

Ce chapitre présente la définition de Green Computing, les déférentes stratégies Green pour le hardware et le software, et enfin les Domaines d'action permettant à une entreprise d'intégrer une démarche de green IT.

1. Définition de Green Computing

Green Computing est une discipline qui étudie, développer et promouvoir des techniques d'amélioration de l'efficacité énergétique et la réduction des déchets dans le cycle de vie complet des équipements informatiques de la fabrication initiale, grâce à la livraison, l'utilisation, l'entretien, le recyclage et l'élimination d'une manière économiquement réaliste [33,34].

Alors qu'il est décourageant d'examiner les façons dont l'utilisation généralisée des ordinateurs contribue à perdre, il est encourageant de reconnaître que les chercheurs et les éducateurs en sciences informatiques peuvent trouver des solutions pour réduire ce gaspillage.

Haute performance et l'économie d'énergie sont des objectifs contradictoires dans Green Computing. Une façon d'économiser de l'énergie est de réduire les tensions logiques Cependant, cela provoque des circuits lents et des fréquences basses, ce qui conduit à une dégradation des performances .Au cours des dernières années, le Cloud computing a gagné beaucoup de popularité car il réduit le temps d'exécution d'un programme en le distribuant sur différentes machines sur un réseau. Pour résoudre un grand problème de calcul intensif, un cluster de plusieurs machines de faible capacité est plus bénéfique en termes de coût et de performance par rapport à une machine à haute capacité. Parce qu'une machine à haute capacité n'utilise pas pleinement ses ressources à la fois, d'où il consomme plus d'énergie. Par conséquent, l'amélioration significative des performances avec la conservation de l'énergie peut être obtenue par le biais distribué ou cloud computing parmi verte mise en œuvre des stratégies.

Optimisations pour la conservation de l'énergie peuvent être réalisées à des niveaux matériels et logiciels. Hardware optimisations énergétiques sont atteints grâce à la conception de circuits, des circuits auto détection de repos et des techniques actives et de polarisation .

Le niveau software d'optimisation de l'énergie est mis en œuvre dans le système d'exploitation grâce à des techniques de planification green qui analysent les processus actifs

pour les besoins énergétiques, et par les compilateurs green à travers l'analyse du programme au moment de la compilation et le code remodelant lors des transformations.

L'énergie peut aussi être conservée, au cours du cycle de vie du développement logiciel telles que l'analyse de logiciels et de conception, en appliquant différents aspects green mentionnés plus loin dans les sections suivantes. En outre, l'utilisation de l'énergie des outils logiciels conscients peut aider dans la réalisation de l'optimisation de l'énergie au niveau du logiciel.

2. Les stratégies Green Computing

2.1. stratégies green pour les compilateurs

Compilateurs conscients de l'énergie analysent les programmes de logiciels au moment de l'exécution et de remodeler le logiciel code source en appliquant plusieurs aspects green pendant la transformation de code.

Voici quelques techniques greens qui peuvent être appliquées au niveau de la procédure locale, mondiale ou inter faire programme d'énergie au courant [35].

2.1.1. Cache skipping

Dans la programmation, les boucles d'environnement ont une importance significative. Dans les boucles, la réplication donne des performances élevées, mais provoque la consommation de haute énergie due à la répétition de la même chose. Une bonne approche peut être sautée des opérations de cache lors de la réplication inutile.

Une technique efficace pour résoudre le problème du cache-skipping par une modification dans le compilateur et le matériel.

Dans cette technique le compilateur a besoin de séparer les blocs qui a moins de chance d'exécuter pour le bloc exemple d'exception. Cette étude présente que dans le cas idéal, il n'y a pas d'utilisation du cache et donc cette technique se traduit par une consommation d'énergie réduite.

2.1.2. Use of register operands

Chaque machine a une consommation différente d'énergie pour accéder aux ressources de la mémoire. La plupart des études montrent que la mémoire des lectures et écritures ont un coût plus élevé par rapport à l'utilisation des opérandes de registre. L'inscription opérandes ont moins abstraction que la mémoire des accès (lecture / écriture) et donc consommer moins d'énergie.

2.1.3. Instruction clustering

Certains environnements sont de type spécial de l'architecture qui permet à un compilateur pour exécuter pair ou cluster d'instructions en un seul cycle. Par exemple dans les applications de traitement du signal, un groupe de signaux connexes ou similaires peut être compilé en un seul passage. Il permettra de réduire le temps d'exécution du programme et conduit à la conservation de l'énergie.

2.1.4. Instruction re-ordering and memory addressing

Parfois, l'ordre des instructions et d'adressage de mémoire ne sont pas en faveur du mode sans échec de l'énergie. La consommation d'énergie peut être réduite de manière significative en modifiant l'ordre d'instruction et à partir du mode de mise sous tension en toute sécurité.

2.1.5. Use of energy cost database

Compilateurs conscients de l'énergie à maintenir la base de données de coût de l'énergie pour chaque transaction / instruction. Cette base de données peut être utilisée pour analyser le code et les algorithmes de génération d'arbres. Dans la première série de traitement de code lors de la compilation, tous les arbres d'analyse possibles sont générés et leur coût énergétique respectif est affecté en utilisant la base de données de coût de l'énergie. Dans la course suivante, moins coût de l'arbre sera sélectionné pour une compilation. Ce mécanisme permet la sélection de l'arbre optimisé des coûts énergétiques.

2.1.6. Loop optimization

Plusieurs techniques sont présentées pour l'optimisation de la boucle pour accroître la sensibilisation et l'efficacité énergétique dans le programme. L'un d'eux est la boucle Fission. Dans cette technique, des boucles généralement imbriquées sont vérifiées à travers la dépendance graphique. Dépendance graphique sont préparés pour le corps de la boucle dans laquelle les nœuds représentent les états et le bord correspond à la dépendance de données.

S'il n'y a pas de cycle dans le graphique, le compilateur va créer la boucle pour chaque déclaration et les exécuter en parallèle en utilisant le traitement entrelacé.

2.1.7. Dynamic power management

Consommation en Complementary Métal-Oxide Semi-conducteurs (CMOS) est classée en statique et dynamique. La consommation électrique est dynamique lorsque le circuit est en état de fonctionnement et aucune fuite de courant ne se produit. Alors que la consommation d'énergie est statique, lorsque le circuit ne sera pas en forme en cours d'exécution, mais il est toujours sous tension. Le système de gestion de puissance dynamique définit la puissance de son matériel en temps vrai, sans dégradation des performances pour diminuer la perte d'énergie probable.

2.1.8. Resource hibernation

Hibernation est le processus de l'utilisation du mode de faible puissance. Comme mentionné dans la section I, des ressources d'inactivité peut être maintenu dans l'état mise en veille prolongée, mais le passage à partir de cet état peut être un gaspillage de ressources précieuses et de temps.

2.1.9. Cloud aware task mapping

Couverture cartographie de la tâche consciente est l'utilisation des services prêts à l'emploi qui peuvent être fournis par les différents nuages. Une technique de compilation utilise les services de cloud au niveau de l'hôte pour un éventuel calcul par le traitement parallèle, et la tenue de registres d'état.

Une machine compilateur indépendant peut également utiliser tous les services de nuages à distance, et pendant la progression de ces services il peut aller en mode hibernation. Cette technique souffre de plusieurs problèmes; l'un d'eux est le coût de la machine virtuelle et le coût de la migration des machines hôtes. Le deuxième problème est l'échec des machines du réseau provoquant retards d'exécution ou d'une autre utilisation des services.

2.1.10. Eliminate recursion

Compilateur exécute les procédures récursives utilisant la pile. Parfois, cette technique prend beaucoup d'espace et de temps causent des dégradations de la performance ainsi que la consommation d'énergie supplémentaire. Certain compilateur convertit récursivité en itération. Cette technique peut faire gagner du temps et de l'énergie dans certains cas.

2.2. Stratégies green pour les softwares

L'énergie peut être conservée au cours du cycle de vie du développement logiciel telles que l'analyse du logiciel, la conception et la mise en œuvre. Au niveau de la conception, l'énergie peut être conservée en faisant l'énergie structure efficace du logiciel. Logiciel Implémenter développeurs peuvent utiliser des stratégies suivantes lors de la mise en œuvre du logiciel [36].

2.2.1. Use of green IDE& compiler

Utilisation de l'énergie consciente ou compilateurs Green aide à économiser l'énergie. Plusieurs open source et l'énergie sous licence compilateurs sont disponibles au courant par exemple compilateur Green Hill pour C et C ++, l'énergie compilateur courant DECC pour C ++.

2.2.2. Use of grid & cloud computing

Grille & Cloud Computing sont les grandes tendances qui rendent l'utilisation des ressources informatiques prêtes à la demande. Il y a beaucoup de ressources logicielles et

matériel disponible en tant que service par différents fournisseurs pour le convertisseur exemple de monnaie, calculatrices, etc. L'utilisation de ces ressources prêtes dans le programme sera bénéfique en termes de coût, de temps et d'énergie.

2.2.3. Les running time

En général, toute stratégie qui peut réduire le temps d'exécution de l'algorithme peut être utile pour réduire la consommation d'énergie. La Complexité de l'algorithme peut être calculée en termes de notations Big O. Les algorithmes qui ont la complexité linéaire seront au courant plus d'énergie par rapport à ceux qui ont équation exponentielle.

2.2.4. Use of energy aware data structure

Les structures de données ont un effet significatif dans l'exécution d'une conservation du programme et de l'énergie en tant que structure de données efficace est plus conservatrice de l'énergie. L'étude en [36] montre que le tri par fusion consomme moins d'énergie avec une structure de données de réseau alors qu'il consomme plus d'énergie dans le cas de la liste de liens structure de données. En outre, la compilation de diverses structures de données API telles que la liste des liens, les matrices, etc.…

Avec un peu d'énergie compilateur conservatrice rendent ces données sur l'énergie de la structure consciente, et l'utilisation d'entre eux dans le programme peut conduit la conservation de l'énergie.

2.2.5. Réduire le nombre de serveurs

Réduire le nombre de serveurs. Il est évident, de faire vos serveurs travaillent plus dur en fournissant de multiples services. Par exemple, deux serveurs redondants consomment moins d'énergie que huit petits serveurs individuels avec un seul service chacun.

2.2.6. Outils de gestion de serveur

Utiliser des outils de gestion de serveur qui réduisent la consommation d'énergie sous des charges légères. Renseignez-vous sur les outils offerts par votre fournisseur préféré et les utiliser ensuite. Achat d'équipement qui a des composants et / ou des caractéristiques respectueuses de l'environnement.

Alimentations, ne sont pas les seuls éléments qui ont des cotes écologiques. Lorsque cela est possible, d'utiliser la nouvelle énergie caractéristique efficace lors de l'achat du nouveau matériel.

2.2.7. La virtualisation

Il existe deux types de virtualisation : serveur et client.

- La virtualisation des serveurs est le masquage des ressources du serveur (tels que les serveurs individuels physiques, processeurs et systèmes d'exploitation) des utilisateurs du serveur.
- La virtualisation Client, ou bureau, est la séparation du bureau de l'ordinateur personnel (tels que des applications, des fichiers et des données) de la machine physique.

Dans quelles situations la virtualisation des serveurs est bonne?

Lorsque vous avez plusieurs serveurs physiques sous-utilisés envisager la virtualisation. Vous pouvez avoir un serveur dédié à un seul utilisateur ou un petit groupe, qui utilise seulement une petite fraction des ressources du serveur (CPU, mémoire, stockage).

Les serveurs physiques occupent de l'espace physique, consomment des ressources énergétiques, et nécessitent de refroidissement. Considérer qu'un serveur physique à l'aide, au maximum, 20% de ce serveur de CPU, la mémoire, le stockage et la bande passante du réseau est encore en utilisant 100% des autres composants de ce serveur (ventilateurs, LED, dispositifs optiques, carte (s) d'interface réseau, et de la puissance la fourniture). Cette surcharge physique peut être réduite par la consolidation de ces serveurs performants dans un ensemble de serveurs virtuels hébergés par un grand serveur physique.

La virtualisation de postes peut être utile en fonction de votre application. Lorsqu'il est combiné avec la technologie de client léger (voir ci-dessous), desservant les postes de travail a pratiquement un grand potentiel en termes de réduction du nombre d'ordinateurs personnels physiques. Les coûts globaux du matériel sont considérablement réduits, parce que toutes les demandes sont traitées sur un serveur, ce qui signifie que les ordinateurs (ou clients légers), les utilisateurs utilisent pour se connecter au serveur peut être utilisé pour plus long que le cycle de remplacement standard.

La virtualisation est grande pour le logiciel run-of-the-mill et la productivité de bureau, applications spécialisées (comme les logiciels scientifiques ou d'ingénierie), souvent ne fonctionnent pas bien dans un environnement virtualisé.

2.2.8. BIOS Settings

Dans la plupart des cas, vous pouvez régler les paramètres d'économie d'énergie dans le menu BIOS d'un ordinateur.

- Les paramètres BIOS auront la priorité sur les paramètres du logiciel.
- Les paramètres BIOS varient selon le fabricant.

En particulier, assurez-vous d'activer l'ACPI (Advanced Configuration and Power Interface) et spécifier le "S3" (Suspend to RAM) paramètre ACPI pour permettre des fonctionnalités de gestion d'énergie plus avancées. Vous pouvez spécifier plusieurs paramètres de gestion de l'alimentation au niveau du BIOS, mais contrôler plus fin est typiquement obtenu par l'intermédiaire du système d'exploitation ou des logiciels tiers.

2.2.9. Software Power Options

Nous vous recommandons de changer le paramètre de gestion de l'alimentation de sorte que le moniteur est éteint après une certaine période d'inactivité, comme 20 minutes. (Pour les ordinateurs de classe où une présentation ou une vidéo peut être affichée avec aucune entrée d'utilisateur pour un temps plus long, nous recommandons un réglage de 60 minutes ou plus.)

De même, modifiez le paramètre gestion de l'alimentation qui désactive les disques durs pour correspondre au réglage de l'écran.

2.2.10. Automatic Power Off/On

La plupart des systèmes d'exploitation prennent en charge une fonction de mise en veille prolongée, où le contenu de la RAM est écrit sur le disque dur avant d'éteindre l'ordinateur. Lorsque l'ordinateur est rallumé, il lit le contenu du cache disque vers la RAM et se restaure à l'état où il était avant l'hibernation.

Un ordinateur hiberner n'utilise pas plus de puissance qu'un ordinateur qui est mis hors tension. Nous vous recommandons d'activer cette fonctionnalité sur la plupart des ordinateurs. Encore une fois, examiner soigneusement l'utilisation appropriée de cette technologie pour chaque utilisateur et ordinateur; par exemple, un laboratoire informatique ne peut pas être un endroit approprié pour ce paramètre. Le système d'exploitation devrait être en mesure d'invoquer automatiquement la période d'hibernation du système après un certain laps de temps est écoulé sans aucune activité de l'utilisateur.

Dans des laboratoires d'informatique, dont la plupart fonctionnent 24 heures, il y a des ordinateurs qui consomment de l'énergie pour une grande partie de la nuit lorsqu'ils ne sont pas en cours d'utilisation. Il semble que ce serait une application idéale pour une fonction automatique mise en veille prolongée. Un script automatisé qui éteindre l'ordinateur, si deux conditions sont remplies:

Après un certain moment de la journée (par exemple a 22:00h) l'ordinateur doit avoir été inactif pendant au moins 60 minutes. En outre un paramètre BIOS pour permettre à l'ordinateur alimenter automatiquement à une certaine heure de la journée (7:00h)

2.2.11. Consumables

Le logiciel peut être utilisé pour atténuer les impressions inutiles et le gaspillage. Une application freeware appelé Green Print, par exemple, identifie et supprime automatiquement les pages inutiles ou espace blanc à partir des impressions avant qu'ils ne soient imprimés, résultant en moins de papier et de l'encre ou de l'utilisation de toner.

2.2.12. Upgrade with Efficient Components

Mise à niveau des composants inefficaces à l'intérieur d'un ordinateur peut améliorer sont efficacités globale, bien que le coût plus élevé est parfois un facteur interdisant. Avec des mises à niveau de composants nécessitant parfois d'autres composants requis pour être remplacés en premier.

Un coût de remplacement plus efficace pour les mises à niveau des composants est de chercher délibérément l'ordinateur le plus écologique disponible quand vient le temps pour le remplacement [17].

2.3. Stratégies green pour les Hardware

- **Peripherals**

Les moniteurs sont une source évidente d'économies d'énergie. Retire ces vieux moniteurs CRT et les remplacer par des moniteurs LCD, Remplacement d'un "moniteur CRT en utilisant 95 watts avec un moniteur LCD utilisant 30 watts réalisera une économie de 20 $ par année. Vos utilisateurs vous en seront reconnaissants pour l'espace de bureau supplémentaire, aussi bien.

- **Printers**

Le réglage de la mise hors tension automatique le temps de 30 minutes est un bon compromis entre les économies d'énergie et les attentes de l'équipement qui répond à leurs besoins des utilisateurs. Si vous avez plusieurs imprimantes du même modèle, un outil de gestion de logiciel peut être disponible auprès du fabricant qui permet de nombreuses imprimantes à gérer à partir d'une interface centrale, qui peut vous faire économiser un peu de temps à configurer les imprimantes.

- **Audio/Visual Equipment**

Certains équipements, notamment des projecteurs vidéo, peut être contrôlé avec une connexion RS-232 et un équipement de contrôle appropriée pour activer ou désactiver à intervalles prédéfinis ou après l'entrée d'un dispositif tel qu'un contrôleur de l'écran tactile.

Pour les équipements électroniques "muets", tels que les haut-parleurs, une bande de puissance intelligente peut être utilisée pour éteindre cet équipement par le contrôleur quand il n’est pas en cours d'utilisation projecteurs vidéo ont généralement une mise hors tension

automatique (souvent appelé "mode veille") de réglage qui peut être ajustée en fonction de vos besoins. Certains projecteurs ont un réglage qui permet à la lampe pour alimenter automatiquement lorsqu'aucun signal d'entrée n'a été détecté après un certain laps de temps. Ce paramètre est préférable à une minuterie de sommeil simple.

Vous devez évaluer ce que les réglages automatiques appropriées de mise hors tension sont pour chaque chambre; par exemple, vous ne voulez pas d'un vidéo projecteur dans une classe d'une heure pour éteindre après 15 minutes. En règle générale, un cadre de 60 ou 90 minutes est une puissance automatique raisonnable hors du temps.

3. Domaines d'action permettant à une entreprise d'intégrer une démarche de green IT

3.1. Réaliser des économies d'énergie

Les technologies de l'information et de la communication consomment 13,5% de l'électricité du pays et sont responsables au titre de 5% des émissions de CO2. [37]

3.1.1. A l'échelle d'une grande entreprise

Beaucoup de grands groupes s'intéressent au green IT car, au-delà d'intégrer une démarche de développement durable, ils peuvent réduire leurs coûts énergétiques. Ils sont, par exemple, plus vigilants sur la consommation électrique de certains matériels comme les serveurs, ou font l'acquisition de logiciels permettant de piloter la consommation électrique d'un parc informatique (diagnostic automatique du temps réel d'utilisation des équipements informatiques par jour, calcul prévisionnel des économies d'énergie réalisables en définissant une politique énergétique de mise en veille, etc.).

Par ailleurs, certaines grandes entreprises s'orientent vers la virtualisation. En effet, des solutions permettent maintenant de faire tourner plusieurs serveurs « virtuels » sur un même serveur. Bien que nécessitant des serveurs plus puissants, cette approche limite la consommation électrique et la production de chaleur induite.

3.1.2. A l'échelle d'une PME

Seules des entreprises disposant d'un volume important de matériels informatiques pourront constater de réelles économies d'énergie. Les mesures prises par les PME ne réduisent pas significativement le montant de leur facture électrique.

Toutefois, même si elles ne peuvent espérer en retirer un intérêt pécuniaire, les PME peuvent, elles aussi, faire un geste pour préserver l'environnement :

- Eteindre les postes informatiques le soir

- Paramétrer les ordinateurs pour qu'ils se mettent en veille automatiquement après un certain temps de non utilisation
- Utiliser des imprimantes se mettant automatiquement en veille après un certain temps de non utilisation
- Remplacer les écrans à tube cathodique par des écrans plats.

3.1.3. Gérer le cycle de vie du matériel informatique

3.1.4. Le matériel d'occasion reconditionné

Certaines entreprises changent leur équipement informatique lorsque le délai de garantie se termine. Des vendeurs d'occasion appelés « brokers » rachètent ce matériel et revendent celui qui est en bon état assorti d'une garantie de 6 mois à 3 ans. Les équipements fragiles - ordinateurs portables ou écrans plats – font l'objet d'une attention particulière. L'achat de matériel d'occasion reconditionné évite de déclencher inutilement la fabrication d'un matériel neuf qui deviendra à son tour un déchet. Par ailleurs, une économie de 60 à 80% est réalisée par rapport à l'achat de matériel neuf.

3.1.5. Le matériel informatique « éco-conçu »

Le matériel neuf « éco-conçu » contient généralement moins de substances toxiques et utilise des matières premières recyclées. Il est conçu pour être plus facilement recyclable, pour fonctionner plus longtemps, et, en fin de vie, pour être recyclé plus facilement.

- **Le recyclage**

➢ Le recyclage du matériel informatique :

Depuis le 13 août 2005 et l'adoption de la directive DEEE (Déchets d'Equipement Electriques et Electroniques), les constructeurs/producteurs informatiques sont tenus de prendre en charge le cycle de fin de vie du matériel et son recyclage. Les entreprises peuvent donc s'appuyer sur ces derniers qui devront mettre en œuvre un plan de prise en charge de l'appareil mis au rebus (ordinateurs de bureau, serveurs, périphériques, composants...). Les principales étapes de ce processus sont : la collecte, le stockage et l'acheminement vers des lieux de traitements appropriés.

Néanmoins, les fournisseurs ne sont pas tenus de proposer un service de recyclage pour les produits achetés avant 2005. Souvent, les entreprises se séparent de leur matériel informatique car leur performance est devenue insuffisante, bien que ce matériel soit toujours en état de fonctionner

Avant d'envisager le recyclage, pensez à céder votre matériel mis au rebus à un organisme caritatif ou un revendeur de matériel informatique d'occasion (broker).

Si toutefois votre matériel ne trouvait pas d'acquéreur, il est possible de le faire recycler par une entreprise spécialisée. L'ADEME (Agence de l'Environnement et de la Maîtrise de l'Energie) a créé une base de données des sites de retraitement des déchets dangereux.

- Le recyclage des consommables :

Certains fournisseurs de consommables les reprennent gratuitement au moyen d'enveloppes prépayées. Des sociétés spécialisées dans la collecte de consommables usagés se chargent de les reconditionner (ancienne cartouche d'encre remplie) ou de les traiter comme déchet ultime.

3.2. Réduire les déplacements grâce aux TIC

3.2.1. Le télétravail

Il s'agit de permettre aux salariés de travailler quelques jours par semaine à domicile ou dans un télécentre8 à proximité de leur domicile. Les technologies de l'information et de la communication permettent aujourd'hui de travailler à distance, dans les mêmes conditions qu'un collaborateur physiquement présent en entreprise.

3.2.2. Le travail collaboratif

Au lieu d'organiser des réunions physiques régulières avec des partenaires, pourquoi ne pas dématérialiser ces échanges ? Les plateformes de partage de documents et la visio/web conférence permettent à des équipes distantes de travailler ensemble en limitant les déplacements professionnels.

3.2.3. Le covoiturage

De plus en plus de covoiturages s'organisent grâce à des portails dédiés sur Internet. Ces plateformes ont pour but de mettre en relation des personnes assurant les mêmes itinéraires.

4. L'optimisation d'itinéraire

Certains logiciels permettent d'optimiser le trajet d'un commercial ou celui d'un livreur à partir des renseignements saisis sur la base de données clients.

4.1. Dématérialiser les échanges

La numérisation des documents administratifs (devis, factures, déclarations,…) permet de réduire la production de supports papier. Afin de garantir l'authenticité de document numérique, il convient d'utiliser des certificats électroniques.

5. Conclusion

Aujourd'hui en 2016, presque tous les constructeurs informatiques ont adopté cette démarche de développement, nommée « Green IT ».

Le Green IT n'est pas seulement une façon de penser ou une promesse, ce sont des actes quotidiens qui permettent de réduire l'empreinte de carbone sur la planète, la réduction des coûts énergétiques…

Suite l'apparition du Green IT les entreprises ont dépensé des sommes astronomiques pour mettre à jour leur parc informatique, mais aussi responsabiliser leur collaborateurs, par le biais d'informations, de formations (newsletter, affiches).

Pour que les entreprises puissent réduire leur coûts il faut que tous les collaborateurs contribuent à une démarche de développement durable. L'objectif du Green IT, il est nécessaire de mesurer et d'améliorer les performances énergétiques de l'outil informatique des entreprises au niveau mondial.

CHAPITRE 4

IMPLANTATION DE LA MÉTHODOLOGIE SCRUM au sein d'une équipe de développement une application WEB

Dans les chapitres précédents nous avons présenté toutes les connaissances requises reliées à SCRUM, et les techniques de Green Computing. A cet effet, ce chapitre présentera un moyen d'implanter cette méthodologie basé sur les paramètres Green au sein d'une équipe de développement d'un projet.

1. Implanter SCRUM

La méthode a été proposée par l'équipe de développement. L'équipe et toutes les parties concernées du projet ont participé à une formation sur la méthodologie SCRUM. Après quoi, ils ont décidé d'adopter cette méthode de travail.

1.1. Sélectionner les intervenants

Premièrement, un gestionnaire de produit connaissant les caractéristiques et l'utilité du produit final. Il détiendra tout le pouvoir décisionnel concernant le produit et guidera l'équipe dans la conception et la réalisation du produit final.

Le gestionnaire de produit passera beaucoup de temps à analyser et détailler le carnet de produits. De plus, il devra répondre aux questions des autres membres de l'équipe, et ce, quotidiennement, il approuvera aussi régulièrement des spécifications selon les échéanciers. Il est donc fortement suggéré de libérer cette personne de ses tâches régulières.

Dans notre projet, et après la discussion entre le gestionnaire de produit et le client, nous sommes arrivé a ces user stories :

ID	user Story
1	En tant qu'utilisateur j'ai besoin d'une page de démarrage pour me authentifier les createurs de site,
2	En tant qu'utilisateur, je dois vous abonner au site
3	en tant qu'utilisateur je voudrais que les livres sont repartie selon les categories
4	En tant qu'utilisateur, je veux que la Capacité de pannier est de 10 livres
5	en tant qu' utilisateur je voudrais que le paiment soit enligne les API
6	en tant qu' utilisateur je voudrai , un document guide pour m' aider à comprendre le site
7	en tant qu' utilisateur je voudrais une petite description de livre avant d'achetter
8	en tant qu'utilisateur je peut ajouter ou supprimer les livres du panier
9	en tant qu'utilisateur je peut contacter l'administrateur de sit web via un email ou un numero de telephone
10	en tant qu' utilisateur je peut pas voire le contenu de pannier des autre utilisateurs
11	en tant qu' administrateur je voudrais accedes a mon site web via un email et un mot de passe
12	en tant qu' administrateur je voudrais stocker mes livres sur une base de donnée distante pour economiser la consomation d'energie de mon pc
13	en tant qu' administrateur je voudrais utiliser des parametres et des outils pour economiser l'energie et le temps
14	en tant qu' administrateur jene veut plus utiliser trop des couleur et des images, mais un site web simple
15	en tant qu' utilisateur je voudrais voire tout les nouveau livres dans le site
16	en tant qu' administrateur je voudrais ajouter ou supprimer des livres de mon site
17	en tant qu' administrateur je voudrai un site web de haute capacity
18	en tant qu'utilisateur je veut que la recherche d'un ouvrage se fait via d'un moteur de recherche,
19	en tant que administrateur de site je voudrais un site e-commerce qui ne consomme pas trop d'energie et d'espace mémoire

Figure 4.1 liste des user stories

Deuxièmement, le maître SCRUM doit être sélectionné et idéalement sa participation devrait être volontaire. Qu'il soit développeur, architecte ou ingénieur, ce membre clé devra partager son temps pour effectuer les tâches du maître SCRUM et celle de développeur. Selon la portée du projet, la quantité de travail du maître SCRUM variera, mais pourrait accaparer tout son temps. Dans ce dernier cas, il est possible qu'il ne développe pas le produit, mais assumerait tout de même le rôle. La personne sélectionnée devra aussi être libérée temporairement de ses fonctions régulières pour concentrer son attention sur le travail exigeant de maître SCRUM.

Troisièmement, l'équipe de développement doit être choisie en fonction de leurs intérêts pour développer le produit et leur ouverture d'esprit (après tout ils devront utiliser la méthodologie SCRUM et l'adopter).

1.2. Itération 0

L'itération 0 est le début de l'utilisation de SCRUM. Elle est la fondation des itérations à venir, elle doit donc être soigneusement exécutée. Dans cette section nous aborderons l'ensemble des étapes de l'itération 0 présenté à la **figure 4.2**

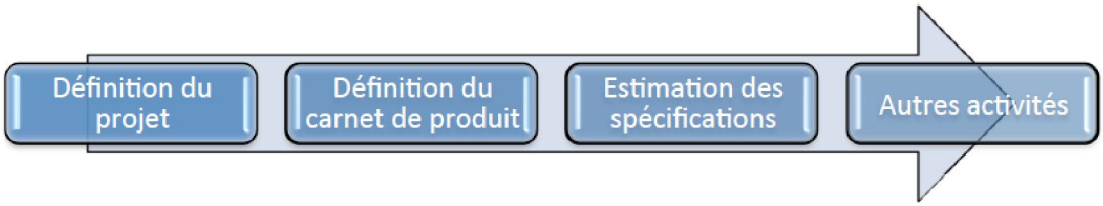

Figure 4.2 Progression des étapes de l'itération 0

1.2.1. Définition du projet

Maintenant que l'équipe est officiellement formée, la première itération demande une préparation particulière pour sa mise en place et celles qui la suivront. La première tâche

d'une équipe SCRUM est de définir le projet, a lui seul, le gestionnaire de produit pourrait s'acquitter de cette tâche, néanmoins lorsqu'elle est effectuée en équipe, celle-ci aura une meilleure compréhension du produit et des demandes qui en découleront.

La définition du projet consiste à définir les objectifs et les risques potentiels. Cette étape est optionnelle, mais peut aider l'équipe à amorcer leurs collaborations. Notre projet AMABOOK est la construction d'un site web e-commerce basé sur les paramètres Green Computing, qui fournit comme service, la vente des livres en ligne, selon les besoin et les conditions d'un client, dans le but de :

- Vendre des services en ligne, et optimiser l'énergie et le temps.
- Fidéliser ses clients
- Promouvoir un produit, un service ou un événement
- Donner une information

1.2.2. Le carnet de produit

Ayant une meilleure idée du produit ainsi que de l'objectif du produit à atteindre, l'équipe et le gestionnaire de produit peuvent passer à la création d'un carnet de produit de haut niveau. Encore une fois, cette tâche pourrait être fait exclusivement par le gestionnaire de produit, mais lorsque l'équipe est impliquée on voit un niveau de compréhension du projet supérieur ce qui aide l'équipe à se mobiliser.

Dans notre cas et parmi les besoin de client, et après la négociation avec le gestionnaire de produit et l'équipe nous sommes arrivés à ce carnet de produit :

task name	conseils basé sur Green Computing
choisir un outil pour la creation et la gestion de la base de donnée	PHPMyAdmin et Mysql server,
choisir le type de la base de donnée	base de donnée de type MYISAM
créer les tables: books, users, et categories,	
créer la structure de colomn	les types des variable doit etre specifier exacte
utiliser la methode d'i'ndexation	pour minimiser le temp de recherche
créer les relations entre les tables	
la base de donnée et distante	notion de cloud server
verifier, et sauvegarder le travail	
choisir le langage de programmation pour le coté server	le langage PHP , open source
choisir le langage de programmation pour le coté client	ajax et javascript, pour diminuer les temps de latence, et rapide dans l'execution
créer le CRUD pour chaque tables	separer les fichier CRUD,pour diminuer le temp des requettes
utiliser la technique Cache memory pour le coté server	pour diminuer la consomation d'energie de serveur
utiliser la technique COOKIES pour le coté client	pour diminuer la consomation d'energie de serveur
separer les fichier des images et les choses statiques de la base de donnée	pour minimiser la consomation de capacité de la base de donnée
utiliser Apache HTTP server	C'est gratuit, fiable et facile à configurer, plus repondu, et execute le php
choisir la methode de paiement	paiement electronique onutilisant les API pour diminuer la consomation de l'espace de la l
contruire la maquette	utiliser que 2 couleurs, bleu et blanc, pour economiser l'energie, et utiliser la technique m
tester , valider le travail et construire un rapport final , et mise en ligne le site	

Figure 4.2 Product backlog

Le gestionnaire de produit peut découper les spécifications de haut niveau vers des Spécifications plus précises.

1.2.2. Estimation des spécifications

Le gestionnaire de produit a fourni à l'équipe une liste de spécifications toutefois, il n'a aucune idée de l'effort nécessaire pour les développer. L'équipe est en mesure de fournir un nombre d'heure au gestionnaire de produit.

Sprint	start date	end date	hour working duration	group meeting	stakeholders meeting
	08/04/2016	09/04/2016	8		4
	10/04/2016	11/04/2016	6		4
0	12/04/2016	26/04/2016	60	14	14
1	27/04/2016	05/05/2016	32	4	6
2	06/05/2016	06/06/2016	120	10	10
3	07/06/2016	21/06/2016	60	4	10
4	22/06/2016	01/07/2016	40	8	12

Figure 4.3 temps nécessaires pour chaque itération

1.3. La première itération

La première itération d'une équipe SCRUM peut être exigeante et il est donc conseillé de s'assurer de suivre les règles de SCRUM à la lettre.

Par exemple :

- Durant la mêlée quotidienne, les membres de l'équipe doivent répondre aux questions les uns après les autres dans le délai prescrit de 15 minutes;
- Les spécifications doivent être développées simplement, sans initiative du développeur, dans le doute le développeur doit vérifier avec le gestionnaire de produit;
- Développer une dynamique d'équipe qui se fait dans le respect mutuel des membres.

Dans notre projet AMABOOK, durant cette itération l'équipe Scrum doit créer une base de donné qui porte les caractéristiques suivantes :

- Le SGBD utilisé : est MySQL à cause des caractéristiques suivantes
  - Rapide : Le serveur MySQL est très rapide. Des tests de performances sont disponibles sur le site de MySQL
  - Cout : l'utilisation de MySQL est gratuite.
- L'interface d'administration pour le SGBD MySQL à base de donnée est le phpMyAdmin à cause :

Il existe plusieurs façons d'accéder à la base de données et d'y faire des modifications. On peut utiliser une ligne de commande (console), exécuter les requêtes en PHP ou faire appel à un programme.

Ici, je vous propose le phpMyAdmin, un des outils les plus connus permettant de manipuler une base de données MySQL.

PhpMyAdmin n'est pas un programme mais un ensemble de page PHP toutes prêtes dont on se sert pour gagner du temps.

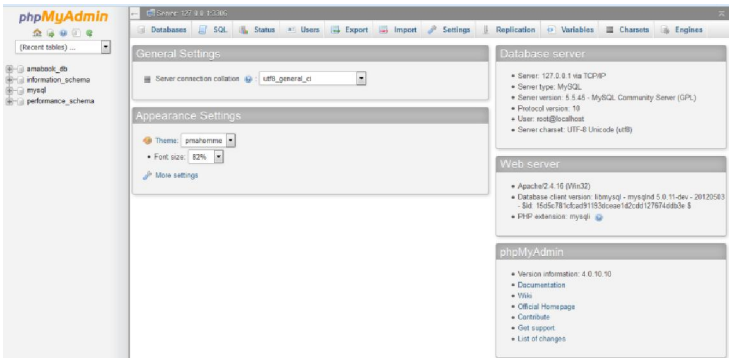

Figure 4.4 interface de PhpMyAdmin

- base de donnée avec un moteur de stockage de type MyISAM : MyISAM est un moteur de stockage du SGBDR MySQL, il est Très rapide pour effectuer des requêtes.

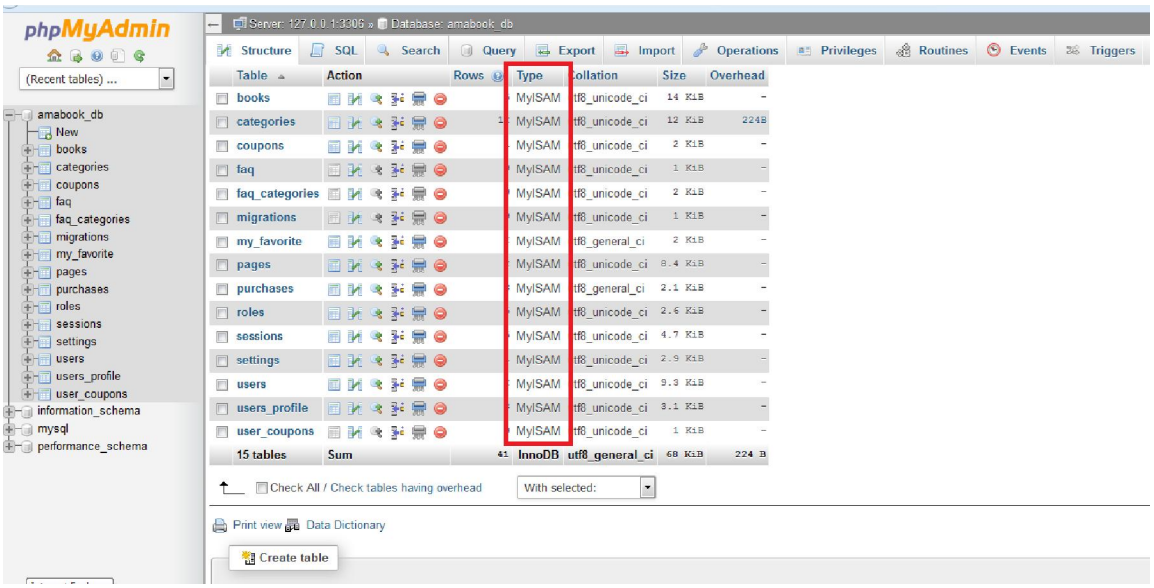

Figure 4.5 représente le moteur de recherche MyISAM

Utiliser la notion de Cloud server : Amazon Cloud Storage offre aux développeurs et aux organisations informatiques de disponibilité de stockage d'objets, tout en gardant vos coûts, et votre espace mémoire.

Pour utiliser un espace de stockage dans Amazon, il faut utiliser l'instruction suivante :

```php
/**
 * Store a newly created Category.
 *
 * @return Response
 */
public function store(CreateBookRequest $request)
{
    $books                  = new Books;
    $books->book_title      = $request->book_title;
    $books->book_alias      = $request->book_alias ? Str::slug($request->book_alias) : Str::slug($request->book_title);
    $books->book_price      = $request->book_price;
    $books->category_id     = $request->category_id;
    $books->book_description = $request->book_description;
    $books->book_status     = $request->get('book_status') ? 'enabled' : 'disabled';

    if (null !== $request->file('book_filename')) {
        $file      = $request->file('book_filename');
        $extension = $file->getClientOriginalExtension();
        Storage::disk('s3')->put('books/' . $file->getFilename() . '.' . $extension, File::get($file));
        $books->mime          = $file->getClientMimeType();
        $books->book_filename = $file->getFilename() . '.' . $extension;
    }
    $books->save();

    return Redirect::action('administrator\BooksManagementController@index')
        ->with('success', 'Le livre ajouté avec succès');
}

/**
 * Show the form for editing the specified resource.
 *
```

Figure 4.6 représente l'instruction utiliser pour réservé un espace de stockage AMAZON

Et pour consulté les fichiers stocker dans Amazon, il faut utiliser l'instruction suivante :

```php
public function get(Request $request, $book_alias)
{
    $user         = $request->user();
    $user_profile = Users_profile::where('user_id', $user->id)->first();
    $entry        = Books::where('book_alias', '=', $book_alias)->first();
    $user_balance = $user_profile->user_balance;
    $book_price   = $entry->book_price;
    if ($book_price <= $user_balance) {
        $user_profile->user_balance = $user_profile->user_balance - $book_price;
        $file                       = Storage::disk('s3')->get('/books/' . $entry->book_filename);
        if ($user_profile->save()) {
            $purchases = new Purchases;
            $purchases->book_id = $entry->book_id;
            $purchases->user_id = $user->id;
            $purchases->save();
            return redirect()->route('download_file', ['book_alias' => $book_alias]);
        }
    }
    return back()->with('success', 'Désolé votre solde est bas pour faire cette opération!');
}
```

Figure 4.7 représente l'instruction utilisée pour consulter les fichiers stockés dans AMAZON

1.4. La deuxième itération

Cette itération est resservie pour le développement de site AMABOOK, et porte les conseils Green suivantes :

- le langage de programmation pour le coté server est le langage PHP : est un langage de script, exécuté du côté serveur et open source.

  PHP est excellent pour des réalisations qui nécessitent la perfection. La qualité d'une réalisation doit être mise en regard du coût associé. Et là, le couple qualité/coût de PHP écrase toutes les autres solutions.

- le langage de programmation pour le coté client est JavaScript et la technique Ajax:
  Le JavaScript est pour :
  - Vitesse : Les fonctions du JavaScript ne doivent pas attendre pour des réponses de leurs serveurs pour agir, ce qui accélère l'ouverture des sites web.
  - Versatilité : Le JavaScript ne nécessite pas un programme spécial pour l'interpréter (Flash Player, "plug-ins"), ni pour l'écrire. De plus, JavaScript n'occupe pas un grand espace sur les sites web.

  Et la technique Ajax est pour :
  - Contient un ensemble de technologies destinées à réaliser de rapides mises à jour du contenu d'une page Web, sans qu'elles nécessitent le moindre rechargement visible par l'utilisateur de la page Web.
  - Une rapidité d'exécution car seules les données à modifier.
- séparer les fichiers CRUD de chaque table : sépare les fichiers Create, Read, Update, delate des tables pour diminuer le temp des requêtes.
- utiliser les techniques cache Memory et cookies :

Cache Memory: une mémoire qui enregistre temporairement des copies de données provenant d'une source, afin de diminuer le temps d'un nouvel accès (en lecture) d'un matériel informatique (en général, un processeur) à ces données.

Les cookies sont des fichiers de texte qui sont téléchargés sur le terminal de l'utilisateur. Et qui sont enregistrés sur la mémoire de votre navigateur. Elles optimiser le temps de réponse d'une requête.

- séparer les images et les fichiers statiques de la base de données.
- utiliser la méthode de paiement électronique :

1.5. La troisième itération

Dans cette itération l'équipe scrum construit la maquette de site AMABOOK, la maquette que nous avons choisie contient peu de couleurs qui ne consomment pas beaucoup d'énergie telle que le vert claire et le bleu, selon la maquette suivante.

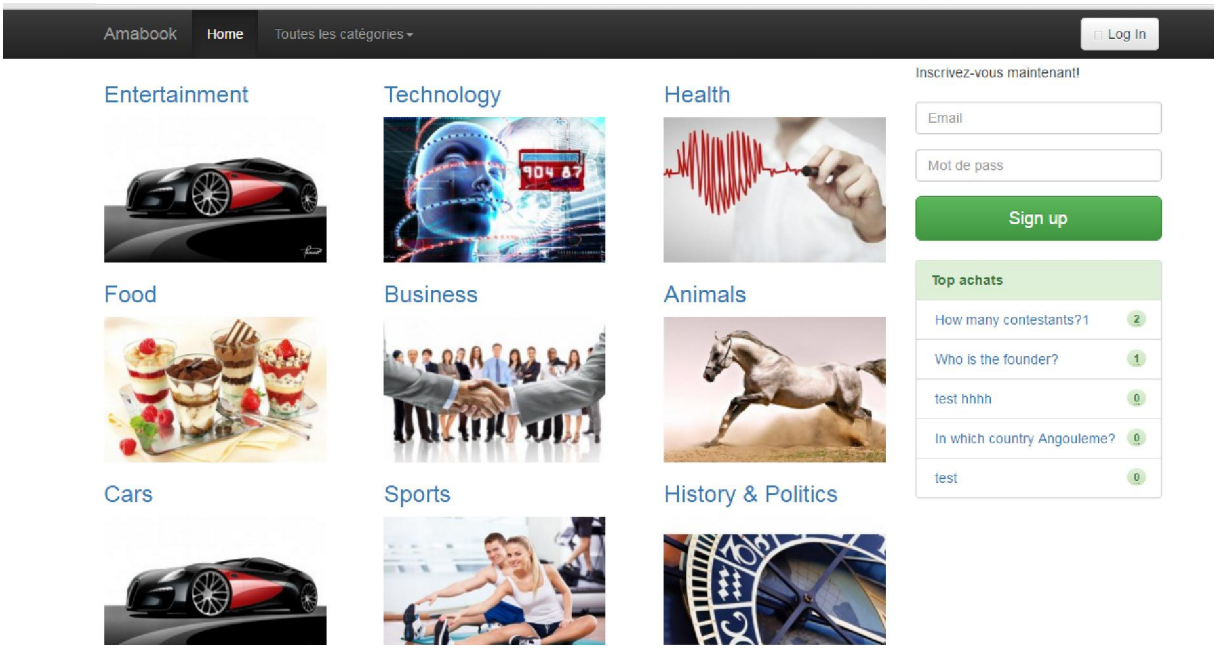

Figure 4.8 l'interface d'application web AMABOOK

- utiliser la technique minified : est pour compresser les fichiers css et JavaScript afin d'optimiser d'espace de stockage, par exemple dans ce fichier unminified qui occupe un espace de 142 Ko.

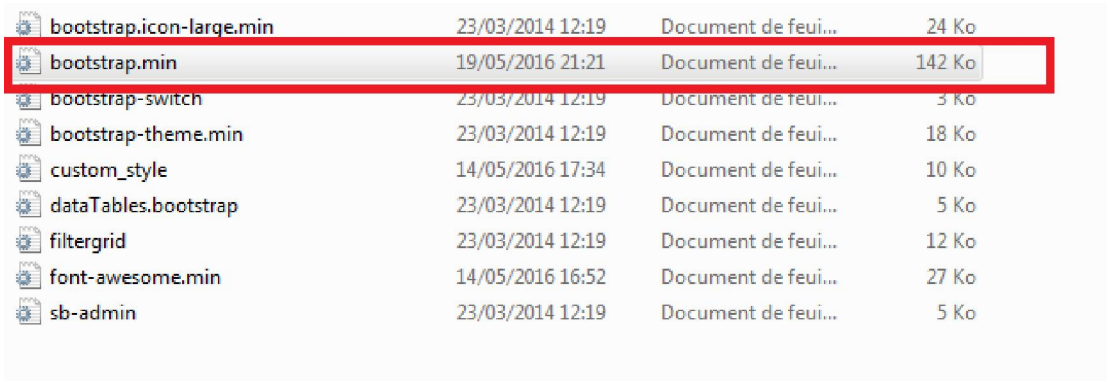

Figure4.9 représente un fichier avant l'utilisation de technique minified

Après l'utilisation de la technique minified sur le même fichier :

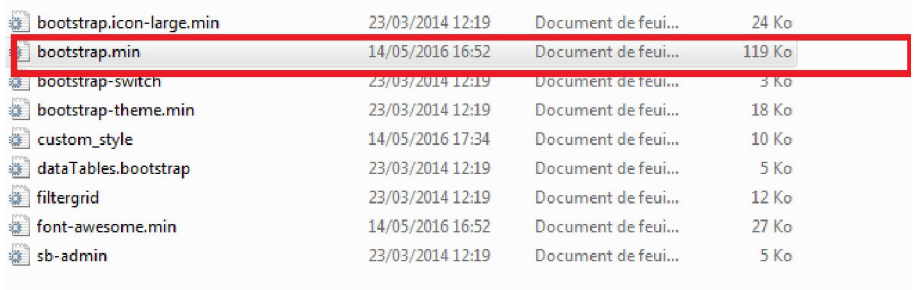

Figure4.10 représente un fichier après l'utilisation de technique minified

Nous avons gagné (143-119= 23Ko) d'espace de stockage, imaginons que nous avons1000 utilisateurs qui accèdent au site AMABOOK au même temps par exemple.

1.6. Quatrième Itération

Est réservé pour tester, valider le travail et construire un rapport final, et mise en ligne le site.

2. CONCLUSION

SCRUM est un changement de mentalité drastique pour plusieurs développeurs. Il demande de la transparence dans les interventions avec le gestionnaire de produit, de la collaboration entre les membres d'une équipe et demande aux individus une adaptation constante aux nouvelles règles d'équipe.

D'un autre côté la méthodologie SCRUM est flexible, à chaque itération elles consistent d'adapter le produit en fonction des nouvelles spécifications.

De plus, elle permet aux entreprises de livrer le produit au moment voulu puisqu'à la fin de chaque itération le produit est livrable et fonctionnel. Les membres de l'équipe profitent de l'expérience acquise durant les itérations passées et se mobilise dans un seul but commun.

La haute performance et l'économie d'énergie sont des objectifs contradictoires dans Green Computing.

L'énergie peut aussi être conservé, au cours du cycle de vie du développement logiciel telles que l'analyse de logiciels et de conception, en appliquant différents aspects et techniques green tel que le Cloud server ou la technique minified …

C'est ce que nous avons proposé durant les étapes de notre recherche une composition agile Scrum on utilisant des paramètres Green Computing au sein d'une équipe de développement d'un logiciel.

CONCLUSION GENERALE

Les approches traditionnelles ont encore leur place et demeurent un bon choix pourcertaines organisations. Cependant, elles offrent peu de solutions aux petites équipes etrestent accrochées à un processus complexe. Elles sont prédictives et l'adéquation desréalisations avec la planification est un gage de succès.

SCRUM propose d'alléger le processus et comporte différents intérêts. Pour l'équipe d e développement, c'est une meilleure collaboration avec ses collègues. Pour le produit, c'est un rehaussement de qualité et d'évolutivité. Pour la gestion de projet, c'est le contrôle de l'échéancier et du budget. Pour le client, c'est d'obtenir un système opérationnel valable rapidement.

Ces méthodes sont adaptatives et intègrent les changements pour maximiserl'efficacité du produit dans sa mission.

Dans notre travail j'ai proposé une composition SCRUM ont utilisons la notion de Green Computing pour améliorer les performances de l'équipe et minimiser l'impact de l'application sur l'écosystème. En fin, nous avons essayé une démarche pour développer une application WEB

Je conclus en mentionnant que je suis satisfait du travail accompli et que j'ai la sensation d'avoir fait progresser les choses. J'ai appliqué les éléments découverts au cours de mes lectures et ce, au profit de mon travail.

Je n'aurais pas été en mesure d'implanter ces éléments avec autant de convictions, sans connaître les solides théories supportantl'agilité. Avec un peu de recul, je ne cherche plus à convaincre mais, seulement à présenter l'agilité et le Green Computing.

Références Bibliographiques

[1] Highsmith, J. *Agile Software* ***Developement Ecosystems.*** Addison-Wesley, Boston, 2002.

[2] Alliance agile. Manifesto for Agile Software Development. Sur le site officiel de *l'alliance agile (http://www.agilemanifesto.org/)* consulté en mars 2016.

[3] Fowler, M. The Agile Manifesto: where it came from and where it may go. Sur le site de ***Martin Fowler*** *(http://martinfowler.com/articles/agileStory.html*) consulté en mars 2016.

[4] analyse des principes du génie logiciel au niveau du developement. P39.

[5] Cockburn, A. ***Agile Software Development.*** Addison-Wesley Professional, 2001.

[6] Fowler, M. ***and*** **Highsmith, J.** The agile manifesto. 2001. Sur le site de ***Dr. Dobb's portal*** *(http://www.ddj.com/dept/architect/184414755*) ***consulté en mars 2016.***

[7] Augustine, S. and Woodcock, S. Agile Project Management, 2003. Sur le site de ***CC Pace inc****. (www.ccpace.com)* consulté en mars 2016.

[8] Cockburn, A. Characterizing people as non-linear, first-order components in software development. ***Proceeding at the 4th International Multi-Conference on Systems, Cybernetics and Informatics***, Orlando, Florida, June, 2000.

[9] Maslow, A.H. A Theory of Human Motivation. 1943. Sur le site de ***Classics in the History of Psychology,*** **Originally Published in** ***Psychological Review,*** 50, 370-396.

[10] Arsenault, L. ***Dictionnaire des compétences TRIMA. Arsenault Formation Carrière, 1997.***

[11] Advanced Development Methods, Inc (ADM). Agile Processes – Emergence of Essential Systems. Sur le site officiel de ***SCRUM***

(*http://www.controlchaos.com/download/Emergence%20of%20Essential%20Systems.pdf*) consulté en mars 2016.

[12] Boehm, B. A Spiral Model of Software Development and Enhancement. ***IEEE Computer***, May 1988, p. 61-72.

[13] Deming, W. E. ***Out of the Crisis***. MIT Press, February 2, 1982.

[14] Ambler, S.W. ARE YOU AGILE OR FRAGILE - SCOTT W. AMBLER nov 27, 2014.

[15] Beauregard, F. Visual Studio Talk Show : Entrevue sur les approches 'Agiles' de développements logiciels.

[16] Highsmith, J. What Is Agile Software Development? ***CROSSTALK The Journal of Defense Software Engineering***, October 2002.

[17] Cockburn, A. *Crystal* ***Clear: A Human-Powered Methodology for Small Teams.*** Addison-Wesley, 2004.

[18] Boehm, B., Turner, R. ***Balancing Agility and Discipline: A Guide for the Perplexed***. Addison-Wesley, Boston, MA. 2004.

[19] Boehm, B. Get Ready for Agile Methods, with Care. ***IEEE Computer***, January 2002, p.64-69.

[20] Fowler, M. The new methodology. Sur le site de ***Martin Fowler***. (*http://www.martinfowler.com/articles/newMethodology.html*) consulté en mars 2016.

[21] Levine, L. Reflections On Software Agility And Agile Methods: Challenges, Dilemmas, and the Way Ahead. Site du ***Software Engineering Institute Carnegie Mellon University, Pittsburgh, PA U.S.A.***

(*http://www.sei.cmu.edu/programs/acquisition support/publications/reflections.pdf*) consulté en mars 2016.

[22] Advanced Development Methods, Inc (ADM). Controlled Chaos : Living on the Edge. 1996. Sur le site officiel de *SCRUM*

(*http://www.controlchaos.com/download/Living%20on%20the%20Edge.pdf*). Consulté en mars 216.

[23] Encyclopédie libre en ligne Wikipédia *Rapid* ***Application Development consulté en mai 2016***.

[24] Encyclopédie libre en ligne Wikipédia Scrum ***consulté en mai 2016***.

[25] Encyclopédie libre en ligne Wikipédia ***Feature Driven Development consulté en mai 2016***.

[26] Encyclopédie libre en ligne Wikipédia ***Rational Unified Process consulté en mai 2016***.

[27] Encyclopédie libre en ligne Wikipédia ***EXtreme Programming consulté en mai 2016***.

[28] «The Scrum Paper : Nuts, Bolts, and Origins of an Agile Process » En ligne. (*http://jeffsutherland.com/scrumpapers.pdf*). Consulté on mars 2016

[29] Scrum description p4

[30] Jeff Sutherland. 1993-2007. «The Scrum Paper : Nuts, Bolts, and Origins of an Agile Process » En ligne. *(http://jeffsutherland.com/scrumpapers.pdf)*. Consulté on mars 2016.

[31] Guide Léger de la Théorie et de la Pratique de Scrum p5.

[32] La méthode Agile - Optimisation de la relation "client / fournisseur".

[33] Arpad Horvath and Eric Masanet. Enterprise Strategies for Reducing the Life-Cycle Energy Use and Green house Gas Emissions of Personal Computers. Proceedings of the 2006 IEEE International Symposium on Electronics and the Environment, pp. 21-26, 2006.

[34] Andrea Prothero and James A. Fitchett. Greening Capitalism: Opportunities for a Green Commodity. Journal of Macromarketing, 20(1), pp. 46-55, 2000.

[35] Kremer U, Department of Computer Science Rutgers University (2002) Low Power/Energy Compiler Optimizations. International conference of Book power aware computing. CRC Press, 2004, ch. 35.

[36] Green Computing Strategies for Improving Energy Efficiency in IT Systems Jacob John Sinhgad Institute of Business Administration and Computer Application, Lonavala, Pune, India .en juin 2014.

[37] QU'EST-CE QUE LE GREEN IT ? entreprise lorraine.

ملخص

في السنوات الأخيرة ظهرت أساليب و طرق جديدة في سياق تطوير تطبيقات عالية الأداء.
و ناخد في بحثنا الطريقة أجيل كما نأخذ بعين الاعتبار الحوسبة الخضراء التي من اهدافها الحفاظ على الطاقة وإعادة التدوير والتقليل من نفايات الكمبيوتر.
نبدأ بحثنا مع تعريف العناصر الأساسية للطريقة اجيل ، و عناصر التشابه والاختلاف مع الأساليب التقليدية، بعدها نقدم بعض الطرق الاخرى باختصار.
ركزنا تفكيرنا على منهجية سكرم وأدوار الجهات المعنية ،و ما هي مزايا وعيوب هذه المنهجية,
وأخيرا قمنا بتجربة اقترحنا فيها المنهجية سكرم و اخدنا بعين الاعتبار الحوسبة الخضراء.
للتحقق من جدواه، اقترحنا نهجا لتطوير تطبيقات الويب.

Résumé

Depuis quelques années, les méthodes agiles ont émergées et semblent prometteuses dans le cadre de développer des applications de haute performance. D'autre part, le Green Computing est un paradigme (une manière de voir les choses, souvent présenté comme objectif du futur) qui cible l'économie d'énergie, le recyclage et la minimisation des déchets informatiques.

Notre recherche relate les travaux réalisés en but de procéder l'implantation d'une méthode agile de développement logiciel, et prend en considération les paramètres Green Computing.

Mots clé : méthodes agiles, Green Computing, SCRUM.

Abstract

In recent years, agile methods have emerged and appear promising in the context of developing high performance applications.

On the other hand, the Green Computing is a paradigm that target energy conservation, recycling and minimization of computer waste.

Our research describes the work done in order to make the implementation of an agile software development method, and considers the Green Computing parameters.

Keywords: Agile, Green Computing, SCRUM.